Stanisław Jarecki   Gene Tsudik (Eds.)

Public Key Cryptography – PKC 2009

12th International Conference
on Practice and Theory in Public Key Cryptography
Irvine, CA, USA, March 18-20, 2009
Proceedings

Springer

Volume Editors

Stanisław Jarecki
Gene Tsudik
University of California, Irvine
Computer Science Department
Irvine, CA 92697-3435, USA
E-mail: {stasio, gts}@ics.uci.edu

Library of Congress Control Number: 2009921160

CR Subject Classification (1998): E.3, F.2.1-2, C.2.0, K.4.4, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-642-00467-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-00467-4 Springer Berlin Heidelberg New York

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12631902 06/3180 5 4 3 2 1 0

Preface

It has been a real pleasure to have taken part in organizing the 12th International Conference on Practice and Theory in Public Key Cryptography (PKC 2009). PKC 2009 was held March 18-20, 2009, on the campus of the University of California, Irvine (UCI). As usual, it was sponsored by the International Association for Cryptologic Research (IACR) in cooperation with:

- UCI Secure Computing and Networking Center (SCONCE)
- UCI Donald Bren School of Information and Computer Sciences (DBSICS)
- California Institute for Telecommunications and Information Technology (CalIT2)

The PKC 2008 Program Committee (PC) consisted of 33 internationally recognized researchers with combined expertise covering the entire scope of the conference.

Recent growth in the number of cryptography venues has resulted in stiff competition for high-quality papers. Nonetheless, PKC's continued success is evident from both the number and the quality of submissions. PKC 2009 received a total of 112 submissions. They were reviewed by the PC members and a highly qualified team of external reviewers. Each submission was refereed by at least three reviewers. After deliberations by the PC, 28 submissions were accepted for presentation. Based on extensive discussions, the PKC 2009 best paper award was given to Alexander May and Maike Ritzenhofen for their paper "Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint". The conference program also included two invited talks, by Anna Lysyanskaya (Brown University) and Amit Sahai (UCLA).

A number of people selflessly contributed to the success of PKC 2009. First and foremost, we thank the authors of all submissions. They are the backbone of this conference and their confidence and support are highly appreciated. We are similarly grateful to the dedicated, knowledgeable and hard-working PC members who provided excellent reviews (on time and on a tight schedule!) and took part in post-review discussions. Their altruistic dedication and community service spirit are commendable. We are also indebted to the PKC Steering Committee members for their guidance as well as to Shai Halevi and Christian Cachin for valuable technical assistance with revewing and organizational aspects. A special word of thanks to Moti Yung for his encouragement and help in the planning stage. Last, but surely not least, we greatefully acknowledge extramural financial support (especially appreciated in these tough economic times) by Microsoft Research, Google and Qualcomm.

March 2009

Stanisław Jarecki
Gene Tsudik

Organization

General and Program Co-chairs

Stanisław Jarecki and Gene Tsudik
Computer Science Department
University of California, Irvine

Program Committee

Xavier Boyen	Voltage Security, USA
Christian Cachin	IBM Zurich Research, Switzerland
Jan Camenisch	IBM Zurich Research, Switzerland
Jung Hee Cheon	Seoul National University, South Korea
Jean-Sebastien Coron	University of Luxembourg, Luxembourg
Nelly Fazio	CUNY, USA
Bao Feng	i2R, Singapore
Pierre-Alain Fouque	ENS, France
Juan Garay	AT&T Labs – Research, USA
Rosario Gennaro	IBM T.J. Watson Research Center, USA
Amir Herzberg	Bar Ilan University, Israel
Marc Joye	Thomson R&D, France
Seny Kamara	Microsoft, USA
Eike Kiltz	CWI, The Netherlands
Aggelos Kiayias	University of Connecticut, USA
Javier Lopez	University of Malaga, Spain
Breno de Medeiros	Google, USA
David Naccache	ENS, France
Jesper Buus Nielsen	Aarhus University, Denmark
Kenny Paterson	Royal Holloway, UK
Benny Pinkas	University of Haifa, Israel
David Pointcheval	ENS-CNRS-INRIA, France
Ahmed Reza-Sadeghi	Bochum University, Germany
Rei Safavi-Naini	University of Calgary, Canada
Nitesh Saxena	NYU Polytechnic Institute, USA
Berry Schoenmakers	TU Eindhoven, The Netherlands
Hovav Shacham	UC San Diego, USA
Vitaly Shmatikov	UT Austin, USA
Igor Shparlinski	Macquarie University, Australia
Michael Steiner	IBM T.J. Watson Research Center, USA
Serge Vaudenay	EPFL, Switzerland
Ivan Visconti	University of Salerno, Italy
Suzanne Wetzel	Stevens Institute of Technology, USA

External Reviewers

Jaehyun Ahn
Adi Akavia
Martin Albrecht
Frederik Armknecht
Werner Backes
Joonsang Baek
Aurelie Bauer
Olivier Billet
Joppe Bos
Justin Brickell
David Cash
Dario Catalano
Rafik Chaabouni
Xiaofeng Chen
Carlos Cid
Christophe Clavier
Paolo D'Arco
Ivan Damgård
Yevgeniy Dodis
Anna Lisa Ferrara
Matthieu Finiasz
Martin Gagne
Steven Galbraith
David Galindo
Robert Gallant
Maribel Gonzalez-Vasco
Robert Granger
Matthew Green
Javier Herranz
Jason Hinek
Dennis Hofheinz
Sebastiaan de Hoogh
Nick Howgrave-Graham
Malika Izabachène
David Jao
Jonathan Katz
Markulf Kohlweiss
Vladimir Kolesnikov
Ralf Kuesters
Mun-kyu Lee
Arjen Lenstra
Benoit Libert
Moses Liskov
Joseph K. Liu
Hans Loehr
Gilles Macario-Rat
Mark Manulis
Alexander May
Nicolas Méloni
Jorge Nakahara
Gregory Neven
Antonio Nicolosi
Juan Gonzalez Nieto
Claudio Orlandi
Khaled Ouafi
Sylvain Pasini
Jacques Patarin
Serdar Pehlivanoglu
Kun Peng
Tal Rabin
Carla Ràfols
Pankaj Rohatgi
Thomas Schneider
Mike Scott
Igor Semaev
Siamak Shahandashti
Haya Shulman
Alice Silverberg
Thomas Sirvent
William Skeith
Rainer Steinwandt
Qiang Tang
Joe-Kai Tsay
Raylin Tso
Borhan Uddin
Dominique Unruh
Frederik Vercauteren
Jos Villegas
Felipe Voloch
Jonathan Voris
Christian Wachsmann
Daniel Wichs
Hong-Sheng Zhou

Sponsors

Financial support by the following sponsors is gratefully acknowledged:

- Microsoft Research
- Google
- Qualcomm
- Secure Computing and Networking Center (SCONCE) at UCI[1]
- California Institute for Telecommunications and Information Technology (CalIT2)
- Donald Bren School of Information and Computer Science (DBSICS) at UCI

[1] PKC 2009 support made possible by a grant from the Experian Corporation.

Table of Contents

Number Theory

Applications and Protocols

Multi-Party Protocols

Identity-Based Encryption

Signatures

Encryption

New Cryptosystems and Optimizations

Group Signatures and Anonymous Credentials

Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint*

Alexander May and Maike Ritzenhofen

Horst Görtz Institute for IT-security
Faculty of Mathematics
Ruhr-University of Bochum, 44780 Bochum, Germany
alex.may@ruhr-uni-bochum.de,
maike.ritzenhofen@ruhr-uni-bochum.de

Abstract. We address the problem of polynomial time factoring RSA moduli $N_1 = p_1q_1$ with the help of an oracle. As opposed to other approaches that require an oracle that *explicitly* outputs bits of p_1, we use an oracle that gives only *implicit* information about p_1. Namely, our oracle outputs a different $N_2 = p_2q_2$ such that p_1 and p_2 share the t least significant bits. Surprisingly, this implicit information is already sufficient to efficiently factor N_1, N_2 provided that t is large enough. We then generalize this approach to more than one oracle query.

Keywords: Factoring with an oracle, lattices.

1 Introduction

Factoring large integers is one of the most fundamental problems in algorithmic number theory and lies at the heart of RSA's security. Consequently, since the invention of RSA in 1977 [18] there have been enormous efforts for finding efficient factorization algorithms. The Quadratic Sieve [16], the Elliptic Curve Method [9] and eventually the Number Field Sieve [10] have led to a steady progress in improving the factorization complexity. However, since 1993 there is little progress from the complexity theoretic point of view when using classical Turing machines as the model of computation.

Shor's algorithm from 1994 [19] demonstrates that the factorization problem is polynomial time solvable on quantum Turing machines. Nowadays, it seems to be highly unclear whether these machines can ever be realized in practice.

The so-called *oracle complexity* of the factorization problem was first studied at Eurocrypt 1985 by Rivest and Shamir [17], who showed that $N = pq$ can be factored given an oracle that provides an attacker with bits of one of the prime

* The research leading to these results was supported by the German Research Foundation (DFG) as part of the project MA 2536/3-1 and has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number ICT-2007-216646 - European Network of Excellence in Cryptology II (ECRYPT II).

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 1–14, 2009.

factors. The task is to factor in polynomial time by asking as few as possible queries to the oracle. Rivest and Shamir showed that $\frac{3}{5}\log p$ queries suffice in order to factor efficiently.

At Eurocrypt 1992, Maurer [12] allowed for an oracle that is able to answer any type of questions by YES/NO answers. Using this powerful oracle, he showed that $\epsilon \log p$ oracle queries are sufficient for any $\epsilon > 0$ in order to factor efficiently. At Eurocrypt 1996, Coppersmith [2] in turn improved the Rivest-Shamir oracle complexity for most significant bits to $\frac{1}{2}\log p$ queries. Coppersmith used this result to break the Vanstone-Zuccherato ID-based cryptosytem [21] that leaks half of the most significant bits.

In this work, we highly restrict the power of the oracle. Namely, we allow for an oracle that on input an RSA modulus $N_1 = p_1q_1$ outputs another different RSA modulus $N_2 = p_2q_2$ such that p_1, p_2 share their t least significant bits. Moreover, we assume for notational simplicity that the bit-sizes of p_2, q_2 are equal to the bit-sizes of p_1, q_1, respectively.

Thus, as opposed to an oracle that *explicitly* outputs bits of the prime factor p_1, we only have an oracle that *implicitly* gives information about the bits of p_1. Intuitively, since N_2 is a hard to factor RSA modulus, it should not be possible to extract this *implicit* information. We show that this intuition is false. Namely, we show that the link of the factorization problems N_1 and N_2 gives rise to an efficient factorization algorithm provided that t is large enough.

More precisely, let q_1 and q_2 be α-bit numbers. Then our lattice-based algorithm provably factors N_1, N_2 with $N_1 \neq N_2$ in quadratic time whenever $t > 2(\alpha + 1)$. In order to give a numerical example: Let N_1, N_2 have 750-bit p_1, p_2 and 250-bit q_1, q_2. Then the factorization of N_1, N_2 can be efficiently found provided that p_1, p_2 share more than 502 least significant bits. The bound $t > 2(\alpha+1)$ implies that our first result works only for imbalanced RSA moduli. Namely, the prime factors p_i have to have bit-sizes larger than twice the bit-sizes of the q_i.

Using more than one oracle query, we can further improve upon the bound on t. In the case of $k-1$ queries, we obtain $N_2, \ldots, N_k$ different RSA moduli such that all p_i share the least t significant bits. This gives rise to a lattice attack with a k-dimensional lattice L having a short vector $\mathbf{q} = (q_1, \ldots, q_k)$ that immediately yields the factorization of all $N_1, \ldots, N_k$. For constant k, our algorithm runs in time polynomial in the bit-size of the RSA moduli. As opposed to our first result, in the general case we are not able to prove that our target vector $\mathbf{q}$ is a shortest vector in the lattice L. Thus, we leave this as a heuristic assumption. This heuristic is supported by a counting argument and by experimental results that demonstrate that we are almost always able to efficiently find the factorization.

Moreover, when asking $k-1$ queries for RSA moduli with α-bit q_i that share t least significant bits of the p_i, we improve our bound to $t \geq \frac{k}{k-1}\alpha$. Hence for a larger number k of queries our bound converges to $t \geq \alpha$, which means that the p_i should at least coincide on α bits, where α is the bit-size of the q_i. In the case of RSA primes of the same bit-size, this result tells us that $N_1 = p_1q_1, \ldots, N_k = p_1q_k$ with the same p_1 can efficiently be factored, which is trivially true by greatest common divisor computations. On the other hand, our result is highly non-trivial whenever the bit-sizes are not balanced.

If we do not restrict ourselves to polynomial running time, then we can easily adapt our method to also factor balanced RSA moduli. All that we have to do is to determine a small quantity of the bits of q_i by brute force search. Using these bits we can apply the previous method in order to determine at least half of the bits of all q_i. The complete factorization of *all* RSA moduli N_i is then retrieved by the aforementioned lattice-based algorithm of Coppersmith [3].

Currently, we are not aware of an RSA key generation that uses primes sharing least significant bits. The Steinfeld-Zheng system [20] uses moduli $N = pq$ such that p, q itself share least significant bits, for which our algorithm does not apply. Naturally, one application of our result is malicious key generation of RSA moduli, i.e. the construction of backdoored RSA moduli [5,22].

Another application is a novel kind of attack on a public key generator. Suppose an attacker succeeds to manipulate those t registers of an RSA public key generator that hold the least significant bits of one prime factor such that these registers are stuck to some unknown value. E.g., take an attacker that simply destroys the registers with the help of a laser beam such that he has no control on the register's values. If the RSA key parameters are within our bounds, the attacker can easily collect sufficiently many RSA moduli that allow him to factor all of them. Thus, he uses the RSA key generator as an oracle. Notice that the RSA generator will usually not even notice such an attack since the RSA moduli look innocent.

Moreover, we feel that our algorithm will be useful for *constructive* cryptographic applications as well. Consider the task that our oracle has to solve, which we call the *one more RSA modulus problem*, i.e one has to produce on input an RSA modulus $N = pq$ other moduli $N_i = p_i q_i$ whose factors p_i share their least significant bits.

Our construction shows that this problem is for many parameter settings equivalent to the factorization problem. So the *one more RSA modulus problem* might serve as a basis for various cryptographic primitives, whose security is then in turn directly based on factoring (imbalanced) integers.

In addition to potential applications, we feel that our result is of strong theoretical interest, since we show for the first time that quite surprisingly implicit information is sufficient in order to factor efficiently. In turn, this implies that already a really weak form of an oracle suffices for achieving a polynomial time factorization process. In the oracle-based line of research, reducing the number of queries and diminishing the power of the oracles is the path that leads to a better understanding of the complexity of the underlying factorization problem.

We organize our paper as follows. In Section 2, we give the necessary facts about lattices. In Section 3, we introduce our rigorous construction with one oracle query, i.e. with two RSA moduli. In Section 4, we generalize our construction to an arbitrary fixed number of queries. This makes our construction heuristic. In Section 5, we adapt our heuristic construction to the case of balanced RSA moduli. In Section 6, we experimentally confirm the validity of our heuristics.

2 Preliminaries

An integer lattice L is a discrete additive subgroup of $\mathbb{Z}^n$. An alternative equivalent definition of an integer lattice can be given via a basis.

Let $d, n \in \mathbb{N}$, $d \leq n$. Let $\mathbf{b_1}, \dots, \mathbf{b_d} \in \mathbb{Z}^n$ be linearly independent vectors. Then the set of all integer linear combinations of the $\mathbf{b_i}$ spans an integer *lattice* L, i.e.

$$L = \left\{ \sum_{i=1}^{d} a_i \mathbf{b_i} \mid a_i \in \mathbb{Z} \right\}.$$

We call $B = \begin{pmatrix} \mathbf{b_1} \\ \vdots \\ \mathbf{b_d} \end{pmatrix}$ a *basis* of the lattice, the value d denotes the *dimension* or *rank* of the basis. The lattice is said to have *full rank* if $d = n$. The *determinant* $\det(L)$ of a lattice is the volume of the parallelepiped spanned by the basis vectors. The determinant $\det(L)$ is invariant under unimodular basis transformations of B. In case of a full rank lattice $\det(L)$ is equal to the absolute value of the Gramian determinant of the basis B.

Let us denote by $||\mathbf{v}||$ the Euclidean ℓ_2-norm of a vector $\mathbf{v}$. Hadamard's inequality [13] relates the length of the basis vectors to the determinant.

Lemma 1 (Hadamard). *Let* $B = \begin{pmatrix} \mathbf{b_1} \\ \vdots \\ \mathbf{b_n} \end{pmatrix} \in \mathbb{Z}^{n \times n}$, $n \in \mathbb{N}$, *be an arbitrary non-singular matrix. Then*

$$\det(B) \leq \prod_{i=1}^{n} ||\mathbf{b_i}||.$$

The successive minima $\lambda_i(L)$ of the lattice L are defined as the minimal radius of a ball containing i linearly independent lattice vectors of L. In a two-dimensional lattice L, basis vectors $\mathbf{v_1}, \mathbf{v_2}$ with lengths $||v_1|| = \lambda_1(L)$ and $||v_2|| = \lambda_2(L)$ are efficiently computable via Gaussian reduction.

Theorem 1. *Let* $\mathbf{b_1}, \mathbf{b_2} \in \mathbb{Z}^n$ *be basis vectors of a two-dimensional lattice* L. *Then the Gauss-reduced lattice basis vectors* $\mathbf{v_1}, \mathbf{v_2}$ *can be determined in time* $O(\log^2(\max\{||\mathbf{v_1}||, ||\mathbf{v_2}||\})$. *Furthermore,*

$$||\mathbf{v_1}|| = \lambda_1(L) \text{ and } ||\mathbf{v_2}|| = \lambda_2(L).$$

Information on Gaussian reduction and its running time can be found in [13].

A shortest vector of a lattice satisfies the Minkowski bound, which relates the length of a shortest vector to the determinant and dimension of the lattice.

Theorem 2 (Minkowski [14]). *Let* $L \subseteq \mathbb{Z}^{n \times n}$ *be an integer lattice. Then* L *contains a non-zero vector* $\mathbf{v}$ *with*

$$||\mathbf{v}|| = \lambda_1(L) \leq \sqrt{n} \det(L)^{\frac{1}{n}}.$$

Vectors with short norm can be computed by the LLL algorithm of Lenstra, Lenstra, and Lovász [11].

Theorem 3 (LLL). *Let L be a d-dimensional lattice with basis $\mathbf{b_1}, \ldots, \mathbf{b_d} \in \mathbb{Z}^n$. Then the LLL algorithm outputs a reduced basis $\mathbf{v_1}, \ldots, \mathbf{v_d}$ with the following property:*

$$||\mathbf{v_1}|| \leq 2^{\frac{d-1}{4}} \det(L)^{\frac{1}{d}}.$$

The running time of this algorithm is $O(d^4 n(d+\log b_{\max}) \log b_{\max})$, where $b_{\max} \in \mathbb{N}$ denotes the largest entry in the basis matrix.

For a proof of the upper bound of a shortest LLL vector compare [11]. The running time is the running time of the so-called L^2-algorithm, an efficient LLL version due to Nguyen and Stehlé [15].

The LLL algorithm can be used for factoring integers with partly known factors as Coppersmith showed in [3].

Theorem 4 ([3] Theorem 5). *Let N be an n-bit composite number. Then we can find the factorization of $N = pq$ in polynomial time if we know the low order $\frac{n}{4}$ bits of p.*

3 Implicit Factoring of Two RSA Moduli

Assume that we are given two different RSA moduli $N_1 = p_1 q_1, N_2 = p_2 q_2$, where p_1, p_2 coincide on the t least significant bits. I.e., $p_1 = p + 2^t \tilde{p}_1$ and $p_2 = p + 2^t \tilde{p}_2$ for some common p that is *unknown* to us. Can we use the information that the prime factors of N_1 and N_2 share their t least significant bits without knowing these bits *explicitly*? I.e., can we factor N_1, N_2 given only *implicit* information about one of the factors?

In this section, we will answer this question in the affirmative. Namely, we will show that there is an algorithm that recovers the factorization of N_1 and N_2 in quadratic time provided that t is sufficiently large.

We start with

$$\begin{aligned} (p + 2^t \tilde{p}_1) q_1 &= N_1 \\ (p + 2^t \tilde{p}_2) q_2 &= N_2. \end{aligned}$$

These two equations contain five unknowns p, p_1, p_2, q_1 and q_2. By reducing both equations modulo 2^t, we can eliminate the two unknowns $\tilde{p}_1, \tilde{p}_2$ and get

$$\begin{aligned} pq_1 &\equiv N_1 \bmod 2^t \\ pq_2 &\equiv N_2 \bmod 2^t. \end{aligned}$$

Since q_1, q_2 are odd, we can solve both equations for p. This leaves us with $\frac{N_1}{q_1} \equiv \frac{N_2}{q_2} \bmod 2^t$, which we write in form of the *linear* equation

$$(N_1^{-1} N_2) q_1 - q_2 \equiv 0 \bmod 2^t. \tag{1}$$

The set of solutions

$$L = \{(x_1, x_2) \in \mathbb{Z}^2 \mid (N_1^{-1}N_2)x_1 - x_2 \equiv 0 \bmod 2^t\}$$

forms an additive, discrete subgroup of $\mathbb{Z}^2$. Thus, L is a 2-dimensional integer lattice. L is spanned by the row vectors of the basis matrix

$$B_L = \begin{pmatrix} 1 & N_1^{-1}N_2 \\ 0 & 2^t \end{pmatrix}.$$

Let us briefly check that the integer span of B_L, denoted by $\mathrm{span}(B_L)$, is indeed equal to L. First, $\mathbf{b_1} = (1, N_1^{-1}N_2)$ and $\mathbf{b_2} = (0, 2^t)$ are solutions of $(N_1^{-1}N_2)x_1 - x_2 = 0 \bmod 2^t$. Thus, every integer linear combination of $\mathbf{b_1}$ and $\mathbf{b_2}$ is a solution which implies that $\mathrm{span}(B_L) \subseteq L$.

Conversely, let $(x_1, x_2) \in L$, i.e. $(N_1^{-1}N_2)x_1 - x_2 = k \cdot 2^t$ for some $k \in \mathbb{Z}$. Then $(x_1, -k)B_L = (x_1, x_2) \in \mathrm{span}(B_L)$ and thus $L \subseteq \mathrm{span}(B_L)$.

Notice that by Eq. (1), we have $(q_1, q_2) \in L$. If we were able to find this vector in L then we could factor N_1, N_2 easily. Let us first provide some intuition under which condition the vector $\mathbf{q} = (q_1, q_2)$ is a short vector in L. We know that the length of the shortest vector is upper bounded by the Minkowski bound $\sqrt{2}\det(L)^{\frac{1}{2}} = \sqrt{2} \cdot 2^{\frac{t}{2}}$.

Since we assumed that q_1, q_2 are α-bit primes, we have $q_1, q_2 \leq 2^\alpha$. If α is sufficiently small, then $||\mathbf{q}||$ is smaller than the Minkowski bound and, therefore, we can expect that $\mathbf{q}$ is among the shortest vectors in L. This happens if

$$||\mathbf{q}|| \leq \sqrt{2} \cdot 2^\alpha \leq \sqrt{2} \cdot 2^{\frac{t}{2}}.$$

So if $t \geq 2\alpha$ we expect that $\mathbf{q}$ is a short vector in L. We can find a shortest vector in L using Gaussian reduction on the lattice basis B in time $\mathcal{O}(\log^2(2^t)) = \mathcal{O}(\log^2(\min\{N_1, N_2\}))$. Hence, under the heuristic assumption that $\mathbf{q} = (q_1, q_2)$ is a shortest vector in L we can factor N_1, N_2 in quadratic time. Under a slightly more restrictive condition, we can completely remove the heuristic assumption.

Theorem 5. *Let $N_1 = p_1q_1, N_2 = p_2q_2$ be two different RSA moduli with α-bit q_i. Suppose that p_1, p_2 share at least $t > 2(\alpha + 1)$ bits. Then N_1, N_2 can be factored in quadratic time.*

Let

$$B_L = \begin{pmatrix} 1 & N_1^{-1}N_2 \\ 0 & 2^t \end{pmatrix}$$

be the lattice basis defined as before.

B_L spans a lattice L with shortest vector $\mathbf{v}$ that satisfies

$$||\mathbf{v}|| \leq \sqrt{2}\det(L)^{\frac{1}{2}} = 2^{\frac{t+1}{2}}.$$

Performing Gaussian reduction on B_L, we get an equivalent basis $B = \begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix}$ such that

$$||\mathbf{b_1}|| = \lambda_1(L) \text{ and } ||\mathbf{b_2}|| = \lambda_2(L).$$

Our goal is to show that $\mathbf{b_1} = \pm\mathbf{q} = \pm(q_1, q_2)$ which is sufficient for factoring N_1 and N_2.

As L is of full rank, by Hadamard's inequality we have

$$||\mathbf{b_1}||\,||\mathbf{b_2}|| \geq \det(L).$$

This implies

$$||\mathbf{b_2}|| \geq \frac{\det(L)}{||\mathbf{b_1}||} = \frac{\det(L)}{\lambda_1(L)}.$$

Substituting $\det(L) = 2^t$ and using $\lambda_1(L) \leq 2^{\frac{t+1}{2}}$ leads to

$$||\mathbf{b_2}|| \geq \frac{2^t}{2^{\frac{t+1}{2}}} = 2^{\frac{t-1}{2}}.$$

This implies for any lattice vector $\mathbf{v} = a_1\mathbf{b_1} + a_2\mathbf{b_2}$ with $||\mathbf{v}|| < 2^{\frac{t-1}{2}}$ that $a_2 = 0$, as otherwise $\lambda_2(L) \leq ||\mathbf{v}|| < ||\mathbf{b_2}||$ which contradicts the optimality of $\mathbf{b_2}$ from Theorem 1. Thus, every $\mathbf{v}$ with $||\mathbf{v}|| < 2^{\frac{t-1}{2}}$ is a multiple of $\mathbf{b_1}$. Notice that $\mathbf{q} = (q_1, q_2) \in L$ fulfills $||\mathbf{q}|| = \sqrt{2} \cdot 2^{\alpha} = 2^{\frac{2\alpha+1}{2}}$. Consequently, we have $||\mathbf{q}|| < ||\mathbf{b_2}||$ if

$$2^{\frac{2\alpha+1}{2}} < 2^{\frac{t-1}{2}} \Leftrightarrow 2(\alpha + 1) < t$$

Therefore, we get $\mathbf{q} = a\mathbf{b_1}$ for some $a \in \mathbb{Z} - \{0\}$. Let $\mathbf{b_1} = (b_{11}, b_{12})$, then $\gcd(q_1, q_2) = \gcd(ab_{11}, ab_{12}) \geq a$. But q_1, q_2 are primes and wlog $q_1 \neq q_2$, since otherwise we can factor N_1, N_2 by computing $\gcd(N_1, N_2)$. Therefore, $|a| = 1$ and we obtain $\mathbf{q} = \pm\mathbf{b_1}$, which completes the factorization.

The running time of the factorization is determined by the running time of the Gaussian reduction, which can be performed in $\mathcal{O}(t^2) = \mathcal{O}(\log^2(\min\{N_1, N_2\}))$ steps. □

4 Implicit Factoring of k RSA Moduli

The approach from the previous section can be generalized to an arbitrary fixed number $k - 1$ of oracle queries. This gives us k different RSA moduli

$$\begin{aligned} N_1 &= (p + 2^t\tilde{p}_1)q_1 \\ &\vdots \\ N_k &= (p + 2^t\tilde{p}_k)q_k \end{aligned} \tag{2}$$

with α-bit q_i.

We transform the system of equations into a system of k equations modulo 2^t

$$\begin{aligned} pq_1 - N_1 &\equiv 0 \pmod{2^t} \\ &\vdots \\ pq_k - N_k &\equiv 0 \pmod{2^t} \end{aligned}$$

in $k + 1$ variables.

Analogous to the two equation case, we solve each equation for p. This can be done because all the q_i are odd. Thus, we get $\frac{N_1}{q_1} = \frac{N_i}{q_i} \pmod{2^t}$ for $i = 2, \ldots, k$. Writing this as $k-1$ linear equations gives us:

$$\begin{aligned} N_1^{-1}N_2q_1 - q_2 &\equiv 0 \pmod{2^t} \\ &\vdots \\ N_1^{-1}N_kq_1 - q_k &\equiv 0 \pmod{2^t}. \end{aligned}$$

With the same arguments as in the preceding section the set

$$L = \{(x_1, \ldots, x_k) \in \mathbb{Z}^k \mid N_1^{-1}N_ix_1 - x_i \equiv 0 \pmod{2^t} \text{ for all } i = 2, \ldots, k\}$$

forms a lattice. This lattice L is spanned by the row vectors of the following basis matrix

$$B_L = \begin{pmatrix} 1 & N_1^{-1}N_2 & \cdots & & N_1^{-1}N_k \\ 0 & 2^t & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 2^t \end{pmatrix}.$$

Note that $\mathbf{q} = (q_1, \ldots, q_k) \in L$ has norm $||\mathbf{q}|| \leq \sqrt{k}2^\alpha$. We would like to have $||\mathbf{q}|| = \lambda_1(L)$ as in Section 3. The length $\lambda_1(L)$ of a shortest vector in L is bounded by

$$\lambda_1(L) \leq \sqrt{k}(\det(L))^{\frac{1}{k}} = \sqrt{k}(2^{t(k-1)})^{\frac{1}{k}}.$$

Thus, if $\mathbf{q}$ is indeed a shortest vector then

$$||\mathbf{q}|| = \sqrt{k}2^\alpha \leq \sqrt{k} \cdot 2^{t\frac{k-1}{k}}. \tag{3}$$

This implies the condition $t \geq \frac{k}{k-1}\alpha$. We make the following heuristic assumption.

Assumption 6. *Let $N_1, \ldots, N_k$ be as defined in Eq. (2) with $t \geq \frac{k}{k-1}\alpha$. Further, let $\mathbf{b_1}$ be a shortest vector in L. Then $\mathbf{b_1} = \pm(q_1, \ldots, q_k)$.*

Theorem 7. *Let $N_1, \ldots, N_k$ be as defined in Eq. (2) with $t \geq \frac{k}{k-1}\alpha$. Under Assumption 6, we can find the factorization of all $N_1, \ldots, N_k$ in time polynomial in $(k^{\frac{k}{2}}, \max_i\{\log N_i\})$.*

We show the validity of Assumption 6 experimentally in Section 6.

The running time is determined by the time to compute a shortest vector in L [8,7]. This implies that for any lattice L of rank k such that $k^{\frac{k}{2}} = \text{poly}(\max_i\{\log N_i\})$, i.e. especially for lattices with fixed rank k, we can compute the factorization of all N_i in time polynomial in their bit-size.

For large k, our bound converges to $t \geq \alpha$. This means that the amount t of common least significant bits has to be at least as large as the bit-size of the q_i. In turn, this implies that our result only applies to RSA moduli with different bit-sizes of p_i and q_i. On the other hand, this is the best result that we could hope for in our algorithm. Notice that we construct the values of the q_i by solving equations modulo 2^t. Thus, we can fully recover the q_i only if their bitsize α is smaller than t. In the subsequent section, we will overcome this problem by avoiding the full recovery of all q_i, which in turn leads to an algorithm for balanced RSA moduli.

Remark: All of our results still hold if 2^t is replaced by an arbitrary modulus $M \geq 2^t$. We used a power of two only to illustrate our results in terms of bits.

5 Implicit Factoring of Balanced RSA Moduli

We slightly adapt the method from Section 4 in order to factor balanced n-bit integers, i. e. $N_i = p_i q_i$ such that p_i and q_i have bitsize $\frac{n}{2}$ each. The modification mainly incorporates a small brute force search on the most significant bits.

Assume that we are given k RSA moduli as in (2). From these moduli we derive $k-1$ linear equations in k variables:

$$\begin{aligned} N_1^{-1}N_2q_1 - q_2 &\equiv 0 \pmod{2^t} \\ &\vdots \\ N_1^{-1}N_kq_1 - q_k &\equiv 0 \pmod{2^t} \end{aligned}$$

The bitsize of the q_i is now fixed to $\alpha = \frac{n}{2}$ which is equal to the bitsize of the p_i, i. e. now the number t of bits on which the p_i coincide has to satisfy $t \leq \alpha$. In the trivial case of $t = \alpha = \frac{n}{2}$ we can directly factor the N_i via greatest common divisor computations as then $p_i = p$ for $i = 1, \ldots, k$.

Thus, we only consider $t < \frac{n}{2}$. With a slight modification of the method in Section 4, we compute all $q_i \pmod{2^t}$. Since $t < \frac{n}{2}$, this does not give us the q_i directly, but only their t least significant bits. But if $t \geq \frac{n}{4}$, we can use Theorem 4 for finding the full factorization of each N_i in polynomial time. In order to minimize the time complexity, we assume $t = \frac{n}{4}$ throughout this section.

To apply Theorem 7 of Section 4 the bit-size of the q_i has to be smaller than $\frac{k-1}{k}t$. Thus, we have to guess roughly $\frac{1}{k} \cdot t = \frac{n}{4k}$ bits for each q_i. Since we consider k moduli, we have to guess a total number of $\frac{n}{4}$ bits. Notice that this is the same amount of bits as for guessing one half of the bits of *one* q_j, which in turn allows to efficiently find this q_j using Theorem 4. With a total amount of $\frac{n}{4}$ bits however, our algorithm will allow us to efficiently find *all* q_i, $i = 1, \ldots, k$.

Let us describe our modification more precisely. We split $q_i \pmod{2^{\frac{n}{4}}}$ into $2^\beta \tilde{q}_i + x_i \pmod{2^{\frac{n}{4}}}$. The number β depends on the number of oracle calls $k-1$ such that the condition $\beta < \frac{(k-1)}{k} \cdot \frac{n}{4}$ holds. We therefore choose β to be the largest integer smaller than $\frac{(k-1)n}{4k}$. This implies that the $x_i \leq 2^\beta$ are small enough to be determined analogous to Section 4, provided that the $\tilde{q}_i$ are known. As discussed before, in practice we can guess an amount of $\frac{n}{4k}$ bits for determining each $\tilde{q}_i$, or we can find these bits by other means, e.g. by side-channel attacks.

Suppose now that the $\tilde{q}_i$ are given for each i. We obtain the following set of equations

$$\begin{aligned} N_1^{-1}N_2x_1 - x_2 &\equiv 2^{\beta}(\tilde{q}_2 - N_1^{-1}N_2\tilde{q}_1) \pmod{2^{\frac{n}{4}}} \\ &\vdots \\ N_1^{-1}N_kx_1 - x_k &\equiv 2^{\beta}(\tilde{q}_k - N_1^{-1}N_k\tilde{q}_1) \pmod{2^{\frac{n}{4}}}. \end{aligned} \tag{4}$$

Let $c_i = 2^{\beta}(\tilde{q}_i - N_1^{-1}N_i\tilde{q}_1)$, $i = 2, \ldots, k$, denote the known right-hand terms. In contrast to Section 4, the equations (4) that we have to solve are inhomogenous. Let us first consider the lattice L that consists of the homogenous solutions

$$L = \{(x_1, \ldots, x_k) \in \mathbb{Z}^k \mid N_1^{-1}N_ix_1 - x_i \equiv 0 \pmod{2^{\frac{n}{4}}}, i = 2, \ldots, k\}.$$

L is spanned by the rows of the following basis matrix

$$B_L = \begin{pmatrix} 1 & N_1^{-1}N_2 & \cdots & & N_1^{-1}N_k \\ 0 & 2^{\frac{n}{4}} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 2^{\frac{n}{4}} \end{pmatrix}.$$

Let $l_i \in \mathbb{Z}$ such that $N_1N_i^{-1}x_1 + l_i2^t = x_i + c_i$. Then we let

$$\mathbf{q}' := (x_1, l_2, \ldots, l_k)B_L = (x_1, x_2 + c_2, \ldots, x_k + c_k) \in L.$$

Moreover, if we define the target vector $\mathbf{c} := (0, c_2, \ldots, c_k)$, then the distance between $\mathbf{q}'$ and $\mathbf{c}$ is

$$||\mathbf{q}' - \mathbf{c}|| = ||(x_1, \ldots, x_k)|| \leq \sqrt{k}2^{\beta} \leq \sqrt{k} \cdot 2^{\frac{(k-1)n}{4k}}.$$

This is the same bound that we achieved in Section 4 for the length of a shortest vector in Eq. (3) when $t = \frac{n}{4}$. So instead of solving a shortest vector problem, we have to solve a closest vector problem in L with target vector $\mathbf{c}$. Closest vectors can be found in polynomial time for fixed lattice dimension k (see Blömer [1]). We make the heuristic assumption that $\mathbf{q}'$ is indeed a closest vector to $\mathbf{c}$ in L.

Assumption 8. *Let $N_1, \ldots, N_k$ be as defined in Eq. (4) with $\beta < \frac{(k-1)n}{4k}$. Further, let $\mathbf{b_1}$ be a closest vector to $\mathbf{c}$ in L. Then $\mathbf{b_1} = \pm\mathbf{q_1'}$.*

Theorem 9. *Let $N_1, \ldots, N_k$ be as defined in Eq. (4) with $\beta < \frac{(k-1)n}{4k}$. Under Assumption 8, we can find the factorization of all $N_1, \ldots, N_k$ in time $2^{\frac{n}{4}} \cdot \mathrm{poly}(k!, \max_i\{\log N_i\})$.*

The running time is determined by the time for guessing each $\tilde{q}_i$ and the time for finding a closest vector in L.

6 About Our Heuristic Assumptions

In this section we have a closer look at the two heuristics from the previous sections, Assumption 6 and Assumption 8. We first give a counting argument that supports our heuristics and then demonstrate experimentally that our constructions work very well in practice.

6.1 A Counting Argument That Supports Our Assumptions

Recall that in Section 4, the lattice L consists of all solutions $\mathbf{q} = (q_1, \ldots, q_k)$ of the system of equations

$$\begin{aligned} N_1^{-1}N_2q_1 &\equiv q_2 \pmod{2^t} \\ &\vdots \\ N_1^{-1}N_kq_1 &\equiv q_k \pmod{2^t} \end{aligned} \tag{5}$$

As $\gcd(N_1^{-1}N_i, 2^t) = 1$ for any i, the mapping $f_i : x \mapsto N_1^{-1}N_ix \pmod{2^t}$ is bijective. Therefore, the value of q_1 uniquely determines the values of q_i, $i = 2, \ldots, k$.

In total the system of equations has as many solutions as there are values to choose q_1 from, which is 2^t. Now suppose $q_1 \leq 2^{\frac{(k-1)t}{k}}$. How many vectors $\mathbf{q}$ do we have such that $q_i \leq 2^{\frac{(k-1)t}{k}}$ for all $i = 1, \ldots, k$ and thus $||\mathbf{q}|| \leq \sqrt{k}2^{\frac{(k-1)t}{k}}$?

Assume for each $i = 2, ..., k$ that the value q_i is uniformly distributed in $\{0, \ldots, 2^t - 1\}$ and that the distributions of q_i and q_j are independent if $i \neq j$. Then the probability that $q_i \leq 2^{\frac{(k-1)t}{k}}$ is

$$\Pr\left(q_i \leq 2^{\frac{(k-1)t}{k}}\right) = \frac{2^{\frac{(k-1)t}{k}}}{2^t} = 2^{-\frac{t}{k}}.$$

Furthermore, the probability that $q_i \leq 2^{\frac{(k-1)t}{k}}$ for all $i = 2, \ldots, k$ is

$$\Pr\left(q_2 \leq 2^{\frac{(k-1)t}{k}}, \ldots, q_k \leq 2^{\frac{(k-1)t}{k}}\right) = \left(2^{-\frac{t}{k}}\right)^{k-1} = 2^{-\frac{(k-1)t}{k}}$$

Consequently, for a given value of $q_1 \leq 2^{\frac{(k-1)t}{k}}$ the expected number of vectors $\mathbf{q}$ such that $q_i \leq 2^{\frac{(k-1)t}{k}}$ for all $i = 1, \ldots, k$ is $2^{\frac{(k-1)t}{k}} \cdot 2^{-\frac{(k-1)t}{k}} = 1$. Therefore, we expect that only one lattice vector, namely $\mathbf{q}$, is short enough to satisfy the Minkowski bound. Hence, we expect that $\pm\mathbf{q}$ is a unique shortest vector in L if its length is significantly below the bound $\sqrt{k}2^{\frac{(k-1)t}{k}}$. This counting argument strongly supports our Assumption 6.

Remark: In order to analyze Assumption 8 we can argue in a completely analogous manner. The inhomogenous character of the equations does not influence the fact that the q_i are uniquely determined by q_1.

6.2 Experiments

We verified our assumptions in practice by running experiments on a Core2 Duo 1.66GHz notebook. The attacks were implemented using Magma[1] Version 2.11. Instead of taking a lattice reduction algorithm which provably returns a basis with a shortest vector as first basis vector we have used the LLL algorithm [11], more precisely its L^2 version of Nguyen and Stehlé [15] which is implemented in Magma. Although by LLL reduction the first basis vector only approximates a shortest vector in a lattice, for our lattice bases with dimensions up to 100 LLL-reduction was sufficient. In nearly all cases the first basis vector was equal to the vector $\pm\mathbf{q} = \pm(q_1, \ldots, q_k)$, when we chose suitable attack parameters.

First, we considered the cased of imbalanced RSA moduli from Theorem 7. We chose $N_i = (p + 2^t\tilde{p}_i)q_i$, $i = 1, \ldots, k$, of bit-size $n = 1000$ with varying bitsizes of q_i. For fixed bitsize α of q_i and fixed number k of moduli, we slightly played with the parameter t of common bits close to the bound $t \geq \frac{k}{k-1}\alpha$ in order to determine the minimal t for which our heuristic is valid.

Table 1. Attack for imbalanced RSA moduli

bitsize α of the q_i	no. of moduli k	bound $\frac{k}{k-1}\alpha$	number of shared bits t	success rate
250	3	375	377	0%
250	3	375	378	97%
350	10	389	390	0%
350	10	389	391	100%
400	100	405	409	0%
400	100	405	410	100%
440	50	449	452	16%
440	50	449	453	97%
480	100	485	491	38%
480	100	485	492	98%

The running time of all experiments was below 10 seconds.

In Table 1, we called an experiment successful if the first basis vector $\mathbf{b}_1$ in our LLL reduced basis was of the form $\mathbf{b_1} = \pm\mathbf{q} = \pm(q_1, \ldots, q_k)$, i.e. it satisfied Assumption 6. There were some cases, where other basis vectors were of the form $\pm\mathbf{q}$, but we did not consider these cases.

As one can see by the experimental results, Assumption 6 only works smoothly when our instances were a few extra bits beyond the bound of Theorem 7. This is not surprising since the counting argument from Section 6.1 tells us that we loose uniqueness of the shortest vector as we approach the theoretical bound. In practice, one could either slightly increase the number t of shared bits or the number k of oracle calls for making the attack work.

[1] http://magma.maths.usyd.edu.au/magma/

Analogously, we made experiments with balanced RSA moduli to verify Assumption 8. Instead of computing closest vectors directly, we used the well-known standard embedding of a d-dimensional closest vector problem into an $(d+1)$-dimensional shortest vector problem ([6], Chapter 4.1), where the shortest vector is of the form $\mathbf{b_1} = (\mathbf{q}' - \mathbf{c}, c')$, c' constant. Since $\mathbf{c}$ and c' are known, this directly yields $\mathbf{q}'$ and therefore the factorization of all RSA moduli. For solving the shortest vector problem, we again used the LLL algorithm.

As before we called an experiment successful, if $\mathbf{b_1}$ was of the desired form, i.e. if Assumption 8 held. In our experiments we used 1000 bit N_i with a common share p of $t = 250$ bits.

Table 2. Attack for balanced 1000-bit N_i with 250 bits shared

no. of moduli k	bound $\lceil \frac{n}{4k} \rceil$	bits known from q_i	success rate
3	84	85	74%
3	84	86	99%
10	25	26	20%
10	25	27	100%
50	5	8	46%
50	5	9	100%

All of our experiments ran in less than 10 seconds. Here, we assumed that we know the required bits of each q_i, i.e. the running time does not include the factor for a brute-force search.

Similar to the experimental results for the imbalanced RSA case, our heuristic Assumption 8 works well in the balanced case, provided that we spend a few extra bits to the theoretical bound in order to enforce uniqueness of the closest vector.

References

1. Blömer, J.: Closest vectors, successive minima, and dual HKZ-bases of lattices. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 248–259. Springer, Heidelberg (2000)
2. Coppersmith, D.: Finding a small root of a bivariate integer equation, factoring with high bits known. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 178–189. Springer, Heidelberg (1996)
3. Coppersmith, D.: Small solutions to polynomial equations and low exponent vulnerabilities. Journal of Cryptology 10(4), 223–260 (1997)
4. Coppersmith, D.: Finding small solutions to small degree polynomials. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 20–31. Springer, Heidelberg (2001)
5. Crépeau, C., Slakmon, A.: Simple backdoors for RSA key generation. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 403–416. Springer, Heidelberg (2003)
6. Micciancio, D., Goldwasser, S.: Complexity of Lattice Problems: A cryptographic perspective. Kluwer International Series in Engineering and Computer Science, vol. 671. Kluwer Academic Publishers, Boston (2002)

7. Helfrich, B.: Algorithms to Construct Minkowski Reduced and Hermite Reduced Lattice Basis. Theoretical Computer Science 41, 125–139 (1985)
8. Kannan, R.: Minkowski's Convex Body Theorem and Integer Programming. Mathematics of Operations Research 12(3), 415–440 (1987)
9. Lenstra Jr., H.W.: Factoring Integers with Elliptic Curves. Ann. Math. 126, 649–673 (1987)
10. Lenstra, A.K., Lenstra Jr., H.W.: The Development of the Number Field Sieve. Springer, Heidelberg (1993)
11. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 513–534 (1982)
12. Maurer, U.M.: Factoring with an oracle. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 429–436. Springer, Heidelberg (1993)
13. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. Cambridge University Press, Cambridge (2000)
14. Minkowski, H.: Geometrie der Zahlen. Teubner-Verlag (1896)
15. Nguyen, P.Q., Stehlé, D.: Floating-point LLL revisited. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)
16. Pomerance, C.: The quadratic sieve factoring algorithm. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 169–182. Springer, Heidelberg (1985)
17. Rivest, R.L., Shamir, A.: Efficient factoring based on partial information. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 31–34. Springer, Heidelberg (1986)
18. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
19. Shor, P.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, pp. 124–134. IEEE Computer Science Press, Los Alamitos (1994)
20. Steinfeld, R., Zheng, Y.: An advantage of low-exponent RSA with modulus primes sharing least significant bits. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 52–62. Springer, Heidelberg (2001)
21. Vanstone, S.A., Zuccherato, R.J.: Short RSA Keys and Their Generation. Journal of Cryptology 8(2), 101–114 (1995)
22. Young, A., Yung, M.: The prevalence of kleptographic attacks on discrete-log based cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 264–276. Springer, Heidelberg (1997)

The Security of All Bits Using List Decoding

Paz Morillo and Carla Ràfols

Dept. Matemàtica Aplicada IV
Universitat Politècnica de Catalunya
C. Jordi Girona 1-3, E-08034 Barcelona, Spain
{paz,crafols}@ma4.upc.edu

Abstract. The relation between list decoding and hard-core predicates has provided a clean and easy methodology to prove the hardness of certain predicates. So far this methodology has only been used to prove that the $O(\log \log N)$ least and most significant bits of any function with multiplicative access —which include the most common number theoretic trapdoor permutations— are secure. In this paper we show that the method applies to all bits of any function defined on a cyclic group of order N with multiplicative access for cryptographically interesting N. As a result, in this paper we reprove the security of all bits of RSA, the discrete logarithm in a group of prime order or the Paillier encryption scheme.

Keywords: bit security, list decoding, one-way function.

1 Introduction

One-way functions are one of the most fundamental cryptographic primitives and it is not an overstatement to say that they are behind most of modern cryptography. If some reasonable computational assumptions hold, a one-way function is easy to compute but hard to invert. In some cases, this security requirement may not be enough: in particular, the definition of one-way function does not say anything about how much information it can leak. A predicate of the preimage, P, is a *hard-core* of f if f does not give away any information about P, that is, if there exists a polynomial time reduction from guessing P to inverting f.

The study of hard-core predicates is of interest for various reasons, not only because it strengthens our understanding of the real hardness of the considered one-way function, but also because of its applications, which include the construction of secure bit commitment schemes or cryptographically strong pseudorandom generators. Further, the study of bit security has led to important techniques and insights which have found other applications. For instance, the study of the security of the least significant bit of RSA led to the two-point based sampling technique introduced in [2], later used to prove the well known result of the Goldreich and Levin bit — GL from now on — which states that every one-way function has a hard-core bit. We emphasize that the importance

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 15–33, 2009.

of the GL result reaches far beyond the domain of bit security, and many works in other lines of research are in some way indebted to it, for instance in learning theory [4],[7].

Many bit security results have very technical and sophisticated proofs. Although many proofs for different one-way functions have a similar structure, they have to be adapted to each particular case. In contrast, Akavia, Goldwasser and Safra [1] give a very elegant and general methodology to prove bit security results. In particular, they show how this methodology applies to prove the security of $O(\log\log N)$ least and most significant bits of any function with *multiplicative access* - such as RSA and DL, for instance.

Akavia *et al.* raised the question whether this methodology applies to prove the security of internal bits, a question which we answer in the affirmative in this paper. Since the existing security proofs for the hardness of internal bits of RSA and DL are particularly technical and cumbersome to follow in all detail — we refer the reader to [8] for an example— , we feel that a more readable proof should contribute much to the public discussion of the results and thus also to their credit and diffusion.

1.1 Previous Work

The GL result, which gives a hard-core predicate for any one-way function, can be reinterpreted as a list decoding algorithm for the Hadamard code. This suggested the possibility of a general methodology to prove bit security results. This methodology was formalized by Akavia *et al.* in 2003, where it was used to prove (or often *reprove* known results) the hardness of certain predicates for one way functions defined on a cyclic group $\mathbb{G}$ of order N and having the property of multiplicative access, that is, functions f for which given $f(x)$, $f(x \cdot y)$ can be computed for any known y in almost all of the cases.

The most common number theoretic trapdoor permutations in cryptography, such as $RSA_{N,e}(x) = x^e \mod N$, $Rabin(x) = x^2 \mod N$, $EXP_{p,g}(x) = g^x \mod p$ and $ECL_{p,a,b,Q}(x) = xQ$ — exponentiation in the group of $\mathbb{F}_p$- rational points of an elliptic curve $E_{p,a,b}(\mathbb{F}_p)$ — have this property.

A part from the formalization of the list decoding methodology, one of the key contributions of Akavia *et al.* is to give a learning algorithm for functions $f : \mathbb{Z}_N \to \mathbb{C}$, which is a necessary piece to provide the aforementioned results.

The security of the internal bits had already been proved for some one-way functions with multiplicative access such as RSA and the discrete logarithm in [8] and the Paillier encryption scheme in [3].

1.2 Organization

Sections 2, 3 and 4 are introductory: in section 2 we define the concept of hard-core predicate and give some of the results of Akavia *et al.* Sections 3 and 4 are devoted to basics of Fourier analysis and list decodable codes and to the relation between hard-core predicates and list decoding. We stress that many of the results and definitions given in these sections are taken from the work of

Akavia *et al.* but are necessary to introduce our contributions. In section 5 we prove one of our main results concerning the security of all bits of any one-way function with multiplicative access for special N, while in section 6 we prove it for all N of cryptographic interest. Next, in section 7 we prove the security of all bits of the Paillier encryption scheme. Possible extensions of these results are discussed in section 8. In section 9 we summarize our contribution.

2 Parts That Are as Hard as the Whole

Informally, a hard-core bit for a one-way function $f : \mathcal{D} \rightarrow R$ is a boolean predicate $P : \mathcal{D} \rightarrow \{\pm 1\}$ which does not leak from f. Obviously, we cannot prevent an adversary from taking a random guess, but the point is that there should not be any strategy to predict P which works *significantly better* than the random one. Define

Definition 1. $maj_P \overset{def}{=} max_{b\in\{\pm1\}} Pr\big(P(x) = b : x \leftarrow D\big)$ *and* $minor_P \overset{def}{=} 1 - maj_P$.

We write $x \leftarrow D$ to indicate that we choose an element x in D according to the uniform distribution.

Definition 2. *A function* $\nu(\cdot)$ *is negligible if for any constant* $c \geq 0$ *there exists* $n_0 \in \mathbb{Z}$*, s.t* $\nu(n) < n^{-c}$ *for all integers* $n \geq n_0$*.*

This definition of hard-core predicate is taken from [1].

Definition 3. *For each* $n \in \mathbb{N}$*, let* I_n *be a countable index set, and set* $I = (I_n)_{n\in\mathbb{N}}$ *. Let* $F = (f_i : D_i \rightarrow R_i)_{i\in I}$ *be a family of one-way functions and* $\mathcal{P} = (P_i : D_i \rightarrow \{\pm1\})_{i\in I}$ *a family of Boolean predicates, where w.l.o.g. if* $i \in I_n$ $D_i \subset \{0,1\}^n$*. We will say that* $\mathcal{P}$ *is a family of* hard-core *predicates for* F *if and only if, for all* $n \in \mathbb{N}$ *and* $i \in I_n$*:*

- P_i *can be computed by means of a Monte-Carlo algorithm* $\mathcal{A}_1(i, x)$*.*
- $P_i(x)$ *is not computable by means of any efficient algorithm from* $f_i(x)$*; that is, for any PPT algorithm* $\mathcal{A}_2$*,*

$$Pr(\mathcal{A}_2(i, f_i(x)) = P_i(x) : x \leftarrow D_i) \leq maj_{P_i} + \nu(n),$$

where $\nu(\cdot)$ *is a negligible function.*

While there are predicates which are a hard-core of any one-way function, like the GL bit [6] or all the bits of $ax+b \mod p$ [9], there are also many results concerning the security of a certain bit of the binary representation of the preimage for a specific one-way function (e.g. the least significant bit of RSA or Rabin, see for instance [2]).

Given an element $x \in \mathbb{Z}_N$, define $[x]$ as the representative of the class of x in $[0, N)$ and $abs_N(x) = min\{[x], N - [x]\}$. The i-th bit of an element $x \in \mathbb{Z}_N$ is defined as $B_i(x) = 1$ if the i-th bit of the binary representation of $[x]$ is 0 and as -1 otherwise.

Akavia *et al.* prove the security of any basic t-segment predicate, $t \in poly(n)$, for any one-way function with multiplicative access having domain in $\mathbb{Z}_N$, where $n \overset{\mathbf{def}}{=} \lfloor \log N \rfloor$.

Definition 4. *A predicate $P_N : \mathbb{Z}_N \rightarrow \{\pm 1\}$ is said to be a basic t-segment predicate if there are at most t values of $x \in \mathbb{Z}_N$ for which $P_N(x+1) \neq P_N(x)$.*

In particular, their result implies that the predicate B_{n-i}, where $i \in poly(\log n)$, is a hard-core of any one-way function with multiplicative access, since trivially B_{n-i} is a basic t-segment predicate, where $t = 2^{i+1}$.

Further, there is a correspondence between B_i, where $i \in O(\log n)$ and some t-basic segment predicate with $t \in poly(n)$. For instance, it is easy to verify that $lsb_N(x) = half_N(\frac{x}{2})$, where $half_N(x)$ is a basic 2-segment predicate which is equal to 1 if $[x] \leq N/2$ and is -1 otherwise. This correspondence allows to prove that the predicates B_i, where $i \in O(\log n)$ are also hard-core of any one-way function with multiplicative access when N is odd (see [1] for details).

3 Preliminaries

Before sketching the list decoding methodology of [1], we begin with some basic concepts.

3.1 Fourier Analysis in $\mathbb{Z}_N$

In the space of functions from $\mathbb{Z}_N$ to $\mathbb{C}$ it is possible to define the inner product

$$\langle g, h \rangle \overset{\mathbf{def}}{=} \frac{1}{N} \sum_{x \in \mathbb{Z}_N} g(x)\overline{h(x)}.$$

For each $\alpha \in \mathbb{Z}_N$, the α-character is defined as a function $\chi_\alpha : \mathbb{Z}_N \rightarrow \mathbb{C}$ such that $\chi_\alpha(x) = w_N^{\alpha x}$, where $w_N \overset{\mathbf{def}}{=} e^{\frac{2\pi j}{N}}$. It is easy to check that $B_\alpha \overset{\mathbf{def}}{=} \{\chi_\alpha : \alpha \in \mathbb{Z}_N\}$ is an orthonormal basis of the space of functions going from $\mathbb{Z}_N$ to $\mathbb{C}$.

If Γ is a subset of $\mathbb{Z}_N$, it is natural to consider the projection of g in the set of Γ characters, that is,

$$g_{|\Gamma} = \sum_{\alpha \in \Gamma} \widehat{g(\alpha)} \chi_\alpha,$$

where $\widehat{g(\alpha)} = \langle g, \chi_\alpha \rangle$ are the *Fourier coefficients.* Observe that, if $h(y) = g(ay)$ for some $a \in \mathbb{Z}_N^*$, then $\widehat{h(\alpha)} = \widehat{g(\alpha/a)}$.

Because B_α is an orthonormal basis,

$$||g||_2^2 = \sum_{\alpha \in \mathbb{Z}_N} |\widehat{g(\alpha)}|^2 \text{ and } ||g_{|\Gamma}||_2^2 = \sum_{\alpha \in \Gamma} |\widehat{g(\alpha)}|^2.$$

Finally, define

Definition 5. *(Fourier Concentrated) A function $g : \mathbb{Z}_N \to \mathbb{C}$ is Fourier concentrated if for every $\epsilon > 0$ there exists a set $\mathit{\Gamma}$ consisting of poly(n/ϵ) characters, so that*

$$||g - g_{|\mathit{\Gamma}}||_2^2 = \sum_{\alpha \notin \mathit{\Gamma}} |\widehat{g(\alpha)}|^2 \leq \epsilon.$$

In the following, this condition will be referred to as g is ϵ-concentrated on the set $\mathit{\Gamma}$.

The *heavy characters* of g are the characters for which the projection of g has a greater modulus, that is, given $\tau > 0$ and $g : \mathbb{Z}_N \to \mathbb{C}$, define

$$Heavy_\tau(g) \overset{\textbf{def}}{=} \{\chi_\alpha : |\widehat{g(\alpha)}|^2 \ \geq \tau\}.$$

3.2 Codes

A binary code is a subset of $\{\pm 1\}^*$. To encode the elements of $\mathbb{Z}_N$ we will limit ourselves to codewords of length N, in this case the code is a subset $\mathcal{C} \subset \{\pm 1\}^N$. Each codeword C_x can be seen as a function $C_x : \mathbb{Z}_N \to \{\pm 1\}$, expressed as $(C_x(0), C_x(1), \ldots, C_x(N-1))$.

Definition 6. *The normalized Hamming distance between two functions $g, h : \mathbb{Z}_N \to \{\pm 1\}$ is $\mathit{\Delta}(g, h) = Pr\big(g(x) \neq h(x) : x \leftarrow \mathbb{Z}_N\big)$.*

The next definition is a natural extension of the concept of error correcting codes.

Definition 7. *A code $\mathcal{C} = \{C_x : \mathbb{Z}_N \to \{\pm 1\}\}$ is list decodable if there exists a PPT algorithm which given access to a corrupted codeword w and inputs $\delta, \epsilon, 1^n$ returns a list $L \supseteq \{x : \mathit{\Delta}(C_x, w) \leq minor_{C_x} - \epsilon\}$ with probability $1 - \delta$.*

Remark 1. In this definition it says "given access to w" because in our examples it will be computationally infeasible to read the whole word w due to its size.

3.3 List Decodable Codes

In this section we give sufficient conditions for a code to be list decodable, for a detailed explanation we refer the reader to [1].

Definition 8. *A code $\mathcal{C}$ is concentrated if each of its codewords C_x is Fourier concentrated.*

Definition 9. *A code $\mathcal{C}$ is recoverable, if there exists a recovery algorithm, namely, a polynomial time algorithm that, given a character χ_α (for $\alpha \neq 0$), a threshold τ and 1^n, where $n = \lfloor \log N \rfloor$ returns a list L_α containing*

$$\{x \in \mathbb{Z}_N : \chi_\alpha \in Heavy_\tau(C_x)\}.$$

One of the main contributions of Akavia *et al.* is to prove that on input a threshold τ and given access to any function $g : \mathbb{G} \to \mathbb{C}$ where $\mathbb{G}$ is any abelian group with known generators of known orders, it is computationally feasible to obtain a list of all the Fourier coefficients in $Heavy_\tau(g)$. In particular, in the $\mathbb{Z}_N$ case — which is enough for our purposes — they prove that

Theorem 1. *There is an algorithm which, given query access to* $g : \mathbb{Z}_N \to \{\pm 1\}$*,* $0 < \tau$ *and* $0 < \delta < 1$*, outputs a list* L*, of* $O(1/\tau)$ *characters s.t.* $Heavy_\tau(g) \subset L$ *with probability at least* $1 - \delta$*; and the running time of the algorithm is* $\widetilde{O}(n \cdot \ln^2(1/\delta)/\tau^{5.5})$*, where the* $\widetilde{O}()$ *notation indicates that terms with complexity polynomial in* $\log(1/\tau), \log n$ *or* $\ln \ln(1/\delta)$ *have been eliminated.*

Another algorithm to the same purpose was given by Strauss and Mutukrishnan [5], resulting in a running time with improved dependence in $1/\tau$.

This theorem is used in [1] to prove the following

Theorem 2. *Let* $\mathcal{C} = \{C_x : \mathbb{Z}_N \to \{\pm 1\}\}$ *be a concentrated and recoverable code, then* $\mathcal{C}$ *is list decodable.*

The intuition behind the theorem is the following. Suppose that we have access to a corrupted word w which is close enough to a codeword C_x, then:

- Because of the concentration of the code and the closeness of w and C_x, there exists an explicit threshold τ — non-negligible in n— such that χ_β is a τ-heavy coefficient of both w and C_x, that is, there exists a $\beta \in \mathbb{Z}_N$, $\beta \neq 0$, such that
$$\chi_\beta \in Heavy_\tau(w) \cap Heavy_\tau(C_x).$$
This is proven in the *Concentration and agreement lemma* of [1].
- Because of theorem 1, on input this threshold τ, we can recover a list L with all the Fourier coefficients in $Heavy_\tau(w)$ with probability $1-\delta$. We emphasize that if δ is non-negligible in n, both the running time of the algorithm and the length of the list — which is $1/\tau$— is polynomial in n.
- For each of these coefficients χ_β the recovery algorithm will output a list of codewords and C_x will be in at least one of those lists.

4 The Relation between List Decoding and Hard-Core Predicates

In this section we summarize the connection between list decoding and hard-core predicates from [1].

Suppose we want to prove that $P : \mathbb{Z}_N \to \{\pm 1\}$ is a hard-core of $f : \mathbb{Z}_N \to R$. As it is standard in cryptography, the security of P is proved by a *reduction argument*. The proof consists in trying to invert f (recover x) given a challenge $f(x)$ and assuming we have access to an oracle predicting $P(y)$ from $f(y)$ with non-negligible advantage over a random guess.

When f has multiplicative access, the connection between list decoding and hard-core predicates comes from encoding each element $x \in \mathbb{Z}_N$ as $C_x^P = (C_x^P(0), C_x^P(1), \ldots, C_x^P(N-1))$, where $C_x^P(j) = P(jx)$. This is the so-called *multiplication code*. An oracle predicting $P(y)$ from $f(y)$ without errors would give us access to C_x^P, but since the oracle gives incorrect answers we have access to a corrupted codeword w instead. If the code is list decodable we can find a list of codewords containing C_x^P, thus inverting f.

Now, in general, for any function f — not necessarily with multiplicative access— to prove that P is a hard-core predicate of f following the list decoding methodology, it would suffice to somehow encode the elements of $\mathbb{Z}_N$ in such a way that,

- The code is concentrated and recoverable (that is, list decodable).
- Given the challenge $f(x)$ and an oracle predicting P we can devise access to a corrupted codeword w close enough to the encoding of x.

This is formalized in Theorem 2 (List Decoding Approach) of [1],

Theorem 3. *Assume a collection of codes $\mathcal{C}^P = \{\mathcal{C}^{P_i}\}_{i \in I}$ s.t. $\forall i \in I$, (1) $\mathcal{C}^{P_i}$ is list decodable, and (2) $\mathcal{C}^{P_i}$ accessible with respect to f_i. Then $\mathcal{P}$ is hard-core of F.*

The definition of *accessible code* is:

Definition 10. *Let $\mathcal{P}$ be a collection of predicates and F a family of one-way functions. The code $\mathcal{C}$ is accessible with respect to F if there exists a PPT access algorithm $\mathcal{A}$, such that for all $i \in I_n$, $\mathcal{C}^{P_i}$ is accessible with respect to f_i, namely*

1. ***Code access:*** *$\forall x, j \in \mathcal{D}_i$, $\mathcal{A}(i, f_i(x), j)$ returns $f_i(x')$ such that $C_x^{P_i}(j) = P_i(x')$*
2. ***Well spread:*** *For uniformly distributed $C_x^{P_i} \in \mathcal{C}^{P_i}$ and $j \in D_i$, the distribution of x' satisfying $f_i(x') = \mathcal{A}(i, f_i(x), j)$ is statistically close to the uniform distribution on D_i*
3. ***Bias preserving:*** *For every codeword $C_x^{P_i} \in \mathcal{C}^{P_i}$,*

$$|Pr\big(C_x^{P_i}(j) = 1 : j \leftarrow \mathcal{D}_i\big) - Pr\big(P_i(z) = 1 : z \leftarrow \mathcal{D}_i\big)| \leq \nu(n),$$

where ν is a negligible function.

Lemma 3 of [1] proves that if $\mathcal{C}^P$ is accessible with respect to F and an algorithm $\mathcal{B}$ that predicts P from F with probability at least $maj_P + \epsilon$ is given, then, for a non-negligible fraction of the codewords $C_x^P \in \mathcal{C}^P$, given $f(x)$ we have access to a corrupted codeword w_x close enough to C_x^P.

Akavia *et al.* prove that the multiplication code $\mathcal{C}^P$ is accessible with respect to RSA and $EXP_{p,g}$ and they state that it also holds for *Rabin* and *ECL*. In section 7 we prove that $\mathcal{C}^P$ is accessible with respect to the Paillier one-way function.

Once the accessibility of the code with respect to a one-way function f is stablished, to prove that P is a hard-core of f it suffices to see that the multiplication code C_x^P is concentrated and recoverable. Concerning the concentration, observe that if $x \in \mathbb{Z}_N^*$, from the definition of multiplication code, there is a simple relation between the Fourier coefficients of C_x^P and P, $\widehat{C_x^P(\beta)} = \widehat{P\big(\beta/x\big)}$. As a consequence,

Lemma 1. *For all $\epsilon > 0$, if P is ϵ-concentrated in Γ then C_x^P is ϵ-concentrated in $\Gamma' = \{\chi_\beta : \beta = \alpha x \mod N, \chi_\alpha \in \Gamma\}$.*

5 The Security of All Bits for Special N

The purpose of this section is to prove that the predicate $P(x) = B_i(x)$, defined in section 2, is a hard-core predicate of any one-way function defined over $\mathbb{Z}_N$ for which the multiplication code is accessible, for N of special form. Because of theorem 3 it suffices to prove that the multiplication code $\mathcal{C}^{B_i}$ is concentrated and recoverable.

The organization of this section is the following: to prove that P is concentrated, we begin giving an explicit formula for the Fourier coefficients of the ith bit in subsection 5.1. This formula is used in subsection 5.2 to study the asymptotic behavior of $|\widehat{P(\alpha)}|^2$.

In subsection 5.3 we prove that P is concentrated for all N of a special form. Theorem 6 of subsection 5.4 proves one of the main results of the paper namely that the predicate ith bit is hard-core of any one-way function defined over $\mathbb{Z}_N$ for which the multiplication code is accessible, for N of special form. To do this we prove the recoverability of the code $\mathcal{C}^{B_i}$ in theorem 5. It turns out that these partial results are enough to reprove the hardness of $O(\log n)$ most and least significant bits.

5.1 The Fourier Representation of the ith Bit

Let $P(x) = B_i(x)$ be the ith bit as defined in section 2 and $N = r2^{i+1} \pm m$, where $0 < m < 2^i$. Define the function $g(x) = \dfrac{P(x+2^i) + P(x)}{2}$.

Recall that $w_N = e^{\frac{2\pi j}{N}}$. From the definitions given in section 3.1 and using some properties of the Fourier coefficients, we have the following relation

$$\widehat{g(\alpha)} = \frac{(w_N^{2^i\alpha} + 1)}{2}\widehat{P(\alpha)}.$$

We consider two different situations:

- **Case 1** $N = r2^{i+1} - m$. In this case $g(x) = 1$ if and only if $x \in I_1 \overset{\mathbf{def}}{=} [(r-1)2^{i+1} + 2^i - m, (r-1)2^{i+1} + 2^i - 1]$, else $g(x) = 0$.
- **Case 2** $N = r2^{i+1} + m$. In this case $g(x) = 1$ if and only if $x \in I_2 \overset{\mathbf{def}}{=} [r2^{i+1}, r2^{i+1} + m - 1]$, else $g(x) = 0$.

In either of the two cases it is easy to compute the Fourier coefficients of P explicitly. Indeed, it suffices to find $\widehat{g(\alpha)}$ since $w_N^{2^i\alpha} + 1 \neq 0$ because $m \neq 0$. Note that in both Case 1 and Case 2, $g(x)$ is only different from 0 in an interval of length m. As a result the non-zero summands in the expression of $\widehat{g(\alpha)}$ form a geometric progression with exactly m terms and $\widehat{g(\alpha)}$ can be computed explicitly. If $\alpha \neq 0$, in Case 1:

$$\widehat{g(\alpha)} = \frac{1}{N}\sum_{y\in I_1}\overline{\chi_\alpha(y)} = \frac{1}{N}\, w_N^{-\alpha((r-1)2^{i+1}+2^i-m)}\, \frac{(w_N^{-\alpha m} - 1)}{(w_N^{-\alpha} - 1)}.$$

Analogously, in Case 2, $\widehat{g(\alpha)} = \dfrac{1}{N}\ w_N^{-\alpha(r2^{i+1})}\ \dfrac{(w_N^{-\alpha m}-1)}{(w_N^{-\alpha}-1)}$. Moreover, in both cases, $\widehat{g(0)} = \dfrac{m}{N}$.

Taking the modulus, we obtain that in both cases, for any $\alpha \neq 0$

$$|\widehat{g(\alpha)}|^2 = \frac{1}{N^2}\frac{\sin^2(\frac{m\alpha\pi}{N})}{\sin^2(\frac{\alpha\pi}{N})}.$$

Using the fact that $|w_N^{2^i\alpha}+1|^2 = 4\cos^2(\frac{2^i\alpha\pi}{N})$, we obtain

$$|\widehat{P(\alpha)}|^2 = |\widehat{g(\alpha)}|^2 \frac{1}{\cos^2(\frac{2^i\alpha\pi}{N})} = \frac{1}{N^2}\frac{\sin^2(\frac{m\alpha\pi}{N})}{\sin^2(\frac{\alpha\pi}{N})\cos^2(\frac{2^i\alpha\pi}{N})}. \tag{1}$$

Remark. There is an alternative trick to compute $|\widehat{P(\alpha)}|^2$, it suffices to consider the function $G(x) = \dfrac{P(x+1)-P(x)}{2}$. In this case,

$$|\widehat{P(\alpha)}|^2 = |\widehat{G(\alpha)}|^2 \frac{1}{\sin^2(\frac{\alpha\pi}{N})}. \tag{2}$$

Define Case 1 and Case 2 as above. Note that in either one of the cases the function takes values in $\{\pm 1\}$ whenever $x = k2^i - 1$, for $k \in \mathbb{Z}$. Additionally, in Case 1, $G(N-1) = 1$. As a consequence, in both cases the function takes exactly $2r$ non-zero values. This remark will be useful in subsections 5.2 and 5.3.

5.2 Asymptotic Behaviour of the Fourier Coefficients of the ith Bit

We have just seen how to compute the coefficients $\widehat{P(\alpha)}$. In this section we use basic calculus techniques to study its asymptotic behavior.

Proposition 1

$$|\widehat{P(\alpha)}|^2 = \Theta\left(\frac{abs_N(m\alpha)^2}{abs_N(\alpha)^2 abs_N(2^i\alpha - N/2)^2}\right)$$

The proof essentially follows from the following lemma.

Lemma 2. $\pi^2(1-\frac{\pi^2}{12})(abs_N(y))^2 \leq N^2\sin^2(\frac{y\pi}{N}) \leq \pi^2(abs_N(y))^2$

Proof. We use the fact that $x^2 - \frac{x^4}{3} \leq \sin^2 x \leq x^2$, for any $x \in [-\pi, \pi]$, then:

- Left inequality: Let j be the only integer such that $|y - jN| \leq \frac{N}{2}$, then

$$\begin{array}{l} N^2\sin^2(\frac{y\pi}{N}) = N^2\sin^2(\frac{y\pi}{N} - j\pi) \geq N^2(\frac{y\pi}{N} - j\pi)^2(1 - \frac{1}{3}(\frac{y\pi}{N} - j\pi)^2) \geq \\ \geq \pi^2(1-\frac{\pi^2}{12})(abs_N(y))^2 \end{array}.$$

- Right inequality: Similarly, $N^2\sin^2(\dfrac{y\pi}{N}) \leq N^2(\dfrac{y\pi}{N} - j\pi)^2 = \pi^2(abs_N(y))^2$.□

To prove proposition 1, we first define j as the unique odd integer in $[-2^i, 2^i]$ such that $|2^i\alpha - j\frac{N}{2}| \leq \frac{N}{2}$. Since $\cos^2(\frac{2^i\alpha\pi}{N}) = \sin^2(\frac{2^i\alpha\pi}{N} - j\frac{\pi}{2})$, proposition 1 is derived from expression (1) of $|\widehat{P(\alpha)}|^2$ and lemma 2 for $y = \alpha$, $y = m\alpha$ and $y = 2^i\alpha - N/2$.

Summarizing, we have proven proposition 1 and the constants implied in the symbol $\Theta()$ can be found explicitly, indeed,

$$K_1 \cdot \frac{abs_N(m\alpha)^2}{abs_N(\alpha)^2 abs_N(2^i\alpha - \frac{N}{2})^2} \leq |\widehat{P(\alpha)}|^2 \leq K_2 \cdot \frac{abs_N(m\alpha)^2}{abs_N(\alpha)^2 abs_N(2^i\alpha - \frac{N}{2})^2},$$

where $K_1 \overset{\mathbf{def}}{=} (\frac{1}{\pi^2} - \frac{1}{12})$ and $K_2 \overset{\mathbf{def}}{=} \dfrac{1}{\pi^2(1-\frac{\pi^2}{12})^2}$.

Finally, if $K_3 \overset{\mathbf{def}}{=} \dfrac{1}{\pi^2(1-\frac{\pi^2}{12})}$, we prove

Lemma 3. $|\widehat{P(\alpha)}|^2 \leq K_3 \cdot \; min\left\{\dfrac{m^2}{abs_N(2^i\alpha - \frac{N}{2})^2}, \dfrac{4r^2}{abs_N(\alpha)^2}\right\}$.

Proof. The function g is equal to 0 in all but m elements of the domain and therefore $|\widehat{g(\alpha)}|^2 \leq \dfrac{m^2}{N^2}$, since each coefficient is the sum of m terms in the unit circle. The inequalities of lemma 2 and expression (1) imply that $|\widehat{P(\alpha)}|^2 \leq K_3 \cdot \dfrac{m^2}{abs_N(2^i\alpha - \frac{N}{2})^2}$. To prove the other bound observe that $|\widehat{G(\alpha)}|^2 \leq \dfrac{4r^2}{N^2}$ and use expression (2) and lemma 2.

□

5.3 The Concentration of the ith Bit for Certain N

In the previous section we found an expression of the asymptotic behavior of the coefficients of P which is hard to interpret. It is clear that the heavy coefficients of P will be around the points that annihilate the denominator, but otherwise it is not trivial to show that there exists a set Γ of size $poly(n/\epsilon)$ such that $||P - P_{|\Gamma}||_2^2 \leq \epsilon$.

In this section we prove that if $N = r2^{i+1} \pm m$ and either $r \in poly(n)$ or $m \in poly(n)$ then P is concentrated. The result is a consequence of the two following lemmas.

Lemma 4. *For any* $\epsilon > 0$, $||P - P_{|\Gamma_{2^i}}||_2^2 \leq \epsilon$, *where*

$$\Gamma_{2^i} \overset{\textit{def}}{=} \{\chi_\alpha : abs_N(2^i\alpha - N/2) \leq O(\frac{m^2}{\epsilon})\}.$$

Proof. Let $\Gamma_k^c \overset{\mathbf{def}}{=} \{\chi_\alpha : abs_N(2^i\alpha - N/2) > k\}$ then, using one of the bounds of lemma 3

$$\sum_{\chi_\alpha \in \Gamma_k^c} |\widehat{P(\alpha)}|^2 \leq O(m^2) \sum_{\chi_\alpha \in \Gamma_k^c} \frac{1}{abs_N(2^i\alpha - N/2)^2} < O(\frac{m^2}{k}).$$

Taking $k \in O(\frac{m^2}{\epsilon})$, $||P - P_{|\Gamma_{2^i}}||_2^2 \leq \epsilon$. □

Similarly, using the other bound of lemma 3.

Lemma 5. *For any $\epsilon > 0$, $||P - P_{|\Gamma_0}||_2^2 \leq \epsilon$, where*

$$\Gamma_0 \overset{def}{=} \{\chi_\alpha : abs_N(\alpha) \leq O(\frac{r^2}{\epsilon})\}.$$

That is, P is concentrated in $\Gamma \overset{\textbf{def}}{=} \Gamma_0 \cap \Gamma_{2^i}$ and in case either $r \in poly(n)$ or $m \in poly(n)$, the cardinal of this set is $poly(n/\epsilon)$. Because of lemma 1, we have proven the following theorem.

Theorem 4. *The code $\mathcal{C}^{B_i}$ is concentrated for all $N = r2^{i+1} \pm m$, $0 < m < 2^i$, with either $r \in poly(n)$ or $m \in poly(n)$.*

5.4 The Hardness of the ith Bit for Certain N

We study the recoverability of $\mathcal{C}^P$. The recovery algorithm is adapted from lemma 5 of [1] which proved that if B is a t-segment predicate, $\mathcal{C}^B$ is recoverable. Combined with the concentration proven in theorem 4, the recoverability of $\mathcal{C}^P$ will prove the main result about the hardness of the ith bit for certain N.

Theorem 5. *The code $\mathcal{C}^{B_i}$ is recoverable for all $N = r2^{i+1} \pm m$ with either $r \in poly(n)$ or $m \in poly(n)$ for N a prime or an RSA modulus.*

Proof. We first consider the case $r \in poly(n)$. In this case, because of lemma 1 and lemma 5, C_x^P is τ-concentrated in $\Gamma_0' \overset{\textbf{def}}{=} \{\chi_\beta : \beta = \alpha x \mod N, abs_N(\alpha) \leq O(\frac{r^2}{\tau})\}$.

The inputs of the recovery algorithm are a character χ_β and a threshold parameter τ (where $1/\tau \in poly(n)$). The output is a list containing $x \in \mathbb{Z}_N$ such that $\chi_\beta \in Heavy_\tau(C_x^P)$.

Since, C_x^P is τ-concentrated in Γ_0', $\chi_\beta \in Heavy_\tau(C_x^P)$ implies $\chi_\beta \in \Gamma_0'$ and thus $\beta = \alpha x \mod N$ for $abs_N(\alpha) \leq poly(n/\tau)$. The algorithm outputs the union of the lists L_α such that L_α contains all x so that $x = \beta/\alpha \mod N$. If $\alpha \in \mathbb{Z}_N^*$ there is a single solution to this equation. If $gcd(N, \alpha) = d \neq 1$, the solution of $\frac{\alpha}{d}x = \beta \mod \frac{N}{d}$ is either empty or a list

$$L_\alpha = \{x + i \cdot \frac{N}{d} \mod N\}_{i=0,\ldots,d-1}.$$

The union of the lists L_α (over all α such that $abs(\alpha) \leq O(\frac{r^2}{\tau})$) contains all x such that $Heavy_\tau(C_x^P) \ni \chi_\beta$. For the length of the lists and the time of constructing them be $poly(n/\tau)$, it must be that $d \in poly(n)$, since L_α has length d. This condition is trivially satisfied if N is a prime. If N is an RSA modulus, $gcd(N, \alpha) \neq 1$ implies factoring N. So, for an RSA modulus $N = r2^{i+1} \pm m$ with $r \in poly(n)$, either the code $\mathcal{C}^{B_i}$ is recoverable or P is concentrated in a known set Γ_0 of polynomial size which contains some element which allows to factorize N, contradicting the unfeasibility of factoring RSA modulus.

The case $m \in poly(n)$ is proven in a similar way but now taking into account C_x^P is τ-concentrated in $\Gamma'_{2^i} \overset{\mathbf{def}}{=} \{\chi_\beta : \beta = \alpha x \mod N, abs_N(2^i\alpha - N/2) \leq O(\frac{m^2}{\tau})\}$. □

Theorem 2 states that, as a consequence of theorems 4 and 5, the code $\mathcal{C}^{B_i}$ is list decodable for all $N = r2^{i+1} \pm m$ with either $r \in poly(n)$ or $m \in poly(n)$ and N a prime or an RSA modulus.

Finally, we conclude

Theorem 6. *The predicate B_i, ith bit, is a hard-core predicate for any one-way function defined over $\mathbb{Z}_N$ for which the multiplication code $\mathcal{C}^{B_i}$ is accessible, for all $N = r2^{i+1} \pm m$ prime or RSA modulus such that either $r \in poly(n)$ or $m \in poly(n)$.*

This theorem is a consequence of the list decodability of the code $\mathcal{C}^{B_i}$ and theorem 3.

This proves the hardness of all bits for N with a special binary representation, but it also reproves the hardness of the $O(\log n)$ most and least significant bits for all N of cryptographic interest.

- **Most significant bits** If $n - i \in O(\log n)$, then $r \in poly(n)$. Theorem 6 proves the security of the first $O(\log n)$ most significant bits for all N prime or RSA modulus for any one-way function defined over $\mathbb{Z}_N$ for which the multiplication code $\mathcal{C}^{B_i}$ is accessible.
- **Least significant bits** If $i \in O(\log n)$ then $m \in poly(n)$. Theorem 6 proves the security of the first $O(\log n)$ most significant bits for all N prime or RSA modulus for any one-way function defined over $\mathbb{Z}_N$ for which the multiplication code $\mathcal{C}^{B_i}$ is accessible.

6 The Security of All Bits for All N

This section is devoted to prove the hardness of all bits for all cryptographically relevant N. First of all in subsection 6.1 we study the bounds given in section 5.2 more accurately. In subsection 6.2 we proceed to prove that P is concentrated for N prime or RSA modulus. This result, together with the recovery algorithm given in subsection 6.3 and the accessibility of $\mathcal{C}^P$, implies the security of the internal bits for all N of cryptographic interest. This is summarized in theorem 9.

6.1 A Closer Look at the Asymptotic Behavior of $|\widehat{P(\alpha)}|^2$

The bounds of section 5.2 are not enough to prove the concentration in the general case. Therefore, in this section we will study the asymptotic behavior of $|\widehat{P(\alpha)}|^2$ in more detail.

As it was proven in proposition 1

$$|\widehat{P(\alpha)}|^2 = \Theta\left(\frac{abs_N(m\alpha)^2}{abs_N(\alpha)^2 abs_N(2^i\alpha - N/2)^2}\right).$$

We introduce some notation to express the elements $\alpha \in \mathbb{Z}_N$ as a function of some parameters useful to describe $abs_N(\alpha)$ and $abs_N(2^i\alpha - N/2)$. Recall that $N = r2^{i+1} \pm m$, where $0 < m < 2^i$.

Some parameters. We define some parameters depending on $\alpha \geq 0$ or $\alpha < 0$. First consider the case $\alpha \in [0, \frac{N-1}{2}]$. Denote $\delta_\alpha \equiv \alpha 2^i - \frac{N-1}{2} \mod N$, where $\delta_\alpha \in [-\frac{N-1}{2}, \frac{N-1}{2}]$ and let λ_α be the integer in $[0, 2^{i-1} - 1]$ such that

$$\alpha 2^i = (N-1)/2 + \delta_\alpha + \lambda_\alpha N. \tag{3}$$

Let α be an integer in $[-\frac{N-1}{2}, 0)$. Denote $\delta_\alpha \equiv \alpha 2^i + \frac{N+1}{2} \mod N$, where $\delta_\alpha \in [-\frac{N-1}{2}, \frac{N-1}{2}]$ and let λ_α be the integer in $[0, 2^{i-1} - 1]$ such that

$$\alpha 2^i = -(N+1)/2 + \delta_\alpha - \lambda_\alpha N. \tag{4}$$

Finally, for any $\alpha \in [-\frac{N-1}{2}, \frac{N-1}{2}]$ we define $\mu_\alpha \in [0, r]$ as the only integer such that $abs_N(2^i\alpha - (N-1)/2) = \mu_\alpha 2^i + \widetilde{\delta}_\alpha$ with $\widetilde{\delta}_\alpha \in [0, 2^i - 1]$.

From equations 3 and 4, if $\alpha \geq 0$,

$$\alpha = ((N-1)/2 + \delta_\alpha + \lambda_\alpha N)/2^i, \tag{5}$$

and if $\alpha < 0$,

$$\alpha = (-(N+1)/2 + \delta_\alpha - \lambda_\alpha N)/2^i. \tag{6}$$

We emphasize that equations 3, 4, 5 and 6 are integer equalities.

The parameters μ_α and $\widetilde{\delta}_\alpha$ are determined by δ_α. Indeed,

Lemma 6. *For all $\alpha \in \mathbb{Z}_N$, $\mu_\alpha 2^i + \widetilde{\delta}_\alpha = |\delta_\alpha|$.*

Proof. We will prove that $abs_N(2^i\alpha - (N-1)/2) = |\delta_\alpha|$. This is obvious when $\alpha \geq 0$, since $\alpha 2^i - (N-1)/2 \equiv \delta_\alpha \mod N$ and $\delta_\alpha \in [-\frac{N-1}{2}, \frac{N-1}{2}]$. When $\alpha < 0$, note that

$$\alpha 2^i - (N-1)/2 \equiv \alpha 2^i + (N+1)/2 \equiv \delta_\alpha \mod N.$$

Since $\delta_\alpha \in [-\frac{N-1}{2}, \frac{N-1}{2}]$, $abs_N(\alpha 2^i - (N-1)/2) = |\delta_\alpha|$ by definition of absolute value. □

Note that lemmas 4 and 5 in section 5.3 imply that $||P - P_{|\Gamma_{2^i} \cap \Gamma_0}||_2^2 \leq \epsilon$. That is, if there exists a set Γ where P is concentrated, then $\Gamma \subset \Gamma_0 \cap \Gamma_{2^i}$. The choice of these parameters is motivated by the remark that points $\alpha \in \Gamma_0 \cap \Gamma_{2^i}$ should be close to small odd multiples of $N/2^{i+1}$, that is, observing that $N/2^{i+1} \approx r$, we will have that $abs_N(\alpha) \approx (2\lambda_\alpha + 1)r \pm \mu_\alpha$, with λ_α and μ_α small. Indeed,

Lemma 7. *For all $\alpha \in \mathbb{Z}_N$, $abs_N(\alpha) = (2\lambda_\alpha + 1)r \pm \mu_\alpha + R$, with $|R| \leq \lambda_\alpha$.*

Proof. First of all we consider the case $\alpha \in [0, \frac{N-1}{2}]$. In this case $abs_N(\alpha) = \alpha$.

Suppose $\delta_\alpha \in [0, \frac{N-1}{2}]$ and $N = r2^{i+1} - m$ (i.e., Case 1 of section 5.1). Since $\delta_\alpha \geq 0$, lemma 6 implies $\delta_\alpha = \widetilde{\delta}_\alpha + \mu_\alpha 2^i$. Substituting in equation 5, we get

$$abs_N(\alpha) = (2\lambda_\alpha + 1)r + \mu_\alpha + \frac{-(2\lambda_\alpha + 1)m + 2\widetilde{\delta}_\alpha - 1}{2^{i+1}}.$$

Similarly for the rest of cases, that is: (1) $\delta_\alpha \in [-\frac{N-1}{2}, 0)$ and $N = r2^{i+1} - m$, (2) $\delta_\alpha \in [0, \frac{N-1}{2}]$ and $N = r2^{i+1} + m$ and (3) $\delta_\alpha \in [-\frac{N-1}{2}, 0)$ and $N = r2^{i+1} + m$, we obtain

$$abs_N(\alpha) = (2\lambda_\alpha + 1)r \pm \mu_\alpha + \frac{\pm(2\lambda_\alpha + 1)m \pm 2\widetilde{\delta}_\alpha - 1}{2^{i+1}}.$$

On the other hand, in the case $\alpha < 0$, $abs_N(\alpha) = -\alpha$. Considering all the possible combinations for the sign of δ_α and Case 1 and Case 2, as above, and substituting in equation 6, we obtain

$$abs_N(\alpha) = (2\lambda_\alpha + 1)r \pm \mu_\alpha + \frac{\pm(2\lambda_\alpha + 1)m \pm 2\widetilde{\delta}_\alpha + 1}{2^{i+1}}.$$

If $\lambda_\alpha = 0$ it is easy to see that $R = 0$ and the lemma is true. Indeed, $\lambda_\alpha = 0$ implies $\pm m \pm 2\widetilde{\delta}_\alpha \pm 1 = 0 \mod 2^{i+1}$ due to the fact that equations 3 and 4 were integer equalities. Because of the range of definition of m and $\widetilde{\delta}_\alpha$, this congruence is equivalent to the equality $\pm m \pm 2\widetilde{\delta}_\alpha \pm 1 = 0$ and therefore $R = 0$. Now, to prove the lemma proceed by induction over λ_α. □

Corollary 1. *For all $\alpha \in \mathbb{Z}_N$, $abs_N(\alpha) \geq \lambda_\alpha(2r - 1)$.*

Proof. Simply note that $\mu_\alpha \in [0, r]$ and $|R| \leq \lambda_\alpha$. Then,

$$abs_N(\alpha) = (2\lambda_\alpha + 1)r \pm \mu_\alpha + R \geq 2\lambda_\alpha r + R \geq \lambda_\alpha(2r - 1). \qquad \square$$

From these lemmas we can easily prove the following:

Lemma 8. $abs_N(\alpha)^2 abs_N(2^i\alpha - \frac{N-1}{2})^2 \geq \lambda_\alpha^2 \mu_\alpha^2 r^2 2^{2i+2} \frac{1}{4}$.

Proof. As we have seen in corollary 1, $abs_N(\alpha) \geq \lambda_\alpha(2r - 1)$, therefore

$$abs_N(\alpha)^2 abs_N(2^i\alpha - \tfrac{N-1}{2})^2 \geq \lambda_\alpha^2(2r-1)^2(\mu_\alpha 2^i + \widetilde{\delta}_\alpha)^2 \geq \lambda_\alpha^2(2r-1)^2(\mu_\alpha 2^i)^2 \geq$$
$$\geq \lambda_\alpha^2 \frac{(2r-1)^2}{(2r)^2}(2r)^2(\mu_\alpha 2^i)^2 \geq \lambda_\alpha^2 \mu_\alpha^2 r^2 2^{2i+2} \frac{1}{4}. \qquad \square$$

Now it is easy to characterize the asymptotic behavior of $|\widehat{P(\alpha)}|^2$ in terms of λ_α and μ_α.

Proposition 2. *For all $\alpha \in \mathbb{Z}_N$ such that $\lambda_\alpha > 0$ and $\mu_\alpha > 0$*

$$|\widehat{P(\alpha)}|^2 < O(\frac{1}{\lambda_\alpha^2 \mu_\alpha^2}).$$

Proof. Since $abs_N(m\alpha) \leq N/2$, from proposition 1 and the lemma 8, if $\lambda_\alpha > 0$ and $\mu_\alpha > 0$,

$$|\widehat{P(\alpha)}|^2 < O\left(\frac{abs_N(m\alpha)^2}{abs_N(\alpha)^2 abs_N(2^i\alpha - N/2)^2}\right) < O(\frac{N^2}{\lambda_\alpha^2 \mu_\alpha^2 r^2 2^{2i+2}}) < O(\frac{1}{\lambda_\alpha^2\mu_\alpha^2}).\square$$

Before proceeding to prove our main theorem, we note that elements in $\mathbb{Z}_N$ have a convenient representation in these parameters.

Lemma 9. *The following map is injective*

$$\begin{array}{rcl} \pi : [-\frac{N-1}{2}, \frac{N-1}{2}] & \longrightarrow & [0, 2^{i-1}-1] \times [0,r] \times \{\pm 1\} \times \{\pm 1\} \\ \alpha & \longmapsto & (\ \lambda_\alpha, \quad \mu_\alpha, \quad s_\alpha, \quad s_\delta\ ) \end{array}$$

where $s_\alpha = 1$ *if* $\alpha \geq 0$ *and* -1 *otherwise and* $s_\delta = 1$ *if* $\delta_\alpha \geq 0$ *and* -1 *otherwise.*

Proof. Reducing equations 3 and 4 modulo 2^i it is clear that λ_α and s_δ determine $\widetilde{\delta}_\alpha$ modulo 2^i and therefore $\widetilde{\delta}_\alpha$. Then α can be computed from $\pi(\alpha)$ using equations 5 or 6. □

6.2 The Concentration of the ith Bit for All N

As a result of lemma 9 of the above section, we can describe the elements of $\mathbb{Z}_N$ as regions in $[0, 2^{i-1}-1] \times [0,r] \times \{\pm 1\} \times \{\pm 1\}$. Now we can present our result about the concentration of the ith bit.

Theorem 7. *P is ϵ-concentrated in* $\Gamma \overset{\mathbf{def}}{=} \{\chi_\alpha : \lambda_\alpha < O(\frac{1}{\epsilon}), \mu_\alpha < O(\frac{1}{\epsilon})\}$.

Proof. Let $\Gamma_k \overset{\mathbf{def}}{=} \{\chi_\alpha : \lambda_\alpha \leq k, \mu_\alpha \leq k\}$, we will prove that

$$\sum_{\chi_\alpha \notin \Gamma_k} |\widehat{P(\alpha)}|^2 < O(\frac{1}{k}).$$

Note that $\mathbb{Z}_N \backslash \Gamma_k = \{\chi_\alpha : \lambda_\alpha = 0, \mu_\alpha > k\} \cup \{\chi_\alpha : \lambda_\alpha > k, \mu_\alpha = 0\} \cup \{\chi_\alpha : \lambda_\alpha > k, \mu_\alpha \geq 1\} \cup \{\chi_\alpha : \lambda_\alpha \geq 1, \mu_\alpha > k\}$.

To bound the sum of $|\widehat{P(\alpha)}|^2$ over the two first sets, the bounds of lemma 3 of section 5.2 will suffice, while we will need proposition for the other bounds. Indeed, when $\lambda_\alpha = 0$, using one of the bounds of lemma 3,

$$\begin{aligned} \sum_{\lambda_\alpha=0,\mu_\alpha>k} |\widehat{P(\alpha)}|^2 \leq O(m^2) & \sum_{\lambda_\alpha=0,\mu_\alpha>k} \frac{1}{abs_N(2^i\alpha - (N-1)/2)^2} < \\ < O(m^2) & \sum_{\lambda_\alpha=0,\mu_\alpha>k} \frac{1}{(\widetilde{\delta}_\alpha + \mu_\alpha 2^i)^2} < \\ < O(m^2) & \sum_{\lambda_\alpha=0,\mu_\alpha>k} \frac{1}{(\mu_\alpha 2^i)^2} < \\ < O(1) & \sum_{\lambda_\alpha=0,\mu_\alpha>k} \frac{1}{\mu_\alpha^2} < O(\frac{1}{k}). \end{aligned}$$

On the other hand, when $\mu_\alpha = 0$, using the other bound of lemma 3,

$$\sum_{\lambda_\alpha > k, \mu_\alpha = 0} |\widehat{P(\alpha)}|^2 \leq O(r^2) \sum_{\lambda_\alpha > k, \mu_\alpha = 0} \frac{1}{abs_N(\alpha)^2} <$$

$$< O(r^2) \sum_{\lambda_\alpha > k, \mu_\alpha = 0} \frac{1}{\lambda_\alpha^2 (2r-1)^2} <$$

$$< O(1) \sum_{\lambda_\alpha > k, \mu_\alpha = 0} \frac{1}{\lambda_\alpha^2} < O(\frac{1}{k}).$$

To conclude the proof we need to show that

$$\sum_{\lambda_\alpha > k, \mu_\alpha \geq 1} |\widehat{P(\alpha)}|^2 + \sum_{\lambda_\alpha \geq 1, \mu_\alpha > k} |\widehat{P(\alpha)}|^2 < O(\frac{1}{k}).$$

From proposition 6.2,

$$|\widehat{P(\alpha)}|^2 < O(\frac{1}{\lambda_\alpha^2 \mu_\alpha^2}).$$

As a consequence,

$$\sum_{\lambda_\alpha > k, \mu_\alpha \geq 1} |\widehat{P(\alpha)}|^2 + \sum_{\lambda_\alpha \geq 1, \mu_\alpha > k} |\widehat{P(\alpha)}|^2 \leq \sum_{\lambda_\alpha > k, \mu_\alpha \geq 1} \frac{1}{\lambda_\alpha^2 \mu_\alpha^2} + \sum_{\lambda_\alpha \geq 1, \mu_\alpha > k} \frac{1}{\lambda_\alpha^2 \mu_\alpha^2} =$$

$$= \sum_{\mu_\alpha \geq 1} \frac{1}{\mu_\alpha^2} (\sum_{\lambda_\alpha > k} \frac{1}{\lambda_\alpha^2}) + \sum_{\lambda_\alpha \geq 1} \frac{1}{\lambda_\alpha^2} (\sum_{\mu_\alpha > k} \frac{1}{\mu_\alpha^2}) < O(\frac{1}{k}).$$

We conclude that if $k \in O(\frac{1}{\epsilon})$, $\sum_{\chi_\alpha \notin \Gamma} |\widehat{P(\alpha)}|^2 \leq \epsilon$ as stated in theorem 7.

□

6.3 The Hardness of the ith Bit for All N

In the previous section we proved the concentration of the predicate B_i. To complete the proof of the main theorem concerning its hardness, in this section we prove the recoverability of the code $\mathcal{C}^{B_i}$.

Theorem 8. *The code $\mathcal{C}^{B_i}$ is recoverable for all N prime or RSA modulus.*

Proof. This recovery algorithm is almost identical to the one given in [1].

Because of theorem 7 and lemma 1, C_x^P is τ-concentrated in $\Gamma' \stackrel{\mathbf{def}}{=} \{\chi_\beta : \beta = \alpha x \mod N, \chi_\alpha \in \Gamma\}$, where $\Gamma \stackrel{\mathbf{def}}{=} \{\chi_\alpha : \lambda_\alpha < O(\frac{1}{\tau}), \mu_\alpha < O(\frac{1}{\tau})\}$.

The inputs of the recovery algorithm are a character χ_β and a threshold parameter τ, where $1/\tau \in poly(n)$. The output is a list containing $x \in \mathbb{Z}_N$ such that $\chi_\beta \in Heavy_\tau(C_x^P)$.

Since, C_x^P is τ-concentrated in Γ', $\chi_\beta \in Heavy_\tau(C_x^P)$ implies $\chi_\beta \in \Gamma'$ and thus $\beta = \alpha x \mod N$ for $\lambda_\alpha < poly(n/\tau)$ and $\mu_\alpha < poly(n/\tau)$. The algorithm outputs the list $L \stackrel{\mathbf{def}}{=} \{x : x = \beta/\alpha \mod N, \chi_\alpha \in \Gamma\}$. The length of the list and the time of constructing it is $poly(n/\tau)$. □

Theorem 2 states that, as a consequence of theorems 7 and 8, the code $\mathcal{C}^{B_i}$ is list decodable for all N prime or RSA modulus. We conclude

Theorem 9. *If N is prime or is an RSA modulus, the predicate B_i is hard-core for any one-way function defined over $\mathbb{Z}_N$ for which the multiplication code is accessible.*

7 All Bits of the Paillier Encryption Scheme Are Secure

Let $N = p \cdot q$ be an RSA modulus, given an element $g \in \mathbb{Z}_{N^2}^*$ such that N divides the order of g, the Paillier trapdoor permutation, introduced in [10] is the map:

$$\begin{array}{rcl} \mathcal{E}_g : \mathbb{Z}_N^* \times \mathbb{Z}_N & \longrightarrow & \mathbb{Z}_{N^2}^* \\ (r, m) & \mapsto & r^N \, g^m \end{array}$$

Taking r to be a random element and m the plaintext, the Paillier probabilistic encryption scheme encrypts m as $\mathcal{E}_g(r, m)$ and is semantically secure under the Decisional Composite N-th residuosity assumption.

In this section we will sketch the proof of the security of any bit P of the message. We stress that by security of P we mean that we relate the ability of an adversary in predicting $P(m)$ from $\mathcal{E}_g(r, m)$ to the ability of recovering m from $\mathcal{E}_g(r, m)$, and not to the ability of inverting $\mathcal{E}_g$.

The concentration and recoverability of the multiplication code $\mathcal{C}^P$, where $P : \mathbb{Z}_N \to \{\pm 1\}$ is the predicate ith bit of the message, follows from our results of section 6.3. Then, the code $\mathcal{C}^P$ is list decodable.

The one-way function $\mathcal{E}_g$ has domain in $\mathbb{Z}_N^* \times \mathbb{Z}_N$, while the predicate has domain in $\mathbb{Z}_N$, so we need to slightly change the definition of accessibility to fit this situation. We will first give the access algorithm, then we will give the new definition of accessible code and argue that this new definition is enough to apply the list decoding methodology.

The access algorithm $\mathcal{A}$, on input $(N, g, \mathcal{E}_g(r, x), j)$, chooses a random element $\ell \in \mathbb{Z}_N^*$, and outputs $(\mathcal{E}_g(r, x))^j \cdot \ell^N$. Note that

$$(\mathcal{E}_g(r, x))^j \cdot \ell^N \equiv \mathcal{E}_g(r^j \cdot \ell, xj) \mod N^2.$$

It is not hard to see that for this access algorithm $\mathcal{A}$ the code satisfies the following properties:

1. **Code access:** $\forall x, j \in \mathbb{Z}_N$, $\mathcal{A}(N, g, \mathcal{E}_g(r, x), j)$ returns $\mathcal{E}_g(r', x')$ such that $C_x^{P_{N,g}}(j) = P_{N,g}(x')$
2. **Well spread:** For uniformly distributed $C_x^{P_{N,g}} \in \mathcal{C}^{P_{N,g}}$ and $j \in \mathbb{Z}_N$, the distribution of $(r', x') \in \mathbb{Z}_N^* \times \mathbb{Z}_N$ satisfying $\mathcal{E}_g(r', x') = \mathcal{A}(N, g, \mathcal{E}_g(r, x), j)$ is statistically close to the uniform distribution on $\mathbb{Z}_N^* \times \mathbb{Z}_N$
3. **Bias preserving:** For every codeword $C_x^{P_{N,g}} \in \mathcal{C}^{P_{N,g}}$,

$$|\Pr\bigl(C_x^{P_{N,g}}(j) = 1 : j \leftarrow \mathbb{Z}_N\bigr) - \Pr\bigl(P_{N,g}(z) = 1 : z \leftarrow \mathbb{Z}_N\bigr)| \leq \nu(n),$$

where ν is a negligible function.

Compare these properties with the ones that an accessible code must verify (see definition 10 of section 4). Both definitions are almost identical but now the property that the code is well spread is for $(r', x') \in \mathbb{Z}_N^* \times \mathbb{Z}_N$ and not on the domain of the definition of the code.

Lemmas 2 and 3 of [1] prove that if the code $\mathcal{C}^P$ is accessible with respect to F, an oracle $\mathcal{B}$ predicting P from F with probability exceeding $maj_P + \epsilon$ implies access to a corrupted codeword w such that $\Delta(C_x, w) \leq minor_{C_x} - \epsilon$. The property that the code is well spread is used in the proofs of these lemmas. It is immediate to see that to do the same reasoning in the Paillier case we need to require precisely the second condition above.

Summarizing, the definition we gave above is exactly the one we need to prove that an oracle predicting $P(x)$ from $\mathcal{E}_g(r', x)$ gives access to a corrupted codeword w sufficiently close to x. But to prove theorem 3 which states that list decodability of $\mathcal{C}^P$ plus accessibility implies that P is a hard-core of F, accessibility was only necessary to prove access to a corrupted codeword w. Therefore, the argument we gave in this section with the concentration and recoverability of the multiplication code $\mathcal{C}^{P_{N,g}}$ of section 6.3 implies the security of all bits of the message of the Paillier trapdoor permutation.

We emphasize that the security of all bits of the Paillier encryption scheme was only known based on a non-standard computational assumption [3].

8 Other Predicates

We note that theorem 6 is enough to reprove the hardness of segment predicates. Recall from section 2 that a t-segment predicate is a predicate which changes value $t \in poly(n)$ number of times. Define G as

$$G(x) = \frac{P(x+1) - P(x)}{2},$$

and note that $G(x) \neq 0$ for exactly t values of x. Although in this case we cannot compute $\widehat{G(\alpha)}$ explicitly as before, we still have $|\widehat{G(\alpha)}|^2 \leq t^2/N^2$. The same arguments as in lemma 5 prove that P is concentrated up to ϵ in $\Gamma_0 \overset{\mathbf{def}}{=} \{\chi_\alpha : abs_N(\alpha) \leq O(\frac{t^2}{\epsilon})\}$ - this corresponds to claim 4.1 of [1].

In the last section we proved the security of all bits in the binary representation of the preimage for any one-way function defined over $\mathbb{Z}_N$ with multiplicative access provided that N is odd. Note that the same proof would do for any other "almost periodic" predicate. Indeed, for any $d \in \mathbb{N}$ define $P_d : \mathbb{Z}_N \to \{\pm 1\}$ as 1 if $[x] \in [kd, (k+1)d - 1]$, k even, and -1 otherwise. Write $N = r2d \pm m$, with $0 < m < d$. Then all the results proven in the last section are also valid for P_d just writing d instead of 2^i.

9 Conclusion

In our opinion the list decoding methodology formalized in [1] has not received enough attention. Because of the elegance and generality of the method and the power of the different tools it uses it should be considered the starting point of any bit security proof. In this paper we have extended the number of predicates to which the list decoding methodology applies. As a result we prove the security of all bits of any of the usual cryptographic one-way functions with multiplicative access defined on a cyclic group of order N.

Acknowledgements

The authors would like to thank Eike Kiltz for his comments and suggestions that contributed to make the paper more readable.

References

1. Akavia, A., Goldwasser, S., Safra, S.: Proving Hard-Core Predicates Using List Decoding. In: Proc. of the 44th Symposium on Foundations of Computer Science (2003)
2. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.P.: RSA and Rabin functions: certain parts are as hard as the whole. SIAM J.Comp. 17(2) (1988)
3. Catalano, D., Gennaro, R., Howgrave-Graham, N.: Paillier's Trapdoor Function Hides up to $O(n)$ Bits. J.Cryptology 15(4) (2002)
4. Kushilevitz, E., Mansour, Y.: Learning Decision Trees Using the Fourier Spectrum. In: Proc. of the 23rd Annual ACM Symposium on Theory of Computing (1991)
5. Gilbert, A.C., Muthukrishnan, S., Strauss, M.: Improved time bounds for near-optimal sparse Fourier representation via sampling. In: Proc. of SPIE Wavelets XI (2005)
6. Goldreich, O., Levin, L.: A hard-core predicate for all one-way functions. In: Proc. of the 21st Annual ACM Symposium on Theory of Computing (1989)
7. Goldreich, O., Rubinfeld, R., Sudan, M.: Learning Polynomials with Queries: The Highly Noisy Case. SIAM J. Discrete Math. 13(4) (2000)
8. Håstad, J., Näslund, M.: The security of all RSA and discrete log bits. J. ACM 51(2) (2004)
9. Näslund, M.: All Bits ax+b mod p are Hard. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 114–128. Springer, Heidelberg (1996)
10. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)

A New Lattice Construction for Partial Key Exposure Attack for RSA

Yoshinori Aono

Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology, Tokyo, Japan
aono5@is.titech.ac.jp

Abstract. In this paper we present a new lattice construction for a lattice based partial key exposure attack for the RSA cryptography. We consider the situation that the RSA secret key d is small and a sufficient amount of the LSBs (least significant bits) of d are known by the attacker. We show that our lattice construction is theoretically more efficient than known attacks proposed in [2,7].

Keywords: RSA, cryptanalysis, partial key exposure attack, lattice basis reduction, the Coppersmith technique.

1 Introduction

In this paper we present a new lattice construction for a lattice based partial key exposure attack for the RSA cryptography in the situation that the secret key d is small and its LSBs (least significant bits) are exposed.

Boneh and Durfee [2] proposed the lattice based attack for the RSA cryptography. Its basic idea is to reduce the RSA key finding problem to problems of finding small roots of a modular equation such as $f(x_1, \ldots, x_n) \equiv 0 \pmod{W}$, which are solved by the Coppersmith technique [6], the technique that solves a given modular equation by converting it to an algebraic equation by using a lattice basis reduction algorithm such as the LLL algorithm [11]. Boneh and Durfee [2] showed that the secret key d can be computed from a public key pair e and N in polynomial time in $\log N$ when $d < N^{0.292}$.

Since Boneh and Durfee's work, many of its variants have been proposed [4,7]. Blömer and May [4] extended the technique for a partial key exposure attack, i.e., a problem of computing d from e, N and some partial information on d. This approach has been further extended by Ernst et al. [7] for several partial key exposure situations. In this paper we consider one of those situations where the secret key d is small and a sufficient amount of d is given (besides e and N), and we show an improvement over the algorithm by Ernst et al.[7], thereby solving an open problem raised in their paper.

In order to state our improvement we need some notations; see the next section for the precise definition. Let (e, N) be an RSA public key pair and let d be its corresponding secret key. Here as usual we use $\ell_N =$ (the bit-length of N) as

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 34–53, 2009.

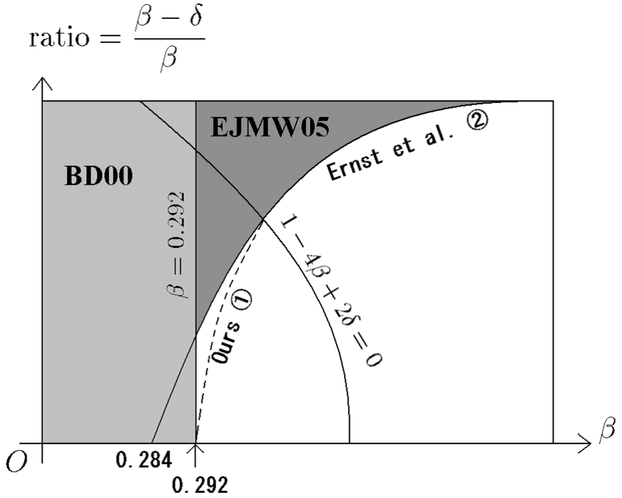


Fig. 1. Our recoverable range. The limit of Boneh and Durfee (1) and that of Ernst et al. (2) are corresponding to the left/above of the line $\beta = 0.292$ and the line `Ernst et al.` ② respectively. Our new improvement area (3) is the left side of the dashed line `Ours`① in the area left of line $1 - 4\beta + 2\delta = 0$.

a security parameter. We consider the situation that $\ell_d =$ (the bit length of d) is relatively small compared with ℓ_N, and some ℓ_0 least significant bits of d are known. Let $\beta = \ell_d/\ell_N$ and $\delta = (\ell_d - \ell_0)/\ell_N$; that is, they are respectively the ratios of the bit-length of d and its unknown part. Now the asymptotic performance of the algorithms in [2,7] can be summarized in Figure 1. (This is a rough image, not accurate.)

The algorithm of [2] works asymptotically when the parameters take values in the left of a vertical line labelled "$\beta = 0.292$". That is, it obtains the secret key for

$$\beta < 1 - \frac{1}{\sqrt{2}} = 0.292\text{... and any } \delta. \tag{1}$$

The algorithm of [7] works when

$$\delta < \frac{5}{6} - \frac{1}{3}\sqrt{1 + 6\beta}. \tag{2}$$

That is it works when β and δ take values in the left/above of a curve labelled `Ernst et al.`②. As shown in the Figure 1, the algorithm of [7] improves the solvable parameter range when δ is small; it has been however left open [7] to develop an algorithm that has a better solvable parameter range than both [2] and [7].

In this paper we propose an algorithm that can work asymptotically when the parameters take values in the left/above of the dashed line of Figure 1. More precisely, it works when

$$1 - 4\beta + 2\delta > 0 \text{ if } 2\sqrt{2(1-2\beta)(\beta-\delta)}(\delta-\beta) - 2\beta^2 + 3\beta + \delta - 1 < 0 \quad (3)$$

and

$$1 - 4\beta + 2\delta \le 0 \text{ if } \delta < \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta}. \quad (4)$$

Note that the range (3) is our new improvement area which is the left/above of the dashed line `Ours` ① in Figure 1, while the range (4) is already given by [7]. Consider the situation that we do not have any information on d, i.e., $\delta = \beta$. Substituting this to (3) and (4), we have $\beta < 1 - \sqrt{2}/2 \approx 0.292$ and $\beta < 7/6 - \sqrt{7}/3 \approx 0.284$ respectively. This means our range covers [7]'s range when $\delta \approx \beta$.

This paper is organized as follows. In Section 2, we introduce some notations and lemmas about the lattice based partial key exposure attack. Section 3 provides the overview of the lattice based partial key exposure attack. In Section 4 we describe the construction and the performance of our lattice. Section 5 provides the results of our computer experiments. The analysis of Section 4 is explained in Section 6.

2 Preliminaries

We introduce some notations and state some known facts used in the following discussions. Then we review some key technical lemmas used in the lattice based attack.

We use standard RSA notations throughout this paper. A given RSA instance is defined by p, q, e, and d, where p and q are large primes, e is a public key, and d is a secret key. Let $N = p \times q$, and let $\varphi(N)$ be the Euler's function; here we will simplify assume that $\varphi(N) = (p-1)(q-1)$. The key relation is

$$ed \equiv 1 \pmod{\varphi(N)}. \quad (5)$$

The partial key exposure attack is to compute the secret key d from partial information on d, and the public key (e, N). In this paper, we consider the situation that some LSBs of d are exposed, that is, recovering d from LSBs of d (together with e and N). We use d_0 to denote the exposed part and $\tilde{d}$ to denote the non-exposed part. That is, we assume that

$$d = \tilde{d} \cdot M + d_0, \quad (6)$$

where $M = 2^k$ and $k = \lg(d_0)$[1]. We will use M for denoting this number throughout this paper. Define $\beta = \log_N d$ and $\delta = \log_N \tilde{d}$. That is, β and δ are the rough ratios of the bit-length of d and $\tilde{d}$ relative to that of N respectively.

In the algorithm, we need to solve a modular equation such as $f(x, y) \equiv 0 \pmod{W}$ for a polynomial $f(x, y)$. Furthermore, we want to obtain a solution in a certain range. In general, this task is not easy. However there are some cases

[1] We use $\lg(x)$ to denote the length of the binary representation of x.

where we may be able to use the standard numerical method for solving modular equations. The Howgrave-Graham lemma [9] provides us with one of such cases.

In order to state the Howgrave-Graham Lemma, we introduce the following notation.

Definition 1 ***XY*-norm.** *Let X and Y be natural numbers and $f(x,y) = \sum_{i,j} a_{i,j}x^i y^j$ be a polynomial with integral coefficient.*

We denote the length of a coefficient vector of $f(Xx, Yy)$ by $||f(x,y)||_{XY}$, i.e.,

$$||f(x,y)||_{XY} \overset{\text{def}}{=} \sqrt{\sum_{i,j} a_{i,j}^2 X^{2i} Y^{2j}}.$$

We call this the XY-norm of $f(x,y)$.

Lemma 1 (Howgrave-Graham [9]). *For any positive integers X, Y and W, let $f(x,y)$ be a bivariate polynomial consisting of w terms with integral coefficient. We suppose that the following holds*

$$||f(x,y)||_{XY} < \frac{W}{\sqrt{w}}.$$

Then we have

$$f(x,y) \equiv 0 \ (mod\ W) \Leftrightarrow f(x,y) = 0$$

within the range of $|x| < X$ and $|y| < Y$.

Note that $f(x,y) = 0$ clearly implies $f(x,y) \equiv 0 \pmod W$. What is important is its converse. This lemma guarantees that the solution of $f(x,y) \equiv 0 \pmod W$ in the target range can be found (if they exist) from the solutions of $f(x,y) = 0$, which can be obtained by the standard numerical method.

In order to use the lemma, we need to obtain a polynomial with a small XY-norm. The key idea of the lattice based attack is to formulate this task as the shortest vector problem and use approximate solutions computed by a polynomial time lattice basis reduction algorithm for the shortest vector problem.

We introduce some definitions and some lemmas about the lattice. Consider linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^{\tilde{n}}$, then the lattice with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is defined by

$$L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i \middle| a_i \in \mathbb{Z} \text{ for } i = 1, \ldots, n \right\}. \tag{7}$$

That is, the lattice is the set of integral linear combinations of its basis vectors. We denote by n a number of vectors, which is usually called *lattice dimension*, and denote by $\tilde{n}$ a number of component of vector in basis, which we call *lattice component size*. Note that the lattice is a additive subgroup of $\mathbb{R}^{\tilde{n}}$.

The shortest vector problem, for given basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$, is to find a vector $\mathbf{v}$ such that $\mathbf{v} \in L(\mathbf{b}_1, \ldots, \mathbf{b}_n) \setminus \{\mathbf{0}\}$ and $|\mathbf{v}| \le |\mathbf{v}'|$ for $^\forall \mathbf{v}' \in L(\mathbf{b}_1, \ldots, \mathbf{b}_n) \setminus \{\mathbf{0}\}$.

That is, this problem is to find a non-zero vector having the minimum length in $L(\mathbf{b}_1, \ldots, \mathbf{b}_n)$.

In order to obtain polynomials with small XY-norms, we need to compute short vectors as approximate solutions of this problem. We will use a polynomial time algorithm, named LLL, proposed in [11]. Some improvements have been proposed [13,14], but as shown later, these improvements are not essential for our application.

The approximation ratio of the LLL algorithm is exponential, it is however enough for our propose. The following theorem guarantees the upper bounds of the length of the computed vectors. The LLL algorithm computes a special basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$, named reduced basis, from given basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$. Our interest is short vectors in the reduced basis in the following theorem.

Theorem 1. *[2, Fact 3.3] Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a given linearly independent basis. Then we can find linearly independent lattice vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ such that*

$$\begin{aligned} |\mathbf{v}_1| &\leq 2^{(n-1)/4} |\det(L)|^{1/n}, \text{ and} \\ |\mathbf{v}_2| &\leq 2^{n/2} |\det(L)|^{1/(n-1)}. \end{aligned} \tag{8}$$

Here, L is the lattice with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$, and $\det(L)$ is the determinant of the lattice defined by using their Gram-Schmidt orthogonal basis $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$ as follows

$$\det(L) = \prod_{i=1}^{n} |\mathbf{b}_i^*|. \tag{9}$$

We will use (9) to evaluate the determinant of our lattice in the later section.

Note that the shortest vector problem is defined on vectors, while our targets are polynomials. Thus we consider a way to map polynomials to vectors. For example, the polynomial $f(x,y) = -3x^3 + 4x^2y - 2xy^2 + 7xy^3$ is mapped to the vector $(-3X^3, 4X^2Y, -2XY^2, 7XY^3)$ by some natural numbers X and Y. To state this correspondence formally, we first need to fix some linear ordering on pairs (i,j) of nonnegative integers. With respect to this ordering let $(i(t)$,$j(t))$ denote the t-th pair. Then our correspondence between polynomials and vectors is defined as follows.

Definition 2 Polynomials ↔ vectors. *Let J be a sequence of pairs of non-negative integers, where we assume some linear order on J, let it be fixed, and let $\tilde{n}$ denote $|J|$, the length of the sequence. We also fix some positive integers X and Y. W.r.t. these X and Y, for any $f(x,y) = \sum_{1 \leq t \leq \tilde{n}} a_{i(t),j(t)} x^{i(t)} y^{j(t)}$, the following vector $\mathbf{b}$ is the vectorisation of $f(x,y)$ with parameter X and Y, and it is denoted by $\mathcal{V}_J(f; X, Y)$.*

On the other hand, for any $\mathbf{b}$ of size $\tilde{n}$, a polynomial $f(x,y)$ defined from $\mathbf{b}$ by interpreting it as below is called the functionalisation of $\mathbf{b}$ and it is denoted by $\mathcal{F}_J(\mathbf{b}; X, Y)$.

$$\begin{array}{lcccccc} f(x,y) = & a_{i(1),j(1)}x^{i(1)}y^{j(1)} & + & a_{i(2),j(2)}x^{i(2)}y^{j(2)} & + \cdots + & a_{i(|J|),j(|J|)}x^{i(|J|)}y^{j(|J|)} & \\ & \downarrow & & \downarrow & & \downarrow & \\ \mathbf{b} \quad = (& a_{i(1),j(1)}X^{i(1)}Y^{j(1)} & , & a_{i(2),j(2)}X^{i(2)}Y^{j(2)} & , \ldots , & a_{i(|J|),j(|J|)}X^{i(|J|)}Y^{j(|J|)} &). \end{array}$$

Remark. When J is clear from the context, we often omit J and write as $\mathcal{V}(f;X,Y)$ and $\mathcal{F}(\mathbf{b};X,Y)$. Then from the definition, the following relationships are immediate.

$$\begin{aligned}&||f(x,y)||_{XY} = |\mathcal{V}(f;X,Y)|, \text{ and}\\ &||\mathcal{F}(\mathbf{b};X,Y)||_{XY} = |\mathbf{b}|.\end{aligned} \tag{10}$$

That is, these are equivalent to the length of a coefficient vector of $f(Xx,Yy)$.

3 Overview of the Partial Key Exposure Attack

We give an overview of the lattice based partial key exposure attack in the situation that LSBs of d are exposed. The goal of the attack is to compute the secret key d from d_0, least significant bits of d, and a given public key pair. The lattice based attack achieves this goal by using a lattice reduction algorithm and the Howgrave-Graham lemma. It is said in [7] (and some papers) that the attack is effective if

(i) d, and unknown part of d are short,
(ii) e and N are of similar bit length, and
(iii) p and q are of similar bit length.

In order to be precede, we consider in this paper, the following conditions.

(a) $\delta = \log_N \tilde{d}$ is smaller than 0.5,
(b) $\lg(e) = \lg(N)$, and
(c) $\lg(p) = \lg(q)$

In the following, we assume all parameters satisfy these conditions. More precisely, we will use the following inequalities in the later.

$$e < \varphi(N) \text{ and } p + q < 3\sqrt{N}. \tag{11}$$

Our objective is to compute d from a public key pair (e,N) and d_0. As explained in Introduction, the key relation is the modular equation (5), from which it is easy to derive $ed = 1 - x\varphi(N) = 1 - x(y+N)$ for some $x, y \in \mathbb{Z}$. Also by using (6), we can deduce from the above that $e(\tilde{d}\cdot M + d_0) = 1 - x(y+N)$ and hence we have

$$x(N+y) + (ed_0 - 1) \equiv 0 \pmod{eM}. \tag{12}$$

We show here that it is relatively easy to enumerate all solutions (x,y) of (12). First note that a solution (x,y) exists if for integer y,

$$\gcd\left(\frac{N+y}{g}, \frac{eM}{g}\right) = 1 \text{ where } g = \gcd(N+y, eM, ed_0 - 1).$$

In fact in this case, we can compute x by $x = \left(\frac{1-ed_0}{g}\right)\cdot\left(\left(\frac{N+y}{g}\right)^{-1} \bmod \frac{eM}{g}\right)$. But clearly what we need is some specific solution of (12). Among solutions (x,y) of (12), we say that (x_0,y_0) is *useful* if it indeed satisfies the following equation, from which we can recover the secret key.

$$d = \frac{1 - x_0(N+y_0)}{e} \tag{13}$$

Thus, our task is not computing *some* solutions (x, y), but computing this useful solution among (x, y) satisfying (12). Below we use (x_0, y_0) to denote this useful solution. Let us consider a size of the useful solution (x_0, y_0). We have the following upper bounds. Here, we use (11) and the fact that $\varphi(N) = N + y_0$ if (x_0, y_0) is the useful solution.

$$\begin{aligned} |x_0| &= \left| \frac{ed-1}{N+y_0} \right| < \frac{ed}{\varphi(N)} < d = N^{\beta}, \text{ and} \\ |y_0| &= |N - \varphi(N)| = p + q - 1 < 3N^{0.5}. \end{aligned} \tag{14}$$

Now let $X = \lceil N^{\beta} \rceil$ and $Y = \lceil 3N^{0.5} \rceil$. Then, the useful solution (x_0, y_0) is a solution of (12) satisfying $|x_0| < X$ and $|y_0| < Y$.

Conversely, we consider some heuristic condition on δ for a solution satisfying $|x| < X$ and $|y| < Y$ is useful. We assume that solutions of (12) are random numbers on $\{0, \ldots, eM-1\}^2$. Since the number of solution pairs of (12) is smaller than eM, we expect the number of solutions satisfy $|x| < X$ and $|y| < Y$ is smaller than

$$eM \cdot \frac{4XY}{(eM)^2} = \frac{4XY}{eM} \approx \frac{4 \cdot N^{\beta} \cdot 3N^{0.5}}{N \cdot N^{\beta-\delta}} \approx N^{\delta - 0.5}.$$

Thus, if this value is smaller than 1, we may expect a solution within the range $|X| < N^{\beta}$ and $|y| < N^{0.5}$ is only one, which is the useful solution guaranteed by (14). From this observation we propose a condition $\delta < 0.5$ and the following heuristic assumption.

Heuristic Assumption. Consider the case $\delta < 0.5$. Then, there is only useful solution (x_0, y_0) within the range of $|x| < N^{\beta}, |y| < 3N^{0.5}$ of the following equation.

$$x(N+y) + (ed_0 - 1) \equiv 0 \pmod{eM}$$

Furthermore we can recover the secret key d by (13)[2]. □

Remark. This assumption shows we can obtain the secret key by the exhaustive search when $\delta < 0.5$.

For our discussion, let us define the following two functions[3]

$$\begin{aligned} f_{\text{main}}(x, y) &\stackrel{\text{def}}{=} x(N+y) + (ed_0 - 1) \\ &= (ed_0 - 1) + Nx + xy, \text{ and} \end{aligned} \tag{15}$$

$$f_{\text{M}}(x, y) \stackrel{\text{def}}{=} M(-1 + x(N+y)). \tag{16}$$

[2] Moreover we can compute the factoring of N by $\varphi(N) = N + y_0$.

[3] Ernst et al. [7] reduced the problem to the problem of finding small solution of an algebraic equation

$$f_{\text{LSB}}(x, y, z) = eMx - Ny + yz + e\tilde{d} - 1 = 0.$$

1. Based on $f_{\mathrm{main}}(x,y)$ (and $f_{\mathrm{M}}(x,y)$), define a certain family of polynomials $h_1(x,y),\ldots,h_n(x,y)$ such that
$$f_{\mathrm{main}}(x,y)\equiv 0 \pmod{eM} \Rightarrow h_c(x,y)\equiv 0 \pmod{(eM)^m} \text{ for } c=1,\ldots,n.$$
(Here m and n are some algorithm parameter defined later.)
2. Set $X=\lceil N^{\beta}\rceil$ and $Y=\lceil 3N^{0.5}\rceil$. Consider vectors by $\mathbf{b}_c=\mathcal{V}_J(h_c;X,Y)$ for $c=1,\ldots,n$. Here, a sequence J is a set of appropriately ordered integer pairs (i,j) such that a monomial x^iy^j appears in $h_c(x,y)$.
3. For $\mathbf{b}_1,\ldots,\mathbf{b}_n$, compute reduced basis by a lattice basis reduction algorithm. We denote by $\mathbf{v}_1,\ldots,\mathbf{v}_n$ this reduced basis.
4. Define $g_1(x,y)$ and $g_2(x,y)$ by $g_a(x,y)=\mathcal{F}_J(\mathbf{v}_a;X,Y)$ respectively. Obtain solutions of $g_1(x,y)=g_2(x,y)=0$ numerically. Then from these solutions, compute d as given by (13).

Fig. 2. Outline of the lattice based attack

$f_{\mathrm{main}}(x,y)$ is the left-hand side of equation (12). The motivation of $f_{\mathrm{M}}(x,y)$ will be explained later; here we only point out that $f_{\mathrm{M}}(x_0,y_0)\equiv 0 \pmod{eM}$, where (x_0,y_0) is a useful solution.

Now we summarise the above explanation. Our technical goal is to obtain the useful solution (x_0,y_0) satisfying (12), in other words, a pair satisfying *both* $f_{\mathrm{main}}(x_0,y_0)\equiv 0 \pmod{eM}$, $|x_0|<N^{\beta}$ *and* $|y_0|<3N^{0.5}$. For achieving this technical goal by solving the modular equation, we make use of the Howgrave-Graham Lemma, and for this purpose, we modify $f_{\mathrm{main}}(x,y)$ to some family of functions with small XY-norm. The task of defining these polynomials is formulated as the shortest vector problem, and a known polynomial time algorithm such as the LLL algorithm is used. This is the rough sketch of the lattice based attack. The outline of our algorithm is stated as Figure 2.

Some remarks may be necessary. Note first that m and n are algorithm parameters; m is chosen appropriately and n is the number of polynomials $h_c(x,y)$ that is also determined appropriately based on m. Secondly note that we have for $|x|<X$ and $|y|<Y$ the following relation between these polynomials.

$$\begin{aligned}&f_{\mathrm{main}}(x,y)\equiv 0 \pmod{eM} \;\Rightarrow\; h_c(x,y)\equiv 0 \pmod{(eM)^m} \text{ for } c=1,\ldots,n\\ &\Rightarrow\; g_a(x,y)\equiv 0 \pmod{(eM)^m} \text{ for } a=1,2 \;\Leftrightarrow\; g_a(x,y)=0 \text{ for } a=1,2.\end{aligned} \tag{17}$$

The key point of (17) is the relation $g_a(x,y)\equiv 0 \pmod{(eM)^m} \Leftrightarrow g_a(x,y)=0$ for $a=1,2$, $|x|<X$ and $|y|<Y$. This holds when $||g_a(x,y)||_{XY}=|\mathbf{v}_a|$ is smaller than $(eM)^m/\sqrt{w}$ from the Howgrave-Graham lemma. (Here w is the number of terms of each $g_a(x,y)$.) Then we have the relation

$$f_{\mathrm{main}}(x,y)\equiv 0 \pmod{eM} \Rightarrow g_a(x,y)=0 \text{ for } a=1,2.$$

Hence we can obtain the useful solution from computing all solutions of $g_1(x,y)=g_2(x,y)=0$ by some numerical method if it exists.

By (8) and (10), we have

$$||g_1(x,y)||_{XY}=|\mathbf{v}_1|\le 2^{(n-1)/4}\det(L)^{1/n}, \text{ and}$$

$$||g_2(x,y)||_{XY} = |\mathbf{v}_2| \le 2^{n/2}\det(L)^{1/(n-1)}.$$

Since the second bound is larger, we obtain the following sufficient condition for using the Howgrave-Graham lemma:

$$2^{n/2}\det(L)^{1/(n-1)} < \frac{(eM)^m}{\sqrt{w}}. \tag{18}$$

We modify (18) to a more simple approximate bound as follows.

$$\det(L)^{1/n} < (eM)^m \tag{19}$$

Now our goal is to construct a lattice satisfying (19), and we will show in the next section that it is possible if β and δ satisfies the condition (3) given in Introduction.

4 Our Construction

In this section, we explain our construction and a main result. The largest difference between former algorithm and ours is a lattice construction satisfying (19). We will give its analysis in detail in Section 6.

Let β and δ be assumed bounds defined above, m be an algorithm parameter introduced in the above outline, τ be a parameter used to optimise the bounds by β and δ. We fix them throughout this section. We introduce index series $I_a(m,\tau,\beta,\delta)$ for constructing our lattice $L(m,\tau,\beta,\delta)$.

Definition 3. *We define our sequence $I_1(m,\tau,\beta,\delta), I_2(m,\tau,\beta,\delta)$ and $I_3(m,\tau,\beta,\delta)$ (In short, I_1, I_2 and I_3 respectively). Here we set*

$$\begin{aligned}
&I_1(m,\tau,\beta,\delta) = \{(i,j)\in\mathbb{Z}\times\mathbb{Z} | 0\le i\le m, 0\le j\le i\},\\
&I_2(m,\tau,\beta,\delta) = \{(i,j)\in\mathbb{Z}\times\mathbb{Z} | 0\le i\le m, i< j\le i+\tau m\} \text{ and}\\
&I_3(m,\tau,\beta,\delta) = \{(i,j)\in\mathbb{Z}\times\mathbb{Z} | 0\le i\le m, j\le 2(1-\beta)i\} \setminus (I_1\cup I_2).
\end{aligned}$$

We consider the order $\prec$ in I_1, I_2 and I_3 by the lexicographic order of (i,j)[4]*. Then we define the index sequence $I(m,\tau,\beta,\delta)$ by concatenating I_1, I_2 and I_3. That is, order of elements in I is defined as follows for $(i,j)\in I_k$ and $(i',j')\in I_{k'}$,*

$$(i,j)\prec(i',j') \Leftrightarrow \begin{cases} k<k' \text{ or} \\ k=k' \text{ and } (i,j)\prec(i',j') \text{ in } I_k. \end{cases}$$

We define our polynomials $f_{i,j}(x,y)$ to construct our lattice (this is $h_c(x,y)$ in the outline).

[4] Notice that a symbol $\prec$ means R.H.S. is exactly larger than L.H.S., not equal. A symbol $\preceq$ means R.H.S. is equal to or larger than L.H.S.

Definition 4

$$f_{i,j}(x,y) = \begin{cases} (eM)^{m-j}x^{i-j}(f_{\mathrm{main}}(x,y))^j & \text{for } (i,j) \in I_1 \\ (eM)^{m-i}y^{j-i}(f_{\mathrm{main}}(x,y))^i & \text{for } (i,j) \in I_2 \\ e^{m-i}y^{j-i}(f_{\mathrm{M}}(x,y))^i & \text{for } (i,j) \in I_3 \end{cases} \tag{20}$$

Then we define a sequence $J(m,\tau,\beta,\delta) = \{(i',j')|$a monomial $x^{i'}y^{j'}$ is appeared in some $f_{i,j}(x,y)\}$ where we assume the standard lexicographic order in $J(m,\tau,\beta,\delta)$. We simply denote this by J.

It is clear that $f_{i,j}(x_0,y_0) \equiv 0 \pmod{(eM)^m}$ for $(i,j) \in I$. The number of polynomials $|I|$ is just n in Figure 2. Note also that $|J| = \tilde{n}$, the number of components of each vector $\mathbf{b}_{i,j}$ is $O(|I|)$ since we can rewrite a set J by $\{(i,j) \in \mathbb{Z} \times \mathbb{Z} | 0 \le i \le m, 0 \le j \le i + (1-2\beta)m\}$. Hence $|I|$ and $|J|$ has a same order $\Theta(m^2)$.

By using these polynomials and indecies, we define our lattice $L(m,\tau,\beta,\delta)$ by

$$L(m,\tau,\beta,\delta) = L(\mathbf{b}_{i_1,j_1},\ldots,\mathbf{b}_{i_n,j_n}).$$

Here $(i_1,j_1),\ldots,(i_n,j_n)$ are the index sequence in I and $\mathbf{b}_{i_\ell,j_\ell} = \mathcal{V}_J(f_{i_\ell,j_\ell};X,Y)$, a vectorisation of $f_{i_\ell,j_\ell}(x,y)$ with parameters $X = \lceil N^\beta \rceil$ and $Y = \lceil 3N^{0.5} \rceil$. The lattice dimension and lattice component size of $L(m,\tau,\beta,\delta)$ are $|I|$ and $|J|$ respectively.

For evaluating the determinant of L, we will show

$$\begin{aligned} |\mathbf{b}^*_{i,j}| &= (eM)^{m-j}X^iY^j \text{ for } (i,j) \in I_1, \\ |\mathbf{b}^*_{i,j}| &= (eM)^{m-i}X^iY^j \text{ for } (i,j) \in I_2, \text{and} \\ e^{m-j}M^mX^iY^j \le |\mathbf{b}^*_{i,j}| &< 2e^{m-j}M^mX^iY^j \text{ for } (i,j) \in I_3. \end{aligned} \tag{21}$$

Here $\mathbf{b}^*_{i_1,j_1},\ldots,\mathbf{b}^*_{i_n,j_n}$ is a Gram-Schmidt orthogonal basis of given basis. We give the proof of these bounds in Section 6. (Lemma 3, Lemma 4 and Lemma 5). Now we assume that these bounds hold, and we introduce an "evaluator" for deciding suitable τ. The evaluator eval(i,j) for $\mathbf{b}^*_{i,j}$ is defined by

$$\mathrm{eval}(i,j) = \log_N \left(|\mathbf{b}^*_{i,j}|/(eM)^m \right). \tag{22}$$

We define the evaluator for index sequence I by

$$\mathrm{eval}(I) = \sum_{(i,j)\in I} \mathrm{eval}(i,j). \tag{23}$$

Then we can state the condition (19) in terms of eval.

Lemma 2. *For our index sequence I satisfying*

$$\mathrm{eval}(I) < 0, \tag{24}$$

the condition (19) holds.

Proof. By (9) and the definition of the evaluator, we have

$$N^{\mathrm{eval}(I)} = N^{\sum_{(i,j)\in I}\mathrm{eval}(i,j)} = \prod_{(i,j)\in I}\left(\frac{|\mathbf{b}^*_{i,j}|}{(eM)^m}\right) = \det(L)/(eM)^{|I|\cdot m}.$$

Thus, (24) is equivalent to $\det(L) < (eM)^{|I|\cdot m}$. Which is indeed the condition (19). □

Thus, we use $\mathrm{eval}(I) < 0$ as our (approximate) sufficient condition that the lattice based attack (under our construction of the lattice L) breaks a given RSA instance. Note that this condition is based on our heuristic assumption and it is only an approximate condition because of the approximation of (18) by (19); yet further approximation is used below for estimating eval. This means that the condition for parameters β and δ we will derive below is, strictly speaking, not accurate nevertheless, we will argue by using our approximation for avoiding unnecessary complications. Justification of our approximation analysis and together with our heuristic assumption will be given by computer experiments shown later.

Now by using the bound (21) (see Proposition 1 in Section 6.1), we can approximately evaluate $\mathrm{eval}(i,j)$ as follows[5].

$$\mathrm{eval}(i,j) \approx \begin{cases} i\beta - (\beta-\delta+0.5)j & (i,j)\in I_1 \\ (\delta-1)i + 0.5j & (i,j)\in I_2 \\ (-1+\beta)i + 0.5j & (i,j)\in I_3. \end{cases}$$

From this we can approximately estimate $\mathrm{eval}(I)$. From some calculation (see Section 6.2) we have

$$\mathrm{eval}(I) = \left(\frac{1}{6}(\beta+\delta-0.5) + \frac{\tau}{4}(2\delta+\tau-1) - \frac{1}{12}\frac{(1-2\beta-\tau)^3}{1-2\beta}\right)m^3 + o(m^3) \tag{25}$$

for $1-2\beta-\tau > 0$, and we have

$$\mathrm{eval}(I) = \left(\frac{1}{6}(\beta+\delta-0.5) + \frac{\tau}{4}(2\delta+\tau-1)\right)m^3 + o(m^3) \tag{26}$$

for $1-2\beta-\tau \le 0$. Note that conditions $1-2\beta-\tau > 0$ and $1-2\beta-\tau \le 0$ are corresponding to the cases $I_3 \ne \phi$ and $I_3 = \phi$ respectively. Then following the argument of [2,7], we analyze the bound for *the ideal case* by assuming that m is sufficiently large. (We draw a figure to show the bounds for some concrete values of m, see Figure 4 in Section 5.)

First we consider the case of $1-2\beta-\tau > 0$. Assuming that m is sufficiently large, the condition (24) is approximately equivalent to

$$\frac{1}{6}(\beta+\delta-0.5) + \frac{\tau}{4}(2\delta+\tau-1) - \frac{1}{12}\frac{(1-2\beta-\tau)^3}{1-2\beta} < 0, \tag{27}$$

[5] Here a symbol $A \approx B$ means $\frac{A}{B} \to 1$ when N (the RSA bit length) goes to infinity.

where its left-hand side is minimised when τ takes value $\tau_0 = \sqrt{2(1-2\beta)(\beta-\delta)}$. Hence, substituting this to (27), we have

$$\frac{1}{3}\sqrt{2(1-2\beta)(\beta-\delta)}(\delta-\beta) - \frac{1}{3}\beta^2 + \frac{1}{2}\beta + \frac{1}{6}\delta - \frac{1}{6} < 0. \tag{28}$$

This is equivalent to (3).

Next we consider the case of $1-2\beta-\tau \leq 0$. Again assuming m is sufficiently large, we have the following approximate condition (24).

$$\frac{1}{6}(\beta+\delta-0.5) + \frac{\tau}{4}(2\delta+\tau-1) < 0. \tag{29}$$

By similar argument, we substitute $\tau_1 = \frac{1-2\delta}{2}$ to τ for minimising (29), and derive the following condition (29).

$$\delta < \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta}. \tag{30}$$

This is equivalent to the condition proposed by [7]. Therefore, we can recover the secret key of RSA in polynomial time in $\log N$ when β and δ satisfies (3) or (4).

5 Computer Experiments

We carried out our *preliminary* computer experiments to check that our approach works and estimate its efficiency. We conducted our computer experiments on the TSUBAME supercomputer[6] We implemented our experiment program by the C++ language using Shoup's NTL [12] of version 5.4.2. We carried out the lattice reduction part by L^2 algorithm [13,14] with parameter $\delta = 0.99$ and $\eta = 0.51$,[7] and implemented the resultant calculation algorithm by [8]. We compiled our source code by `gcc-4.1.2` (64 bit version) with `-O6` option.

The procedure of our experiments is shown in Figure 3. The algorithm part is essentially the same as the outline in Figure 2. At Step 4, the vectors obtained by the L^2 algorithm are sorted by their length; this is because those vectors are approximate ones and we cannot guarantee that $\mathbf{v}_1$ and $\mathbf{v}_2$ are the smallest in reduced basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$. We check the algebraic independence of g_1 and g_2 by checking $R(x) \neq 0$ holds or not. More precisely, we regard the experiment is succeeded if $R(x) \neq 0$ and $R(x_0) = 0$.

Input parameters of this experiments are ℓ, m, β and δ, which are respectively the bit-size of N, the parameter for constructing lattice, the ratio of $\lg(d)$ to

[6] TSUBAME is a grid type supercomputer at Tokyo Inst. of Tech., whose performance is currently (by Top500, Nov. 2008) the 29th in the World. A node of the supercomputer which we used contains eight Opteron Dual Core model 880 processors of 2.4GHz and 32GB RAM. Note, however, we have not been able to make a parallel version of our algorithm; TSUBAME's massive parallelism has been used only for reducing the total experiment time.

[7] This δ is L^2 algorithm's parameter, not relating to the RSA secret key.

Step 1: (Make sample RSA instance) Randomly choose $\ell/2$-bit primes p and q, and let $N = pq$. (In our program, we choose p and q the Euler-Jacobi pseudoprime to bases 2, 3, 5, 7 and 11.) Randomly choose $\lfloor \beta\ell \rfloor$-bit random odd integer as the secret key d such that $\gcd(d, (p-1)(q-1)) = 1$, and let $M = 2^{\lfloor (\beta-\delta)\ell \rfloor}$ and $d_0 = d \bmod M$. Compute the public key $e \equiv d^{-1} \pmod{(p-1)(q-1)}$, and let $f_{\text{main}}(x, y) = x(N + y) + (ed_0 - 1)$, and let $y_0 = 1 - p - q$ and $x_0 = (1 - ed)/((p-1)(q-1))$.

Step 2: Let $\tau = \sqrt{2(1-2\beta)(\beta-\delta)}$, $X = \lfloor N^{\beta} \rfloor$ and $Y = \lfloor 3N^{0.5} \rfloor$. Then construct our lattice $L(m, \tau, \beta, \delta)$.

Step 3: Apply the L^2 algorithm for $L(m, \tau, \beta, \delta)$ with L^2 parameter $(\delta, \eta) = (0.99, 0.51)$.

Step 4: Sort the vectors of reduced basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ by these length to $\mathbf{v}'_1, \ldots, \mathbf{v}'_n$. Compute $g_i(x, y) = \mathcal{F}_I(\mathbf{v}'_i, X, Y)$ for $i = 1, 2$.

Step 5: Check $g_1(x_0, y_0) = 0$ and $g_2(x_0, y_0) = 0$. (If $g_1(x_0, y_0) \neq 0$ or $g_2(x_0, y_0) \neq 0$, the experiment is failure.) Compute $R(x) = \text{Res}(h_1, h_2)$ and check $R(x) \neq 0$ and $R(x_0) = 0$ holds or not.

Fig. 3. Our computer experiment procedure

$\lg(N)$, and the ratio of $\lg(\tilde{d})$ to $\lg(N)$. (See Figure 2 for the parameters m and n, and see Section 3 for the parameters β and δ.) A word "dim." means the lattice dimension in experiments, i.e., $n = |I|$ in our construction. By "total time"and "L^2 time" we mean the CPU time of Step 2 through Step 4 and that of Step 3 respectively. Recall that the lattice component size is bounded by a constant time of the lattice dimension. Here, we disregard the time for computing the resultant of g_1 and g_2. Results are in Table 1 and Table 2. Table 1 shows the qualities of our lattices, that is, whether the experiment is succeeded or not, for these parameters. On the other hand, Table 2 shows the computational time of our experiments for various ℓ and m, and fixed β and δ.

Quality of Lattice

For checking our approach indeed works and estimating the quality of our lattice construction, we carried out our preliminary computer experiments.

Note first that the bounds (3) and (4) are the ideal ones obtained by the asymptotic analysis assuming m is sufficiently large. For each given value of m, we can determine the range of β and δ satisfying $\text{eval}(I(m, \tau_0, \beta, \delta)) < 0$ by numerically analyzing the original expressions (i.e., (34) $\sim$ (36) of Appendix A.2). Figure 4 shows the bounds of β and δ obtained in this way for some m values, and the theoretical limit of our construction (3) and that of [7]'s construction (2). It can be seen that these bounds get close to our ideal bound when m gets large. We focus on the range of $0.28 \leq \beta \leq 0.32$ and $0 \leq \text{ratio} \leq 0.2$, mainly our new improvement area.

Our computer experiments are summarized in Figure 5, which are for the cases $m = 10$ and 14. The instance size ℓ $(= \lg(N))$ is 1024 for both cases; since these are still preliminary ones, we conducted only one execution for each instance. Note that a shade area in each figure is the area that $\text{eval}(I(m, \tau_0, \beta, \delta)) < 0$

obtained numerically for each m. A black circle (resp., a white circle) indicates the parameter (β, δ) (or the point $(\beta, (\beta - \delta)/\beta)$) that the experiment succeeds (resp., fails). Those results are shown in detail in Table 1. For $m = 10$, the word "yes" in the column "success" in the tables means $R(x) \neq 0$ and $R(x) = 0$ at the Step 5. On the other hand, "yes" for $m = 14$ means $g_1(x_0, y_0) = g_2(x_0, y_0) = 0$ at the Step 5. The difference is because of the resultant calculation is too heavy operation for our C++ implementation. We expect this problem will be solved by the technique of Glöbner basis.[8]

The word "no(1)" in the column "success" means $g_1(x_0, y_0) = 0$ whereas $g_2(x_0, y_0) \neq 0$ at the Step 5. In this case, we may expect to get the correct solution by generating enough number of polynomials g_i by changing X and Y randomly within the same bit length. On the other hand, the word "no(0)" means that $g_1(x_0, y_0) \neq 0$ and $g_2(x_0, y_0) \neq 0$, or $R(x_0) \neq 0$ at Step 5, which we regarded as a failure.

Computational Time

Next we examine the efficiency of our algorithm based on our experiments. As seen by our analysis and experiments, the algorithm shows better performance by using large m. On the other hand, by using larger m, the lattice dimension also get larger. More specifically, the lattice dimension is $\Theta(m^2)$ by our construction. We examine how this indeed effects to the running time of the algorithm.

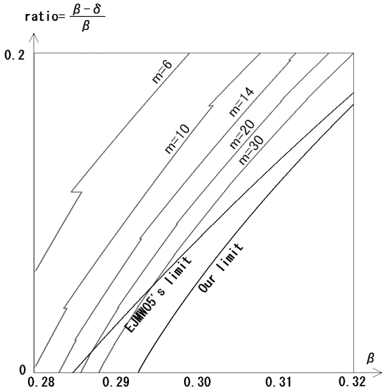


Fig. 4. Our recoverable range for many m

We carried out computer experiments with parameters $(\beta, \delta) = (0.3, 0.225)$ and $(\beta, \delta) = (0.4, 0.12)$, whereas parameters m and ℓ are chosen as $m = 6$, 8, 10 and 12, and $\ell = 512$, 1024, and 2048. Table 2 is experiments for these parameters. Total time and L^2 time in these tables are the average running time of five executions. Notice that we checked the algebraic independence of $g_1(x, y)$ and $g_2(x, y)$, and compute the resultant $R(x)$ for $m = 6$, 8 and 10. All of them are

[8] We carried out the check of algebraic independence because referee's comment suggested it. However some of these checks are not completed at the deadline.

success. However we did not check the algebraic independence for $m = 12$, only checked $g_1(x_0, y_0) = g_2(x_0, y_0) = 0$. At this meaning, experiments for $m = 12$ are success.

These results show the total time and L^2 time are close, which shows a lattice reduction algorithm is the main part of the lattice based attack. From these tables, we obtain our L^2 time is approximately

$$\text{Time} \approx 0.35 \left(\frac{\ell}{512}\right)^2 \cdot \left(\frac{m}{2}\right)^8 \cdot \log_2 \ell \cdot \log_2 m \text{ sec for } \beta{=}0.3 \text{ and } \delta{=}0.225, \text{ and}$$

$$\text{Time} \approx 0.6 \left(\frac{\ell}{512}\right)^2 \cdot \left(\frac{m}{2}\right)^8 \cdot \log_2 \ell \cdot \log_2 m \text{ sec for } \beta = 0.4 \text{ and } \delta = 0.12.$$

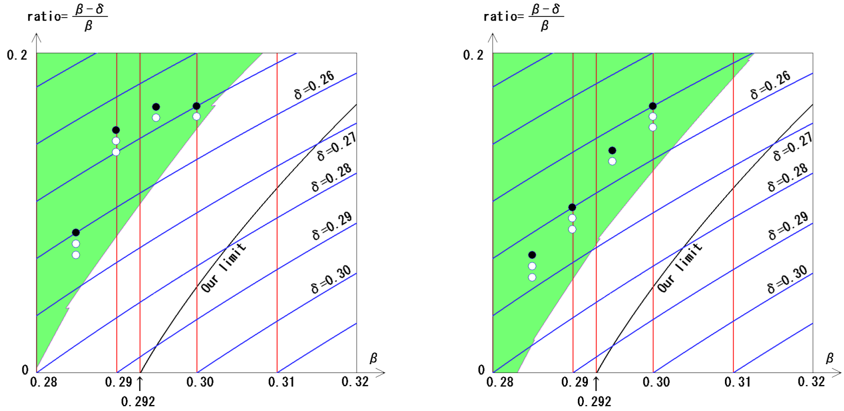
6 Analysis

This section provides the precise analysis for some results in Section 4. Our objective is to derive our theoretical recoverable limit of our construction (28) and (30).

In order to obtain these inequalities, we first prove the approximated estimation

$$\text{eval}(i,j) \approx \begin{cases} \beta i - (\beta - \delta + 0.5)j & (i,j) \in I_1 \\ (\delta - 1)i + 0.5j & (i,j) \in I_2 \\ (-1+\beta)i + 0.5j & (i,j) \in I_3 \end{cases} \tag{31}$$

in Section 6.1. Next in Section 6.2 we explain the way to derive our theoretical limit. We fix algorithm parameters and RSA instances throughout this section, and denote $f^*_{i,j}(x,y)$ the functionalisation of $\mathbf{b}^*_{i,j}$. (We recall that $\{\mathbf{b}^*_{i,j}\}_{(i,j)\in I}$ is the Gram-Schmidt orthogonal basis of our basis $\{\mathbf{b}_{i,j}\}_{(i,j)\in I} \overset{\text{def}}{=} \{\mathcal{V}(f_{i,j}; X, Y)\}_{(i,j)\in I}$.)



$m = 10$ $m = 14$

Fig. 5. Summary of our experiments for $\ell = 1024$

6.1 Approximating the Evaluator

We first compute the exact value of $|\mathbf{b}^*_{i,j}|$ for $(i,j) \in I_1$ and I_2 in Lemma 3 and Lemma 4 respectively. Next we consider the bound for $|\mathbf{b}^*_{i,j}|$ for $(i,j) \in I_3$ in Lemma 5. Then we estimate the approximated value of $\mathrm{eval}(i,j) \overset{\mathrm{def}}{=} \log_N(|\mathbf{b}^*_{i,j}|/(eM)^m)$ in Proposition 1.

The following lemmas are immediately derived by considering the diagonal part of the matrix representation of L. Hence we omit the proofs of them.

Lemma 3. *For any* $(i,j) \in I_1$, *we have*

$$|\mathbf{b}^*_{i,j}| = (eM)^{m-j} X^i Y^j. \tag{32}$$

Lemma 4. *For any* $(i,j) \in I_2$, *we have*

$$|\mathbf{b}^*_{i,j}| = (eM)^{m-i} X^i Y^j. \tag{33}$$

On the other hand, the proof of the following lemma is a bit involved. See the full version of this paper for the proofs of Lemma 3 to Lemma 5.

Lemma 5. *For any* $(i,j) \in I_3$, *we have*

$$e^{m-i} M^m X^i Y^j \leq |\mathbf{b}^*_{i,j}| < 2e^{m-i} M^m X^i Y^j.$$

By using these lemmas, we obtain the approximated bounds for evaluators for each (i,j).

Proposition 1

$$\mathrm{eval}(i,j) \approx \begin{cases} \beta i - (\beta - \delta + 0.5)j & (i,j) \in I_1 \\ (\delta - 1)i + 0.5j & (i,j) \in I_2 \\ (-1+\beta)i + 0.5j & (i,j) \in I_3 \end{cases}$$

Table 1. Quality of our lattice for $m = 10$ and $m = 14$

Experiment parameters			Results	
β	δ	dim.	L^2 time	Success
0.285	0.260	88	3 h 42 m	yes
0.285	0.262	88	3 h 46 m	no(1)
0.285	0.264	88	3 h 53 m	no(0)
0.290	0.246	87	3 h 15 m	yes
0.290	0.248	87	3 h 24 m	no(1)
0.290	0.250	87	3 h 1 m	no(0)
0.295	0.246	92	4 h 2 m	yes
0.295	0.248	87	2 h 53 m	no(0)
0.300	0.250	91	3 h 25 m	yes
0.300	0.252	85	2 h 16 m	no(0)

$m = 10$

Experiment parameters			Results	
β	δ	dim.	L^2 time	Success
0.285	0.264	162	2 d 20 h	yes
0.285	0.266	162	2 d 18 h	no(1)
0.285	0.268	162	2 d 9 h	no(0)
0.290	0.260	165	2 d 22 h	yes
0.290	0.262	165	2 d 16 h	no(1)
0.290	0.264	165	2 d 16 h	no(0)
0.295	0.254	164	2 d 19 h	yes
0.295	0.256	164	2 d 17 h	no(0)
0.300	0.250	163	3 d 7 h	yes
0.300	0.252	163	3 d 14 h	no(1)
0.300	0.254	163	3 d 11 h	no(0)

$m = 14$

Proof. As we explained at the overview, we assumed that

$$X \approx N^{\beta},\ Y \approx N^{0.5},\ \text{and } e \approx N,$$

from which by using (6) we derive $M = \frac{d-d_0}{\tilde{d}} \approx \frac{N^{\beta}}{N^{\delta}} = N^{\beta-\delta}$.

Substituting these for each bound of $\mathbf{b}^*_{i,j}$, we have by using $\text{eval}(i,j) \overset{\text{def}}{=} \log_N \left(\frac{|\mathbf{b}^*_{i,j}|}{(eM)^m}\right)$,

$$\begin{array}{ll} \text{eval}(i,j) = \log_N (eM)^{-j} X^i Y^j \approx \beta i - (\beta - \delta + 0.5)j & \text{for } (i,j) \in I_1, \\ \text{eval}(i,j) = \log_N (eM)^{-i} X^i Y^j \approx (\delta - 1)i + 0.5j & \text{for } (i,j) \in I_2, \text{ and} \\ \text{eval}(i,j) \approx \log_N e^{-j} X^i Y^j \approx (-1+\beta)i + 0.5j & \text{for } (i,j) \in I_3. \end{array} \qquad \square$$

6.2 Some Calculations on eval(I)

This section shows how to get our conditions (28) and (30) in detail.

First we state the expressions for $\text{eval}(I_1)$, $\text{eval}(I_2)$, and $\text{eval}(I_3)$ derived from (23) and Proposition 1.

$$\text{eval}(I_1) = \sum_{i=0}^{m}\sum_{j=0}^{i} \text{eval}(i,j) = \sum_{i=0}^{m}\sum_{j=0}^{i} (i\beta - (\beta - \delta + 0.5)j), \tag{34}$$

$$\text{eval}(I_2) = \sum_{i=0}^{m}\sum_{j=i+1}^{i+\lfloor \tau m \rfloor} \text{eval}(i,j) = \sum_{i=0}^{m}\sum_{j=i+1}^{i+\lfloor \tau m \rfloor} ((\delta - 1)i + 0.5j), \text{ and} \tag{35}$$

$$\text{eval}(I_3) = \sum_{i=\lfloor Bm \rfloor}^{m}\sum_{j=i+\lceil \tau m \rceil}^{\lfloor 2(1-\beta)i \rfloor} \text{eval}(i,j) = \sum_{i=\lfloor Bm \rfloor}^{m}\sum_{j=i+\lceil \tau m \rceil}^{\lfloor 2(1-\beta)i \rfloor} ((-1+\beta) + 0.5j). \tag{36}$$

Here we use B to denote $\frac{\tau}{1-2\beta}$. Figure 6 shows an area of I_1, I_2 and I_3. By definition, these are sets of integral points in each I_k. The cross point of the line $j = 2(1-\beta)$ and $j = i + \tau m$, i.e., $(\frac{\tau}{1-2\beta}m, \frac{2(1-\beta)\tau}{1-2\beta}m)$, exists below the vertex point $(m, 2(1-\beta)m)$ when $1 - 2\beta - \tau > 0$. Thus, we separate our discussion into the case of $1 - 2\beta - \tau > 0$ and that of $1 - 2\beta - \tau \leq 0$.

First we consider the case of $1 - 2\beta - \tau > 0$. This case corresponds $I_3 \neq \phi$ and the following holds.

$$\text{eval}(I_1) = \sum_{i=0}^{m}\sum_{j=0}^{i} \text{eval}(i,j) = \frac{1}{6}(\beta + \delta - 0.5)m^3 + o(m^3),$$

$$\text{eval}(I_2) = \sum_{i=0}^{m}\sum_{j=i+1}^{i+\lfloor \tau m \rfloor} \text{eval}(i,j) = \frac{\tau}{4}(2\delta + \tau - 1)m^3 + o(m^3),$$

$$\text{eval}(I_3) = \sum_{i=\lfloor Bm \rfloor}^{m}\sum_{j=i+\lceil \tau m \rceil}^{\lfloor 2(1-\beta)i \rfloor} \text{eval}(i,j) = -\frac{1}{12}\frac{(1-2\beta-\tau)^3}{1-2\beta}m^3 + o(m^3).$$

Table 2. Total time for ℓ=512, 1024 and 2048

Experiment parameters				Results	
β	δ	m	deg.	L^{2}time	Total time
0.3	0.225	6	36	1 m 6 s	1 m 8 s
		8	58	10 m 31 s	10 m 42 s
		10	91	1 h 25 m	1 h 26 m
		12	124	6 h 13 m	6 h 15 m
		14	171	25 h 28 m	25 h 35 m
0.4	0.12	6	35	2 m 4 s	2 m 7 s
		8	54	16 m 32 s	16 m 46 s
		10	77	1 h 10 m	1 h 11 m
		12	117	10 h 11 m	10 h 14 m
		14	150	29 h 56 m	30 h 3 m

$\ell = 512$

Experiment parameters				Results	
β	δ	m	deg.	L^{2}time	Total time
0.3	0.225	6	36	4 m 34 s	4 m 42 s
		8	58	40 m 5 s	40 m 44 s
		10	91	5 h 33 m	5 h 36 m
		12	124	21 h 25 m	21 h 33 m
		14	171	127 h 0 m	127 h 10 m
0.4	0.12	6	35	8 m 41 s	8 m 52 s
		8	54	64 m 22 s	65 m 10 s
		10	77	4 h 56 m	4 h 59 m
		12	117	43 h 35 m	43 h 45 m
		14	150	159 h 40 m	160 h 4 m

$\ell = 1024$

Experiment parameters				Results	
β	δ	m	deg.	L^{2}time	Total time
0.3	0.225	6	36	21 m 34 s	22 m 3 s
		8	58	2 h 46 m	2 h 48 m
		10	91	24 h 8 m	24 h 17 m
0.4	0.12	6	35	42 m 34 s	43 m 13 s
		8	54	4 h 37 m	4 h 40 m
		10	77	21 h 19 m	21 h 29 m

$\ell = 2048$

Our implementation code is not completely optimized. These are for comparing the ratio of the L^2 times to the total times. The times in these tables are the average running time of five executions. The resultant check was carried out for $m = 6$, 8 and 10.

From these we derive the equation (25) for $\text{eval}(I) \overset{\text{def}}{=} \text{eval}(I_1) + \text{eval}(I_2) + \text{eval}(I_3)$.

We next consider the case of $1 - 2\beta - \tau \leq 0$. This is a case of $I_3 = \phi$, thus by $\text{eval}(I) = \text{eval}(I_1) + \text{eval}(I_2)$ we obtain (26).

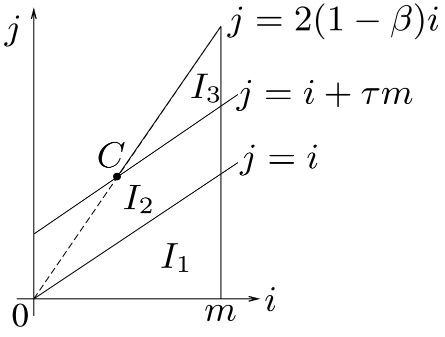


Fig. 6. I_1, I_2 and I_3

7 Conclusion

We gave the new lattice construction for the lattice based attack for the RSA cryptography in the situation that d is small and LSBs of d is exposed. By this construction, the theoretical recoverable range has been improved as shown in Figure 1, which solves the open problem raised in [7]. Also as shown by our preliminary experimental results, the total efficiency of the lattice based attack has been improved significantly compared with [7]. Some more improvement, however, is necessary for using this technique with large m. One possibility is to make use of parallelization for processing the lattice basis reduction algorithm. From a theoretical view point, it would be interesting if we can understand the limitation of this approach.

Acknowledgement

I am grateful to Osamu Watanabe for his advice, careful reading, and correct some expressions. The author was supported by the Global CompView Project. This research was supported in part by JSPS Global COE program "Computationism as a Foundation for the Sciences".

References

1. Aono, Y.: Degree reduction of the lattice based attack for RSA. In: COMP2007-5, vol.107, no. 24(20070419), pp. 33–40 (2007) (in Japanese)
2. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$. IEEE Transactions on Information Theory 46(4), 1339–1349 (2000)
3. Boneh, D., Durfee, G., Frankel, Y.: An attack on RSA given a small fraction of the private key bits. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 25–34. Springer, Heidelberg (1998)
4. Blömer, J., May, A.: New partial exposure attacks on RSA. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 27–43. Springer, Heidelberg (2003)
5. Coppersmith, D.: Finding a small root of a univariate modular equation. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 178–189. Springer, Heidelberg (1996)
6. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology 10(4), 233–260 (1997)
7. Ernst, M., Jochemsz, E., May, A., Weger, B.: Partial key exposure attacks on RSA up to full size exponents. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 371–386. Springer, Heidelberg (2005)
8. Healy, A.D.: Resultants, Resolvents and the Computation of Galois Groups, `http://www.alexhealy.net/papers/math250a.pdf`
9. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
10. Jochemz, E., May, A.: A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (2006)

11. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (1982)
12. Shoup, V.: NTL: A Library for doing Number Theory, `http://www.shoup.net/ntl/index.html`
13. Nguyen, P., Stehlé, D.: Floating-Point LLL revisited. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)
14. Nguyen, P., Stehlé, D.: Floating-Point LLL (Full version), `ftp://ftp.di.ens.fr/pub/users/pnguyen/FullL2.pdf`
15. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptsystems. Communications of the ACM 21(2), 120–128 (1978)
16. Schnorr, C.P.: A more efficient algorithm for lattice basis reduction. Journal of algorithms 9(1), 47–62 (1988)
17. Wiener, M.J.: Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory 36(3), 553–558 (1990)

Subset-Restricted Random Walks for Pollard rho Method on $\mathbf{F}_{p^m}$ [⋆]

Minkyu Kim, Jung Hee Cheon, and Jin Hong

ISaC and Department of Mathematical Sciences
Seoul National University, Seoul 151-747, Korea
{minkyu97,jhcheon,jinhong}@snu.ac.kr

Abstract. In this paper, we propose a variant of the Pollard rho method. We use an iterating function whose image size is much smaller than its domain and hence reaches a collision faster than the original iterating function. We also explicitly show how this general method can be applied to multiplicative subgroups of finite fields with large extension degree.

The construction for finite fields uses a distinctive feature of the normal basis representation, namely, that the p-th power of an element is just the cyclic shift of its normal basis representation, when the underlying field is of characteristic p. This makes our method appropriate for hardware implementations. On multiplicative subgroups of $\mathbf{F}_{p^m}$, our method shows time complexity advantage over the original Pollard rho method by a factor of approximately $\frac{3p-3}{4p-3}\sqrt{m}$.

Through the MOV reduction, our method can be applied to pairing-based cryptosystems over binary or ternary fields. Hence our algorithm suggests that the order of subgroups, on which the pairing-based cryptosystems rely, needs to be increased by a factor of approximately m.

Keywords: discrete logarithm problem, pairing, Pollard rho method, normal basis.

1 Introduction

Let G be a finite cyclic group of order q generated by g. Given $h \in G$, the discrete logarithm problem (DLP) over G is to find the smallest non-negative integer x satisfying $g^x = h$. The integer x, denoted by $\log_g h$, is called the discrete logarithm of h to the base g. Given g and x, exponentiation $g^x(= h)$ can be efficiently computed through the Square-and-Multiply method, but recovering x from $h = g^x$ is considered to be difficult in many cases. Many cryptographic algorithms have been proposed under the assumption that the DLP is hard to solve under a specific presentation of the cyclic group.

The cyclic groups most widely used with cryptosystems are the multiplicative subgroups of finite fields and the subgroups of elliptic curves defined over finite fields. In practical use of finite fields, prime fields have drawn more attention, than other fields, because prime fields are easier to implement on general purpose

[⋆] A part of this paper was made public through [1].

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 54–67, 2009.

hardware and since the index calculus attack [2] is faster on binary fields than on prime fields [6]. In contrast, with pairing-based cryptography, binary and ternary fields are widely used as the base field over which Tate or Weil pairings are defined, due to the size of their embedding degrees [3,4,11]. Note that the base problems on which pairing-based cryptosystems are built can be transformed into the DLP on binary or ternary fields through the MOV reduction [13,10]. This increases the significance of the study of DLP on extension fields.

In this paper, we propose an algorithm to solve the DLP in large order extension fields. The normal basis representation is used, and this makes our algorithm suitable in hardware implementations. Our algorithm exploits the distinctive feature of the normal basis representation, namely, that the p-th power of an element is just the cyclic shift of its normal basis representation, where p is the characteristic of the underlying field. This feature was used in [8,12,27] to speed up the Pollard rho method on elliptic or hyperelliptic curves with efficient automorphisms. Our main contribution is in giving a precise complexity analysis for the case when such feature is used to solve the DLP over finite fields of large extension degree.

The Pollard rho method on a cyclic group G traces a random walk over G by iteratively applying a function to G. The discrete logarithm of the target can be computed when a collision is reached in the random walk. When the iterating function is assumed to be random, a collision is expected to occur after about $\sqrt{|G|}$ applications of the iterating function, due to the birthday paradox. Thus the complexity of Pollard rho method is determined by the size of the base group G on which random walk is performed and the running time of the iterating function. Reducing either one of these two factors will result in speedup of the Pollard rho method.

The recent work [7] proposes an improvement of the Pollard rho method on prime fields by reducing the average time taken for computing each application of the iteration function. In contrast, our method restricts the random walk to a subset S of G so that a collision occurs after about $\sqrt{|S|}$ applications of the iterating function, which is smaller than $\sqrt{|G|}$. The main difficulty in doing this was finding a fast iterating function with an image set S that was much smaller than G. We make an explicit construction for subgroups of $\mathbf{F}_{p^m}^{\times}$, with the small prime and large extension degree case as our main target. Our construction yields speedup over the normal Pollard rho by a factor of $\frac{3p-3}{4p-3}\sqrt{m}$.

We also show how our method can be applied to a pairing-based cryptosystem, in which a bilinear map $e : G_1 \times G_2 \to G_T$ is used. When a Tate or Weil pairing is in use, G_1 is a subgroup of points on an elliptic curve $E(\mathbf{F}_{p^\ell})$ and G_T is a cyclic subgroup of $\mathbf{F}_{p^{k\ell}}^{\times}$, where k is the embedding degree. The MOV attack [13] transforms the DLP on G_1 or G_2 into a DLP on G_T. When our algorithm is applied to G_T, which is a subgroup of $\mathbf{F}_{p^{k\ell}}^{\times}$, the complexity of the DLP on G_T is reduced by a factor of $\frac{3p-3}{4p-3}\sqrt{k\ell}$. There are many cryptosystems [4,11] that suggest the use of bilinear maps over binary or ternary fields, for computational and communication efficiency reasons. To be more explicit, $k\ell$ is set to 1132 and 726 in the binary and ternary curves suggested in [4,11]. On these parameters, our method would give Pollard rho speedup by factors of 20.2 or 18.0, respectively.

We remark that, while we have achieved complexity lower than the straightforward application of Pollard rho, this does not conflict with the complexity lower bound known [18] for generic algorithms solving DLPs, as our method utilizes the *encoding* information for the group. When our construction is forcefully placed within the generic algorithm framework, the gain we achieve can be explained from the fact that some group operations are much easier than others, which is a property that is ignored in the generic algorithm framework. In fact, the result on generic algorithm complexity implies that, to achieve results better than normal Pollard rho, we should either exploit the group encoding or invent a method that mostly uses the fastest of the group operations.

Organization. In Section 2, we introduce the Pollard rho method and its variants. In Section 3, we present a variant of Pollard rho, called the random walk restriction. In Section 4 and 5, we apply this method to finite fields of large extension degree and pairing-friendly elliptic curves over binary or ternary fields, respectively. Section 6 concludes this paper.

2 Pollard rho Algorithm

Throughout this paper $G = \langle g \rangle$ will be a finite cyclic group of prime order q, and $h = g^x$ will be the target we wish to find the discrete logarithm of. In this section, we briefly review some variants of the Pollard rho method.

Let us start by explaining the parts that are common to all the variants. Suppose we know $(a, b), (c, d) \in \mathbf{Z}_q \times \mathbf{Z}_q$, such that $g^a h^b = g^c h^d$ and $b \neq d$. Such a pair implies the linear equation $a + b \cdot x \equiv c + d \cdot x \pmod{q}$ and we can easily solve for x from this equation. All variants of the Pollard rho method suggest how to efficiently obtain such a double expression for a single element of G. The general strategy for obtaining the double expression is explained next.

We say that a function $f : G \to G$ is *exponent traceable*, or allows exponent tracing, with respect to g and h, if it is possible to express the function in the form

$$f(g^a h^b) = g^{f_g(a,b)} h^{f_h(a,b)},$$

for some easily computable functions $f_g, f_h : \mathbf{Z}_q \times \mathbf{Z}_q \to \mathbf{Z}_q$. Let us fix an exponent traceable function f and take a random $(a_0, b_0) \in \mathbf{Z}_q \times \mathbf{Z}_q$. We generate a sequence $(g_i)_{i \geq 0}$ through iterative applications of f, i.e., we set

$$g_0 = g^{a_0} h^{b_0} \quad \text{and} \quad g_{i+1} = f(g_i) \quad \text{for } i \geq 0.$$

Since G is finite, there must be integers $\mu \geq 0$ and $\lambda > 0$ satisfying $g_{\lambda+\mu} = g_\lambda$. The smallest of such integers are called the pre-period and period of the sequence $(g_i)_{i \geq 0}$, respectively. The exponent traceable property of f implies that we have access to the exponents (a_i, b_i) for each $g_i = g^{a_i} h^{b_i}$. Thus, an appropriate *collision* of the generated sequence, i.e., the event of $g_i = g_j$ with

$b_i \neq b_j$, allows us to compute the discrete logarithm of h, through the method described previously.

Below, we shall describe two exponent traceable iterating functions and some methods for detecting collisions. Any combination of the two components will produce a version of the Pollard rho algorithm.

2.1 Iterating Functions

Because the value $\lambda + \mu$, called *rho length*, is expected to be $\sqrt{\pi q/2}$ when the function f is chosen uniformly at random from the set of all functions on G, an iterating function is considered to be of good design if its rho length is close to $\sqrt{\pi q/2}$.

There are two main types of iterating functions. One is the original *Pollard's iterating function* and the other is called *r-adding walks*. They are defined as follows. Let $G = T_0 \cup T_1 \cup T_2$ be a partition of G consisting of roughly equal sized subsets. The Pollard's iterating function f_P [15] is

$$f_P(y) = \begin{cases} gy, & \text{if } y \in T_1, \\ y^2, & \text{if } y \in T_2, \\ hy, & \text{if } y \in T_3. \end{cases}$$

Let r be a small positive integer and let $T_0 \cup \cdots \cup T_{r-1}$ be a partition of G into roughly the same sized subsets. For each $s = 0, \ldots, r-1$, set the multipliers $M_s = g^{m_s} h^{n_s}$ with randomly selected integers m_s, n_s. With the indexing function $s : G \rightarrow \{0, 1, ..., r-1\}$, that specifies to which $T_{s(y)}$ a given element $y \in G$ belongs to, the r-adding walk f_T is defined to be

$$f_T(y) = y \cdot M_{s(y)}.$$

The work [17] shows that the expected rho length for random walk sequences generated by an r-adding walk using any $r \geq 8$ is roughly of the order $\mathcal{O}(\sqrt{q})$ on any cyclic group. Number of tests [25] over elliptic curves have shown that the rho length of Pollard's original iterating function is larger than $\sqrt{\pi q/2}$, but that of 20-adding walks is very close to $\sqrt{\pi q/2}$.

2.2 Collision Detection

The most naive approach to finding collisions is to store all generated points g_i until a collision occurs. The main issues with collision detection is to do this with minimal number of iterating function applications after the actual collision and with a small amount of memory. Recall that μ and λ denote the pre-period and period, respectively. Among the many collision detection methods proposed [9,5,16,23,14], we briefly explain those suggested by Floyd, Brent, and Quisquater-Delescaille.

Floyd. The central idea is to wait for a collision of type $g_i = g_{2i}$ to happen. Three applications of the iterating function are needed at each iteration to update the two current states g_i and g_{2i} to g_{i+1} and $g_{2(i+1)}$, respectively. This will reach a collision within $\lambda + \mu$ iterations, or equivalently, $3(\lambda + \mu)$ applications of the iterating function. So, with a good iterating function, $3\sqrt{\pi q/2}$ applications are expected.

Brent. We explain a method by Teske [24], which is an optimized version of the works [19,5]. Eight most recent g_k, for which the index k are powers of 3, are kept in storage. After each application of the iterating function, the current g_i is compared with the eight stored entries, and g_i replaces the oldest of the eight entries if a new power of 3 has been reached. This will terminate with a collision between current g_i and some g_k from storage in $\{1.25 \cdot \max(\lambda/2, \mu) + \lambda\}$ iterations. When the iterating function is random, this is expected to be $1.229\sqrt{\pi q/2}$ iterating function applications.

Distinguished points. [16] This was originally an idea for use with time-memory tradeoff techniques. Distinguished points are those elements of G that satisfy a certain condition, which is easy to check. For example, with a fixed encoding for G, we may set them to be those elements with a certain number of starting bits equal to zero.

After each application of the iterating function, the current g_i is added to a table, if and only if it is a distinguished point. The algorithm terminates when a collision is found among the stored distinguished points. The distinguished point should be defined so that this table is of manageable size.

Let θ be the fraction of elements in G which satisfy the distinguishing property. The algorithm is expected to terminate with a collision after $\sqrt{\pi q/2} + 1/\theta$ applications of the iterating function. This method has the advantage that it can lead to n-times speedup with n-processor parallelization [26].

3 Random Walk Restriction

In this section, we describe a general approach, previously applied to subgroups of elliptic curves [8,12,27], that results in faster DLP solving than straightforward applications of the Pollard rho variants. Recall that the complexity of Pollard rho variants is $\sqrt{\pi|G|/2}$ evaluations of iterating function when the cost of collision detection is ignored. This shows that two factors influence the complexity of Pollard rho variants. One is the complexity of iterating function evaluation and the other is the size of space on which the random walk is traversed. There are corresponding two approaches to the speedup of Pollard rho. One is the reduction of complexity for the evaluation of the iterating function and the other is the restriction of the random walks to a subset of G. The former was studied in [7] and the latter is the approach taken by this paper. After explaining the general method, which we shall call random walk restriction, its application to more concrete settings shall be explored in the next section.

3.1 Solving DLP with an Iterating Function of Small Image Size

Fix any function $\varphi : G \to G$ which allows exponent tracing, but which may not be useful for solving DLP, i.e., the collisions generated are of the form $g^a h^b = g^c h^d$ with $b = d$. Let $f : G \to G$ be any exponent traceable iterating function suitable for solving DLP on G and consider the function $f_\varphi := \varphi \circ f$. It is clear that f_φ maps $\bar{G}$ to $\bar{G}$, where $\bar{G} = \mathrm{Im}(\varphi)$ is a notation we shall use throughout this section. Since f is always chosen to be exponent traceable, f_φ also satisfies the same condition. Notice that during the random walking of the Pollard rho method variants, we do not need G to be closed under the group operation. Thus, we will have no problem using f_φ as an iterating function on $\bar{G}$, which is a smaller set and which should bring earlier collision.

It remains to see if f_φ will show properties close to a random function. We have the following proposition to support this when φ is uniform in its number of pre-images.

Proposition 1. *Let $\varphi : G \to G$ be a function such that $|\varphi^{-1}(y)| = |G|/|\bar{G}|$ for all $y \in \bar{G} = \mathrm{Im}(\varphi)$. If f is selected uniformly at random from G^G, the set of all functions on G, then $f_\varphi = \varphi \circ f$ is a random function on $\bar{G}$.*

Proof. Consider the function $\tilde{\varphi} : G^G \to \bar{G}^{\bar{G}}$ mapping $f \mapsto (f_\varphi = \varphi \circ f)$. It suffices to show that $\tilde{\varphi}$ is uniform in its number of pre-images, so that each function on $\bar{G}$ is equally likely to be chosen, when each function f on G is equally likely to be chosen.

Let $\mathbf{y} = (y_j)_{j\in\bar{G}} \in \bar{G}^{\bar{G}}$ be any function on $\bar{G}$. We are viewing $\mathbf{y}$ as the function on $\bar{G}$ sending $j \mapsto y_j$. Suppose that $\mathbf{x} = (x_i)_{i\in G} \in G^G$ satisfies $\tilde{\varphi}(\mathbf{x}) = \mathbf{y}$. Then we must have $\varphi(x_j) = y_j$ for each $j \in \bar{G} \subset G$, but the equation $\tilde{\varphi}(\mathbf{x}) = \mathbf{y}$ places no restriction on x_i for $i \in G \setminus \bar{G}$. In fact, we have $\tilde{\varphi}(\mathbf{x}) = \mathbf{y}$ if and only if $\varphi(x_j) = y_j$ for each $j \in \bar{G}$. Since for each $j \in \bar{G}$, x_j may be any one of the $|G|/|\bar{G}|$-many elements of G, we have

$$|\tilde{\varphi}^{-1}(\mathbf{y})| = \left(\frac{|G|}{|\bar{G}|}\right)^{|\bar{G}|} \cdot |G|^{|G\setminus\bar{G}|},$$

which is evidently uniform over all $\mathbf{y} \in \bar{G}^{\bar{G}}$. □

Even though this proposition allows us to intuitively believe that f_φ on $\bar{G}$ is as good a random function on $\bar{G}$ as f is on G, as soon as we fix an iterating function f, the induced function f_φ on $\bar{G}$ is an explicit function, so the above proposition is no logical guarantee that f_φ will provide a good random walk. But this was the situation with the original f to begin with.

For a function f, let us denote its expected time for evaluation by $|f|$. The discussion so far allows us to state the main result of this section.

Theorem 1. *Let $\varphi : G \to G$ be an exponent traceable function. Then, given an exponent traceable iterating function f suitable for solving DLP on G and a collision detection method, by working with $f_\varphi = \varphi \circ f$ over $\bar{G} = \mathrm{Im}(\varphi)$, the expected time to solve the discrete logarithm problem over G can be reduced by a factor of $\sqrt{\frac{|\bar{G}|}{|G|}} \cdot \frac{|f_\varphi|}{|f|}$, compared to that of the original method over G.*

Notice that our result is mostly independent of which variant of Pollard rho algorithm we use. Use of the distinguished point method for collision detection is an exception, if one is to be very exact, but even this case can be dealt with by appropriately increasing its probability of appearance. As the expected rho length has decreased, this should not cause trouble with distinguished point storage requirements.

Remark 1. What can we expect of the random walk provided by f_φ, when φ is not uniform in its number of pre-images? This situation implies that during the walk provided by f_φ, some elements of $\bar{G}$ are more likely to occur than others. So we should arrive at a collision even earlier than expected of a random function. Such unevenness works to the DLP solver's advantage.

Example 1. Let $\omega \in \mathbf{Z}_q$ be of order t. Suppose φ is defined in such a way that it satisfies

$$\varphi(y) = \varphi(y^{\omega^i}), \qquad i = 0, 1, \ldots, t-1,$$

in addition to being exponent traceable. Then the image of such a φ would be at most $\frac{1}{t}$-th of the total space. One possible candidate for such a function is to define $\varphi(y)$ as the minimum of $y, \ldots, y^{\omega^{t-1}}$ under an appropriate ordering. In general, such a φ would have high evaluation complexity, i.e., t exponentiations and comparisons, but in a field of prime characteristic, φ can have low complexity. In Section 4, we follow this lines of reasoning.

It should be clear by now that we do not have to define f_φ as a composition of two functions. The essential idea is to design an iterating function, i.e., an exponent traceable function on G, with an image space that is much smaller than expected of random functions. In practice, it would be harder to design such a mapping directly than by composition.

4 Application to Finite Extension Fields

In this section, we explain how the general theory is applied to a concrete case by suggesting an image-restricting function on large extension fields of small finite fields.

Let p be a prime and consider $\mathbf{F}_{p^m}$, the finite field of p^m elements. In this section, we apply results of the previous section to cyclic subgroups of $\mathbf{F}_{p^m}^\times$. We fix a normal basis $\{\alpha, \alpha^p, \ldots, \alpha^{p^{m-1}}\}$ for $\mathbf{F}_{p^m}$ and write elements of $\mathbf{F}_{p^m}$ using the coordinates in the normal basis, i.e., write $\beta = b_0\alpha + \cdots + b_{m-1}\alpha^{p^{m-1}}$ as $\beta = (b_0, ..., b_{m-1})$. As before, our objective is to solve for $\log_g h$ in a cyclic group $G = \langle g \rangle \subset \mathbf{F}_{p^m}^\times$ of prime order q.

There is a natural way to give $\mathbf{F}_p$ an ordering, and once a basis for $\mathbf{F}_{p^m}$ is fixed, we can give $\mathbf{F}_{p^m}$ the dictionary order using the ordering on $\mathbf{F}_p$. In particular, we have given an ordering to elements of $G \subset \mathbf{F}_{p^m}^\times$. We next define the map $\varphi : G \to G$ by

$$\varphi(y) = \min\{\, y^{p^i} \mid i = 0, \ldots, m-1\}. \tag{1}$$

When $\mathbf{F}_{p^m}$ is encoded using the normal basis, the function φ outputs the smallest of all cyclic shifts of its input. Notice that in any realization of φ that uses the normal basis, the number i producing the minimum will be known, in which case φ is naturally exponent traceable. The exponent are simply multiplied by p^i.

To see the effects of the method given in Section 3.1, we need to compare the image size $|\bar{G}|$ with $|G|$. In the process we shall see that φ is almost uniform in its number of pre-images, but this information is not absolutely necessary for our purpose.

Proposition 2. *Let t be the smallest positive integer such that $q|p^t - 1$. Then $|\varphi^{-1}(y)| = t$ for every $y \in \mathrm{Im}(\varphi) \setminus \{\mathrm{id}\}$.*

Proof. Let $y \in G\setminus\{\mathrm{id}\}$. It suffices to show that the number of distinct cyclic shifts of y is t. Suppose $y^{p^j} = y^{p^i}$ for some $0 \le j < i < m$. Writing this as $y^{p^i - p^j} = 1$ and recalling $|\langle y\rangle| = q$, we know $q|p^j(p^{i-j} - 1)$. But, as $\gcd(q, p^j) = 1$, we must have $q|p^{i-j} - 1$. Thus, the choice of t implies that $h, h^p, \cdots, h^{p^{t-1}}$ are all the distinct elements of G, and that this sequence is repeated for further powers. □

The proof of the above proposition has also shown that t is a divisor of m. If t is a proper divisor of m, then we are looking at the situation $G \subset \mathbf{F}_{p^t} \subset \mathbf{F}_{p^m}$. So, in any cryptographic application of $G \subset \mathbf{F}_{p^m}^\times$, we would have $t = m$, for, if otherwise, we would just be wasting resources. We state implications of this thought as a remark.

Remark 2. In any cryptographic application of $G \subset \mathbf{F}_{p^m}^\times$, the function φ of equation (1) is almost pre-image uniform in that the number of pre-images is $|\varphi^{-1}(y)| = m$ for every $y \in \mathrm{Im}(\varphi) \setminus \{\mathrm{id}\}$. Hence the size $|\mathrm{Im}(\varphi)|$ is very close to $|G|/m$.

So far, from discussions of Section 3.1, we know that, by working with f_φ over $\bar{G}$, the DLP over G can be solved with $\sqrt{1/m}$ factor reduction in the number of iterating function applications. To see how this translates into real advantage over the original direct approach on G, it remains to compare the speed of f to that of f_φ.

To evaluate the function φ given by (1), we need to find the smallest among m integers given as elements of $(\mathbf{F}_p)^m = \mathbf{F}_p \times \cdots \times \mathbf{F}_p$. This can be done with $m-1$ comparisons in $(\mathbf{F}_p)^m$. To compare $|\varphi|$ with $|f|$, we want to count this in terms of operations in $\mathbf{F}_p$.

Lemma 1. *The number of operations in $\mathbf{F}_p$ required to compare two integers in $(\mathbf{F}_p)^m$ is expected to be $\frac{p}{p-1}(1 - \frac{1}{p^m})$.*

Proof. Let $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{y} = (y_1, \ldots, y_m)$ be two integers from $(\mathbf{F}_p)^m$. If we had $x_1 = y_1$, $x_2 = y_2$, $\ldots$, $x_{i-1} = y_{i-1}$, but $x_i \neq y_i$, then i comparisons in $\mathbf{F}_p$ would be required to find the smaller of $\mathbf{x}$ and $\mathbf{y}$. Thus our expected number of comparisons is

$$\sum_{i=1}^{m-1} i \cdot \left(\frac{1}{p}\right)^{i-1}\left(1 - \frac{1}{p}\right) \; + \; m \cdot \left(\frac{1}{p}\right)^{m-1},$$

where the last term is written separately because it looks slightly different. Evaluation of the sum results in our claim. □

This lemma shows that if p is large, the first comparison is likely to show us which of the two is smaller, and for even $p = 2$, two comparisons in $\mathbf{F}_2$ are enough. Thus we have the following lemma concerning the speed of φ.

Lemma 2. *On the cyclic group $G \subset \mathbf{F}_{p^m}^\times$, we can expect to evaluate the function φ given by equation* (1) *using $(m-1)\frac{p}{p-1}(1-\frac{1}{p^m})$ operations in $\mathbf{F}_p$.*

In general, evaluation of f is equal to one group operation in G. To compare $|\varphi|$ and $|f|$, we need to know the complexity of multiplication in $\mathbf{F}_{p^m}$ in normal basis representation.

To multiply two elements from a finite field under normal basis, classically, the matrix $T = (t_{i,j})$, defined by

$$\alpha^{p^i} \cdot \alpha^{p^j} = \sum_{0 \le k < m} t_{i-k,k-j}\alpha^{p^k}$$

is considered. For $\mathbf{x} = (x_0, ..., x_{m-1}),\ \mathbf{y} = (y_0, ..., y_{m-1}) \in \mathbf{F}_{p^m}$, product $\mathbf{xy}$ is computed as

$$\mathbf{z} = \mathbf{xy} = (z_0, ..., z_{m-1}),\ z_i = (x_i, ..., x_{i-1})\ T\ (y_i, ..., y_{i-1})^t,$$

where indexes are computed modulo m. Let d be the number of non-zero entries in T. Then multiplication in $\mathbf{F}_{p^m}$ can be done with at most $2md$ operations in $\mathbf{F}_p$ using T. Furthermore, we always have $d \ge 2m-1$ and better ways are known.

Let us say that a multiplication of two polynomials in $\mathbf{F}_p[x]$ of degree less than m can be done with at most $\mathsf{M}(m)$ operations in $\mathbf{F}_p$. Classical polynomial multiplication yields $\mathsf{M}(m) \le 2m^2$, but it is known [21,22] that we may take $\mathsf{M}(m) \in \mathcal{O}(m \log m \log\log m)$.

At the moment, the fastest method for a multiplication under normal basis is one using Gauss periods [20]. The method for multiplication in $\mathbf{F}_{p^m}$ requires $\mathsf{M}(km) + (2k+1)m - 2$ operations in $\mathbf{F}_p$, for a positive number k corresponding to the type of Gauss period used. Thus, even if we disregard the more expensive $\mathsf{M}(km) \in \mathcal{O}(km \log(km) \log\log(km))$ part, whose multiplicative constant we are unaware of, we see that multiplication in $\mathbf{F}_{p^m}$ requires at least $3m$ operations in $\mathbf{F}_p$.

Putting together Lemma 2 and the above information, we can say that the time taken for φ evaluation is expected to be less than roughly $\frac{p}{3(p-1)}$ multiplications in $\mathbf{F}_{p^m}$. Since a usual iterating function involves one multiplication in $\mathbf{F}_{p^m}$, recalling $f_\varphi = \varphi \circ f$, we can state that $|f_\varphi| < \frac{4p-3}{3p-3}|f|$. The following is now a corollary to Theorem 1.

Theorem 2. *For a cyclic subgroup G of $\mathbf{F}_{p^m}^\times$, by using φ as given by equation* (1) *and by working with f_φ over* $\mathrm{Im}(\varphi) \subset G$, *we can speed up variants of the Pollard's rho algorithm by a factor greater than $\frac{3p-3}{4p-3}\sqrt{m}$.*

For example, on the cyclic subgroups of $\mathbf{F}_{2^{1024}}^\times$, assuming that the normal Pollard rho is faster than the index calculus method, we have $m = 1024$ and obtain complexity reduction of the DLP by a factor of at least $2^{4.26}$.

We end the explanation of our algorithm with a word of caution. When dealing with characteristic p fields, since $\varphi(h) = \varphi(h^p)$, any iterating function which utilizes p-th power cannot be used. In particular, with cyclic subgroups of the binary field, the original iterating function f_P, as suggested by the Pollard, cannot be used. This shows that one should pay attention to any glitches that could occur from interaction between φ and the iterating function.

Comparison with tag tracing. The recent work [7] suggests an improvement of the r-adding walk called the tag tracing method for prime fields. It is also possible to apply the idea of [7] to finite extension fields in the normal basis representation, and we have explained this in the appendix.

Let us compare our random walk restriction and the tag tracing method on $\mathbf{F}_{p^m}$. With random walk restriction, the expected rho length is reduced by a $\sqrt{m}$-factor from that of the original r-adding walks while each step takes a little bit more time, i.e., $|f|+|\varphi|$, which is less than $\frac{4p-3}{3p-3}|f|$. On the other hand, in the tag tracing method, the rho length is preserved, but each step is replaced by a tag tracing step consisting of a tag computation, a table lookup, and an occasional computation of f. That is, random walk restriction reduces the number of steps taken by Pollard rho, while tag tracing decreases the effort taken by each step.

For the explicit example $\mathbf{F}_{2^{1024}}$, our random walk restriction achieves time reduction by a factor of $\frac{4p-3}{3p-3}\frac{1}{\sqrt{m}} = \frac{5}{96}$. Hence unless tag computation and table lookup combined requires less than $\frac{5}{96}|f|$ time, random walk restriction will be advantageous over tag tracing. On hardware implementations, where finite field multiplication can be relatively fast, table lookups may become the bottleneck and the goal of $\frac{5}{96}|f|$ per iteration will be hard to achieve with tag tracing. In addition, tag tracing requires a large storage space, which is not needed with random walk restrictions.

5 Application to Pairing Based Cryptosystem

The security of pairing based cryptosystem using a bilinear map $e : G_1 \times G_2 \to G_T$ depends on the DLP on G_1. For Weil or Tate pairings, G_1 is the order q subgroup of points in an elliptic curve $E(\mathbf{F}_{p^\ell})$ and G_T is a cyclic subgroup of $\mathbf{F}^\times_{p^{k\ell}}$. The positive integer k, called the *embedding degree* of G_1, is usually chosen so that it is difficult to solve DLP over $\mathbf{F}^\times_{p^{k\ell}}$ through the index calculus method. Since the MOV attack [13] reduces the DLP on G_1 to the DLP on a subgroup of G_T, it is anticipated that the best way to solve the DLP over G_1 is to use the Pollard rho method on G_1 or on the corresponding order q subgroup of G_T. We shall discuss the effect of our approach when it is applied to the subgroup of G_T.

Let us look at a specific example. The short signature scheme [4] uses supersingular curves $E^+(\mathbf{F}_{3^\ell}) : y^2 = x^3 + 2x + 1$ and $E^-(\mathbf{F}_{3^\ell}) : y^2 = x^3 + 2x - 1$. These curves have the embedding degree 6 and we have $G_T = \mathbf{F}^\times_{3^{6\ell}}$. Thus our method obtains a reduction of DLP complexity by a factor of $\frac{2}{3}\sqrt{6\ell}$ compared to what was originally expected from the given cyclic group. Hence, one should take care to use a larger cyclic subgroup of the elliptic curve than was previously used.

Table 1. Security parameters for short signature

Curve	ℓ	DLog security ($\lceil \log_2 q \rceil$)	DLog security under our approach
E^-	79	126	118
E^+	97	151	142
E^+	121	155	146
E^+	149	220	211
E^+	163	256	247
E^-	163	259	250
E^+	167	262	253

The table 1 was presented in [4], where DLog security refers to the bit size of largest prime divisor of $|E(\mathbf{F}_{3^\ell})|$. We have added a column giving corresponding values when the random walk restriction approach is considered, under the assumption that the complexity of elliptic curve addition is roughly the same as the complexity of finite field multiplication.

It is also noted in the paper [4] that due to the work [6], the DLP over $\mathbf{F}_{3^{6\ell}}^\times$ is easier than on prime fields of equivalent size, so that a large $\mathbf{F}_{3^{6\ell}}^\times$ should be taken. When this advice is followed, our method can work on the larger $\mathbf{F}_{3^{6\ell}}^\times$ to obtain a better reduction of DLP complexity.

Another example is the ID-based encryption [3]. Originally, these systems were built on elliptic curves over a large prime field $\mathbf{F}_p$ of embedding degree 2, on which our methods would yield no advantage. But for efficiency reasons Galbraith [11] suggested the use of elliptic curves over characteristic 2 or 3 fields with Tate pairing of embedding degrees 4 and 6, respectively. These curves are $y^2 + y = x^3 + x$ and $y^2 + y = x^3 + x + 1$ in characteristic 2 and $y^2 = x^3 + 2x \pm 1$ in characteristic 3. In each of these cases, our attack can be applied to reduce the complexity by a factor of $\frac{3}{5}\sqrt{4\ell}$ and $\frac{2}{3}\sqrt{6\ell}$, respectively. In practice, if one chooses $\ell = 283$ or 397 with the curve $y^2 + y = x^3 + x + 1$, we can speed up the Pollard rho algorithm by a factor of 20.2 or 23.9, respectively.

6 Conclusion

In this paper, we proposed a variant of the Pollard rho method, called random walk restriction, and showed that this idea can be applied to cyclic subgroups of finite fields of large extension degree. The main idea is to restrict the random walk of the Pollard rho to a smaller set which results in an earlier collision. As a result, our algorithm achieves speedup over the Pollard rho method by a factor of approximately $\frac{3p-3}{4p-3}\sqrt{m}$ in subgroups of $\mathbf{F}_{p^m}^\times$ and can be applied to pairing based cryptosystems.

Acknowledgments. Minkyu Kim and Jung Hee Cheon were supported by the Korea Science and Engineering Foundation (KOSEF) grant. (No. R01-2008-000-11287-0)

References

1. Memoirs of the 3rd Cryptology Paper Contest, arranged by a Korean government organization (written in Korean) (2007)
2. Adleman, L.: A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography. In: Proc. of the IEEE 20th Annual Symposium on Foundations of Computer Science (FOCS), pp. 55–60 (1979)
3. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
4. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. J. Cryptology 17, 297–319 (2004)
5. Brent, R.: An improved Monte Carlo Factorization Algorithm. BIT 20, 176–184 (1980)
6. Coppersmith, D.: Fast Evaluation of Logarithms in Fields of Characteristic Two. IEEE Trans. Inform. Theory 30(4), 587–594 (1984)
7. Cheon, J., Hong, J., Kim, M.: Speeding up Pollard Rho Method on Prime Fields. In: Asiacrypt 2008. LNCS, vol. 5350, pp. 471–488. Springer, Heidelberg (2008)
8. Duursma, I., Gaudry, P., Morain, F.: Speeding up the Discrete Log Computation on Curves with Automorphisms. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 103–121. Springer, Heidelberg (1999)
9. Knuth, D.: The Art of Computer Programming. Seminumerical Algorithms, vol. II. Addison-Wesley, Reading (1969)
10. Frey, G., Rück, H.-G.: A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. Math. Comp. 62, 865–874 (1994)
11. Galbraith, S.: Supersingular Curves in Cryptography. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 495–513. Springer, Heidelberg (2001)
12. Gallant, R., Lambert, R., Vanstone, S.: Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves. Math. Comp. 69, 1699–1705 (2000)
13. Menezes, A., Okamoto, T., Vanstone, P.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. Inform. Theory 39(5), 1636–1649 (1993)
14. Nivasch, G.: Cycle Detection using a Stack. Information Processing Letters 90, 135–140 (2004)
15. Pollard, J.: A Monte Carlo Method for Index Computation (mod p). Math. Comp. 32(143), 918–924 (1978)
16. Quisquater, J., Delescaille, J.: How easy is Collision Search? Application to DES. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 429–434. Springer, Heidelberg (1990)
17. Sattler, J., Schnorr, C.: Generating Random Walks in Groups. Ann.-Univ.-Sci.-Budapest.-Sect.-Comput. 6, 65–79 (1985)
18. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
19. Schnorr, C., Lenstra Jr., H.: A Monte Carlo Factoring Algorithm with Linear Storage. Math. Comp. 43(167), 289–311 (1984)
20. Gao, S., von zur Gathen, J., Panario, D., Shoup, V.: Algorithms for Exponentiation in Finite Fields. Journal of Symbolic Computation 29(6), 879–889 (2000)
21. Schönhage, A., Strassen, V.: Schnelle Multiplikation großer Zahlen. Computing 7, 281–292 (1971)

22. Schönhage, A.: Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. Acta Informatica 7, 395–398 (1977)
23. Sedgewick, R., Szymanski, T., Yao, A.: The Complexity of Finding Cycles in Periodic Functions. SIAM Journal on Computing 11(2), 376–390 (1982)
24. Teske, E.: Speeding up Pollard's rho Method for Computing Discrete Logarithms. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 541–554. Springer, Heidelberg (1998)
25. Teske, E.: On Random Walks for Pollard's rho Method. Math. Comp. 70, 809–825 (2001)
26. van Oorschot, P., Wiener, M.: Parallel Collision Search with Cryptanalytic Applications. J. Cryptology 12, 1–28 (1999)
27. Wiener, M., Zuccherato, R.: Fast Attacks on Elliptic Curve Cryptosystems. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 190–200. Springer, Heidelberg (1999)

Appendix: Tag Tracing Method with Normal Basis

In this appendix, we explain how tag tracing [7] can be applied to binary field with normal basis representation. Of course, this can then be easily adapted to any $\mathbf{F}_{p^m}$.

The tag tracing method is an improvement of the Pollard rho variant that uses r-adding walks. The main idea is to reduce the complexity of each iterating step. In the original r-adding walks, the iterative steps compute $g_{i+1} = g_i M_s$ from g_i, where $s = s(g_i)$ is the index of g_i computed by a function $s : G \to \{0, 1, \ldots, r-1\}$. This involves computing a full product at each step. But, if we have another function $\overline{s} : G \times G \to \{0, 1, \ldots, r-1\} \cup \{\text{fail}\}$ which computes the index of $g_i M_s$ without fully computing the product, we can reduce the execution time for each step as follows.

Let $\mathcal{M} = \{M_s = g^{m_s} h^{n_s}\}$. Fix a positive integer ℓ and prepare a pre-computed table of $\mathcal{M}_\ell = (\mathcal{M} \cup \{1\})^\ell$. Given the i-th element $g_i \in G$ of the walk, compute the index $s_i = s(g_i)$. Without a full product of g_i and M_{s_i}, compute $s_{i+1} = \overline{s}(g_i, M_{s_i})$. Now, since $M_{s_i} M_{s_{i+1}} \in \mathcal{M}_\ell$, we can evaluate next index $s_{i+2} = \overline{s}(g_i, M_{s_i} M_{s_{i+1}})$. Continue this process until we arrive at ℓ iterations or need to store current g_j. It is easy to see that the execution time for each step is expected to be

$$|\overline{s}| + (\frac{1}{\ell} + P_{\text{fail}})|f|,$$

where P_{fail} is the probability of $\overline{s}$ evaluation to fail.

To apply the above procedure to $\mathbf{F}_{2^m}$ with the normal basis representation, define $s : G \to \{0, ..., r-1\}$ by

$$s(\mathbf{z}) = (z_0, z_1, \ldots, z_{\lceil \log r \rceil - 1}),$$

for $\mathbf{z} = (z_0, z_1, \ldots, z_{m-1}) \in \mathbf{F}_{2^m}$. Given $\mathbf{x}, \mathbf{y} \in \mathbf{F}_{2^m}$, since we can write their product as $\mathbf{xy} = (\mathbf{x}T\mathbf{y}^t, \ldots, \mathbf{x}^{p^{m-1}} T \big(\mathbf{y}^{p^{m-1}}\big)^t)$, we can compute $s(\mathbf{xy})$ with only

$\lceil \log r \rceil \cdot m$ bit operations using pre-computed $T\mathbf{y}^t$, $T(\mathbf{y}^p)^t$, ..., $T(\mathbf{y}^{p^{\lceil \log r \rceil - 1}})^t$. This does not involve a full product of $\mathbf{x}$ and $\mathbf{y}$. Moreover, $P_{\text{fail}} = 0$.

Hence, the complexity of each steps of tag tracing is expected to

$$\lceil \log r \rceil \cdot m + \frac{1}{l} \cdot \mathsf{Mul},$$

where Mul denote the expected number of bit operations for multiplication in $\mathbf{F}_{2^m}$ using normal basis.

Signing a Linear Subspace: Signature Schemes for Network Coding

Dan Boneh[1,⋆], David Freeman[2,⋆⋆], Jonathan Katz[3,⋆⋆⋆], and Brent Waters[4,†]

[1] Stanford University,
dabo@cs.stanford.edu
[2] CWI and Universiteit Leiden,
freeman@cwi.nl
[3] University of Maryland,
jkatz@cs.umd.edu
[4] University of Texas at Austin,
bwaters@cs.utexas.edu

Abstract. Network coding offers increased throughput and improved robustness to random faults in completely decentralized networks. In contrast to traditional routing schemes, however, network coding requires intermediate nodes to modify data packets *en route*; for this reason, standard signature schemes are inapplicable and it is a challenge to provide resilience to tampering by malicious nodes.

We propose two signature schemes that can be used in conjunction with network coding to prevent malicious modification of data. Our schemes can be viewed as signing linear subspaces in the sense that a signature σ on a subspace V authenticates exactly those vectors in V. Our first scheme is (suitably) *homomorphic* and has *constant* public-key size and per-packet overhead. Our second scheme does not rely on random oracles and is based on weaker assumptions.

We also prove a lower bound on the length of signatures for linear subspaces showing that our schemes are essentially optimal in this regard.

1 Introduction

Network coding [1,23] refers to a general class of routing mechanisms where, in contrast to traditional "store-and-forward" routing, intermediate nodes *modify*

[⋆] Supported by DARPA IAMANET, NSF, and the Packard Foundation.

[⋆⋆] Research conducted at Stanford University. Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

[⋆⋆⋆] Supported by NSF CNS-0447075, NSF CNS-0627306, the U.S. DoD/ARO MURI program, and the US Army Research Laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001.

[†] Supported by NSF CNS-0749931, CNS-0524252, CNS-0716199, the U.S. Army Research Office under the CyberTA Grant No. W911NF-06-1-0316, and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. Portions of this research were conducted while the author was at SRI International.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 68–87, 2009.

data packets in transit. Network coding has been shown to offer a number of advantages with respect to traditional routing, the most well-known of which is the possibility of increased throughput in certain network topologies. It has also been suggested as a means of improving robustness against random network failures since, as with erasure codes [6], the destination can recover the original data (with high probability) once it has received sufficiently many correct packets, even if a large fraction of packets are lost.

Because of these advantages, network coding has been proposed for applications in wireless and/or ad-hoc networks, where communication is at a premium and centralized control may be unavailable; it has also been suggested as an efficient means for content distribution in peer-to-peer networks [22], and for improving the performance of large-scale data dissemination over the Internet [11].

A major concern in systems that use network coding is protecting against malicious modification of packets (i.e., "pollution attacks") by Byzantine nodes; see [13,21] for two recent surveys and Section 2.2 for a discussion of previous work. The problem is particularly acute because errors introduced into even a single packet can propagate and pollute multiple packets making their way to the destination. This propagation is a consequence of the processing that honest nodes, downstream of any corrupted packets, apply to all incoming packets.

We propose two signature schemes that can be used to provide cryptographic protection against pollution attacks even when the adversary can corrupt an arbitrary number of nodes, eavesdrop on all network traffic, and insert or modify an arbitrary number of packets. Of course, the destination cannot possibly recover the file unless it receives a minimum number of uncorrupted packets; once this is the case, however, our schemes ensure that the destination can filter out any corrupted packets and recover the correct file. As our signatures are publicly verifiable, intermediate nodes could discard corrupted packets as well (though whether this is actually done will depend on the computational resources of the intermediate nodes). Our first scheme is particularly efficient, with both public-key size and per-packet overhead being constant. A detailed discussion of our schemes, and their advantages relative to prior work, is given in Section 2.3.

Our schemes can be viewed as signing linear subspaces in the sense that a signature σ on the subspace V authenticates exactly those vectors in V. We prove a lower bound on the signature length for any scheme for signing linear subspaces (under some mild restrictions), showing that our constructions are essentially optimal in this regard.

Outline of the paper. We provide a quick overview of network coding in Section 2.1. In Section 2.2 we discuss prior work addressing adversarial behavior in the context of network coding, and we describe the advantages of our schemes in Section 2.3. In Section 3 we introduce appropriate definitions of security for our setting and give relevant mathematical background. Sections 4 and 5 describe our constructions, and in Section 6 we prove our lower bound.

2 Background

2.1 Linear Network Coding

In a *linear* network coding scheme [23] (the only type with which we will be concerned), a file to be transmitted is viewed as an ordered sequence of n-dimensional vectors $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m \in \mathbb{F}_p^n$, where p is prime. We will sometimes refer to individual vectors as *blocks* or *packeets.* Before transmission, the source node creates the m *augmented vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_m$ given by:

$$\mathbf{v}_i = (\text{—}\bar{\mathbf{v}}_i\text{—}, \overbrace{\underbrace{0, \ldots, 0, 1}_{i}, 0, \ldots, 0}^{m}) \in \mathbb{F}_p^{n+m} \, ;$$

that is, each original vector $\bar{\mathbf{v}}_i$ is appended with the vector of length m containing a single '1' in the ith position. These augmented vectors are then sent by the source as packets in the network. Since this step introduces $\Theta(m^2)$ communication overhead per file, one typically chooses $m \ll n$.

Each node in the network processes packets as follows. Upon receiving packets (i.e., vectors) $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \mathbb{F}_p^{n+m}$ on its ℓ incoming communication edges, node i computes the packet (vector) $\mathbf{w} = \sum_{j=1}^{\ell} \alpha_{i,j} \mathbf{w}_j$, where $\alpha_{i,j} \in \mathbb{F}_p$. The resulting vector $\mathbf{w}$ is then transmitted on the node's outgoing edges. That is, each node transmits a *linear combination* of the packets it receives. Thus, in a fault-free execution of the scheme, all packets transmitted on any link in the network are linear combinations of the original (augmented) file vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$.

The weights $\alpha_{i,j}$ used by the ith node in the network can be established by a central authority. More usefully (and more interestingly), however, these values can also be chosen randomly and independently by each node in a completely decentralized fashion. (In this case the scheme is sometimes referred to as "random network coding".) Although carefully designed codes can potentially have better performance, it has been shown that random network coding does almost as well with high probability [8,14,16].

There may be multiple destination nodes (i.e., receivers) who wish to obtain the original file from the source. When any such node receives m linearly independent vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$, it can recover the original file as follows: For a received vector $\mathbf{w}_i$, let $\mathbf{w}_i^L$ denote the left-most n positions of the vector, and let $\mathbf{w}_i^R$ denote the right-most m positions. The receiver first computes an $m \times m$ matrix G such that

$$G = \begin{pmatrix} \text{—}\mathbf{w}_1^R\text{—} \\ \vdots \\ \text{—}\mathbf{w}_m^R\text{—} \end{pmatrix}^{-1} . \tag{1}$$

(The matrix on the right-hand side is invertible as long as all the received vectors are correct.) The original file $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m$ is then given by

$$\begin{pmatrix} -\bar{\mathbf{v}}_1- \\ \vdots \\ -\bar{\mathbf{v}}_m- \end{pmatrix} = G \cdot \begin{pmatrix} -\mathbf{w}_1^L- \\ \vdots \\ -\mathbf{w}_m^L- \end{pmatrix}.$$

We stress that the receiver need not be aware of the weights $\{\alpha_{i,j}\}$ used by any intermediate node in the network in order to recover the file. On the other hand, if the weights used by the intermediate nodes *are* all known to the receiver (and the receiver is aware of the network topology) then the matrix G can be computed in advance and, in fact, the scheme can be run on the original file vectors $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m$ rather than on the augmented vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$. In our work, however, we will assume that augmented vectors are used.

2.2 Dealing with Adversarial Behavior

Network coding can offer resilience to random packet loss since the receiver can reconstruct the original file from *any* set of m correctly formed, linearly independent vectors. (Notice the similarity with linear erasure codes introduced in other contexts, e.g., [6].) However, the in-network processing done by the nodes makes the basic network coding scheme extremely susceptible to *malicious* errors introduced by even a single intermediate node in the network. For starters, this is because the basic network coding scheme offers no means of isolating the fault: if one of the vectors $\mathbf{w}_i$ received at the destination is incorrect, then that error will be "spread" across (potentially) every block $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_m$ of the reconstructed file (cf. Equation 1). Furthermore, a single error introduced by one malicious node will be propagated by every node further downstream. Thus, even a faulty transmission on a single edge (say, due to a single corrupted node) will eventually cause almost all vectors being forwarded in the network to be incorrect, and will thus prevent reconstruction of even a portion of the file.

It is worth mentioning two trivial approaches that do not solve the problem. The source cannot simply sign the packets it releases into the network using a standard signature scheme, since the packets received at the destination will almost surely be different from those issued by the sender. Signing the entire file does not work either: although this would ensure that the receiver never accepts an incorrect file, there is no efficient way for the receiver to recover the correct file. (Since the receiver cannot distinguish correct packets from corrupt packets *a priori*, it is forced to apply the reconstruction procedure from Section 2.1 to all subsets of received vectors of size m.)

We now survey other techniques for combatting data pollution when network coding is used (see [21] for further discussion).

Information-theoretic approaches. Information-theoretic methods for enabling recovery from malicious faults work by introducing *redundancy* into the original packets transmitted by the sender [15,17,18]. Such techniques have the advantage of not relying on any computational assumptions, but are limited to offering security only against a relatively limited class of adversaries: these constructions all (inherently) assume limitations on the number of nodes the adversary can corrupt, the number of packets that can be modified, and/or the number of links on which the adversary can eavesdrop. Moreover, the communication overhead introduced by these schemes is significant.

Cryptographic approaches. Existing cryptographic schemes (i.e., those that protect only against a computationally bounded adversary) all work by providing a way for honest nodes to verify authenticity of *individual* packets. (Once again we stress that this cannot be achieved in our setting using standard signatures, since packets are modified in transit.) Cryptographic schemes can potentially offer resilience against an adversary who eavesdrops on the entire network and controls arbitrarily many malicious nodes, as long as the destination node receives m correctly formed and linearly independent vectors. Existing schemes also allow the receiver to recover gracefully when fewer than m legitimate vectors are received; for example, if the destination receives k correctly formed vectors spanning the subspace defined by the first k file blocks, then the receiver can at least recover a portion of the original file. Cryptographic schemes have the additional advantage that intermediate nodes in the network can verify correctness of individual packets, and hence reject ill-formed ones.

Although one could imagine using a symmetric-key approach, all existing work focuses on the public-key setting where the sender's public key is known to all other nodes in the network. A public-key scheme makes the most sense when the sender is multi-casting files to many receivers (as is typically the situation when network coding is used), and furthermore enables all intermediate nodes in the network to potentially verify authenticity of received packets.

Krohn et al. [22] (see also [11,12]) suggest *homomorphic hashing* for preventing pollution attacks. In their scheme, the sender computes a hash $h_i = H(\bar{\mathbf{v}}_i)$ of each block of the file; given $\mathbf{x} = (h_1, \ldots, h_m)$, anyone can check whether a packet $\mathbf{w}$ is a correctly formed linear combination of the augmented vectors $\{\mathbf{v}_i\}$. Krohn et al. assume a reliable channel for distributing the hash values for a given file, but it is not hard to show (see Section 5) that signing $\mathbf{x}$ using a standard signature scheme also results in a secure solution. The drawback of this approach is that both the authentication information $\mathbf{x}$ and the public keys are large: $\mathbf{x}$ has size $\Theta(km)$ and the public key has size $\Theta(kn)$, where k is a cryptographic security parameter. (Thus, either the public key or $\mathbf{x}$ has size at least the square root of the file size.) Sending all of $\mathbf{x}$ with each packet introduces a large overhead; on the other hand, if $\mathbf{x}$ is partitioned among multiple packets then intermediate nodes cannot verify authenticity of the packets they receive.

Zhao et al. [25] propose a scheme where the sender computes some authentication information $\mathbf{x}$ derived from a vector *orthogonal* to the space $V = \mathrm{span}(\{\mathbf{v}_1, \ldots, \mathbf{v}_m\})$; this authentication information $\mathbf{x}$ is then signed by the sender (using a standard signature scheme). Unfortunately, this scheme also has relatively poor performance: both $\mathbf{x}$ and the public keys have size $\Theta(k(n + m))$. Furthermore, the scheme can only be used for distributing a single file, after which the public key must be refreshed. (Zhao et al. suggest some approaches for handling multiple files, but do not prove security of any of these suggestions.) An additional drawback of both this scheme and the one of Krohn et al. is that they both require the sender to know the entire file before the authentication information can be computed.

Charles et al. [7] present a *homomorphic signature scheme* [19] based on the aggregate signature scheme of Boneh et al. [4]. This scheme has the property that valid signatures $\sigma_1, \ldots, \sigma_k$ on vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$, respectively, can be combined, without knowledge of the signer's secret key, to produce a valid signature σ on any linear combination $\sum_i \alpha_i \mathbf{w}_i$. The scheme can only be used to sign a *single* file, after which the public key must be refreshed; this restriction clearly limits the scheme's applicability. Public keys in this scheme have size $\Theta(k(m+n))$, meaning that it will be impractical to redistribute public keys over the network even if network coding is used for key distribution. Charles et al. also do not formally prove security of their scheme.

2.3 Our Contributions

We start with clean definitions of the problem at hand and formally define what it means for a signature scheme to be secure in our context. Roughly speaking, we consider signature schemes that can be viewed as authenticating *linear subspaces* in the sense that a signature on the subspace V authenticates all vectors in V. Our security definition requires that no adversary given a signature on a vector subspace V can forge a valid signature for any vector not in V. The application to network coding is clear: to distribute a file, simply sign the subspace $V = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$. (Actually, both our security definition and our constructions directly take into account the distribution of multiple files using a single public key — in contrast to [7,25] — and so the formal definition and the application to network coding is a bit more involved.)

We show two constructions meeting our definition. Our first scheme, called NCS_1, is a homomorphic signature scheme and has the advantage that signatures can be associated with individual vectors rather than an entire subspace. (The signature on a linear subspace V can then be taken as the collection of signatures on a set of basis vectors for V.) Both the public key and per-vector signatures in this scheme have *constant* size, making the scheme ideally suited for network coding. This scheme also supports the transmission of *streaming data*, in the sense that the sender need not be aware of the entire file before computing the signature on the first packet. Security of this scheme is proved based on the computational Diffie-Hellman assumption in bilinear groups, in the random oracle model.

Our second construction, called NCS_2, provides an instantiation of the scheme of Krohn et al. [22] that is secure according to our definition. The primary advantage of this scheme is that it can be proven secure based on a potentially weaker assumption (namely, the discrete logarithm assumption) without random oracles. We also show how our scheme can be viewed as a more efficient version of the scheme of Zhao et al. [25].

Finally, we prove a lower bound on the length of secure signatures for linear subspaces, under some mild assumptions on the signature scheme. Specifically, we show (roughly speaking) that the signature on any subspace V must have length proportional to $\dim(V)$. This shows that our two constructions are essentially optimal in this regard. (Note that although our first scheme offers constant size per-vector signatures, the signature on a subspace V consists of $\dim(V)$ per-vector signatures and thus matches the lower bound.)

3 Definitions and Preliminaries

3.1 Signing a Linear Subspace

We abstract our problem, and seek to design a signature scheme that signs a subspace $V \subset \mathbb{F}_p^N$ so that only vectors $\mathbf{y} \in V$ are accepted as valid. We start by defining the interface provided by such a system and then define security.

As discussed previously, we want our scheme to be useful for the distribution of multiple files using the same public key. As such, every file will be associated with an *identifier* id that is chosen by the sender at the time the first packet associated with the file is transmitted.[1] We then require that every packet forwarded in the system is labeled with the appropriate identifier. (Adversarial nodes, of course, can change the identifier any way they like.) The identifier provides a mechanism for honest nodes, and the receiver in particular, to distinguish packets associated with different files.

Definition 1. A *network coding signature scheme* is a triple of probabilistic, polynomial-time algorithms (Setup, Sign, Verify) with the following functionality:

- $\mathsf{Setup}(1^k, N)$. On input a security parameter 1^k and an integer N, this algorithm outputs a prime p, a public key PK, and a secret key SK.
- $\mathsf{Sign}(SK, \mathsf{id}, V)$. On input a secret key SK, a file identifier $\mathsf{id} \in \{0,1\}^k$, and an m-dimensional subspace $V \subset \mathbb{F}_p^N$ (with $0 < m < N$) described as a set of basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$, this algorithm outputs a signature σ.
- $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. On input a public key PK, an identifier $\mathsf{id} \in \{0,1\}^k$, a vector $\mathbf{y} \in \mathbb{F}_p^N$, and a signature σ, this algorithm outputs either 0 (reject) or 1 (accept).

We require that for each (p, PK, SK) output by $\mathsf{Setup}(1^k, N)$, the following holds: for all m-dimensional subspaces $V \subset \mathbb{F}_p^N$ with $0 < m < N$, and for all $\mathsf{id} \in \{0,1\}^k$, if $\sigma \leftarrow \mathsf{Sign}(SK, \mathsf{id}, V)$ then $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$ for all $\mathbf{y} \in V$.

The signature σ output by Sign can be viewed a signature on the vector space V. "Homomorphic signatures" (cf. [19]) are a special case that is more precisely modeled by a definition in which the Sign algorithm produces signatures $\sigma_1, \ldots,$ σ_m on the basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$, and the collection of these signatures constitutes a signature on V. This is encapsulated in the following definition.

[1] One can think of this identifier as being equivalent to a filename, though for our first scheme we require that identifiers be *unpredictable* (they need not be random). Unpredictability is easily achieved by concatenating an arbitrary filename with a random string.

Definition 2. A *homomorphic network coding signature scheme* is a tuple of probabilistic, polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ with the following functionality:

- $\mathsf{Setup}(1^k, N)$. On input a security parameter 1^k and an integer N, this algorithm outputs a prime p, a public key PK, and a secret key SK.
- $\mathsf{Sign}(SK, \mathsf{id}, \mathbf{v})$. On input a secret key SK, a file identifier $\mathsf{id} \in \{0,1\}^k$, and a vector $\mathbf{v} \in \mathbb{F}_p^N$, this algorithm outputs a signature σ.
- $\mathsf{Combine}(PK, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$. On input a public key PK, a file identifier id, and a set of tuples $\{(\beta_i, \sigma_i)\}_{i=1}^{\ell}$ with $\beta_i \in \mathbb{F}_p$, this algorithm outputs a signature σ.
  (The intuition is that if each σ_i is a valid signature on the vector $\mathbf{v}_i$, then σ is a signature on $\sum_{i=1}^{\ell} \beta_i \mathbf{v}_i$.)
- $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. On input a public key PK, an identifier $\mathsf{id} \in \{0,1\}^k$, a vector $\mathbf{y} \in \mathbb{F}_p^N$, and a signature σ, this algorithm outputs either 0 (reject) or 1 (accept).

We require that for each (p, PK, SK) output by $\mathsf{Setup}(1^k, N)$, the following hold:

- For all id and all $\mathbf{y} \in \mathbb{F}_p^N$, if $\sigma \leftarrow \mathsf{Sign}(SK, \mathsf{id}, \mathbf{y})$ then $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$.
- For all $\mathsf{id} \in \{0,1\}^k$ and all sets of triples $\{(\beta_i, \sigma_i, \mathbf{v}_i)\}_{i=1}^{\ell}$, if it holds that $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{v}_i, \sigma_i) = 1$ for all i, then

$$\mathsf{Verify}\Big(PK,\ \mathsf{id},\ \textstyle\sum_i \beta_i \mathbf{v}_i,\ \mathsf{Combine}\big(PK,\ \mathsf{id},\ \{(\beta_i, \sigma_i)\}_{i=1}^{\ell}\big)\Big) = 1.$$

The following lemma, a proof of which is trivial, shows that homomorphic network coding signatures are indeed a special case of network coding signatures.

Lemma 3. *Let* $\mathcal{S}_2 = (\mathsf{Setup}_2, \mathsf{Sign}_2, \mathsf{Combine}_2, \mathsf{Verify}_2)$ *be a homomorphic network coding signature scheme. Then* $\mathcal{S}_1 = (\mathsf{Setup}_1, \mathsf{Sign}_1, \mathsf{Verify}_1)$ *defined as follows is a network coding signature scheme.*

- $\mathsf{Setup}_1(1^k, N)$ *runs* $\mathsf{Setup}_2(1^k, N)$ *and outputs the result.*
- $\mathsf{Sign}_1(SK, \mathsf{id}, V)$ *runs* $\mathsf{Sign}_2(SK, \mathsf{id}, \mathbf{v}_1)$, …, $\mathsf{Sign}_2(SK, \mathsf{id}, \mathbf{v}_m)$, *where* $\mathbf{v}_1$, …, $\mathbf{v}_m$ *is any basis of* V. *It then outputs* $\sigma = ((\mathbf{v}_1, \sigma_1), \ldots, (\mathbf{v}_m, \sigma_m))$.
- $\mathsf{Verify}_1(PK, \mathsf{id}, \mathbf{y}, \sigma)$ *parses* σ *as* $((\mathbf{v}_1, \sigma_1), \ldots, (\mathbf{v}_m, \sigma_m))$, *and computes coefficients* $\{\beta_i\}$ *such that* $\mathbf{y} = \sum_i \beta_i \mathbf{v}_i$ *(if no solution exists, then it outputs 0). Finally, it outputs* $\mathsf{Verify}_2\Big(PK,\ \mathsf{id},\ \mathbf{y},\ \mathsf{Combine}_2\big(PK,\ \mathsf{id},\ \{(\beta_i, \sigma_i)\}_{i=1}^{m}\big)\Big)$.

We say a basis $\{\mathbf{v}_i\}_{i=1}^m$ of a subspace $V \subseteq \mathbb{F}_p^N$ is *properly augmented* (cf. Section 2.1) if the last m coordinates of each $\mathbf{v}_i$ form a unit vector with a 1 in the ith position. Abusing notation, we say V is properly augmented if it is described using a properly augmented basis. Let $n = N - m$. If $\mathbf{v}_1, \ldots, \mathbf{v}_m$ is a properly augmented basis of V, then for any $\mathbf{y} \in \mathbb{F}_p^N$ with $\mathbf{y} = (y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+m})$ we have

$$\mathbf{y} \in V \iff \mathbf{y} = \sum_{i=1}^{m} y_{n+i} \mathbf{v}_i. \tag{2}$$

This observation allows us to simplify the construction of Lemma 3 when we only use $(\mathsf{Setup}_1, \mathsf{Sign}_1, \mathsf{Verify}_1)$ to sign properly augmented vector spaces. (This suffices for our application to network coding.) Namely, $\mathsf{Sign}_1(SK, \mathsf{id}, V)$ simply outputs $\sigma = (\sigma_1, \ldots, \sigma_m)$ (where the σ_i are computed as in Lemma 3), and $\mathsf{Verify}_1(PK, \mathsf{id}, \mathbf{y}, \sigma)$ outputs

$$\mathsf{Verify}_2\left(PK,\ \mathsf{id},\ \mathbf{y},\ \mathsf{Combine}_2\big(PK,\ \mathsf{id},\ \{(y_{N-m+i}, \sigma_i)\}_{i=1}^{m}\big)\right).$$

Security. We define security of a network coding signature scheme, and say that a homomorphic network coding signature scheme $\mathcal{S}_2$ is secure if the network coding signature scheme $\mathcal{S}_1$ constructed from $\mathcal{S}_2$ as in Lemma 3 is secure.

Definition 4. A network coding signature scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ is *secure* if the advantage of any probabilistic, polynomial-time adversary $\mathcal{A}$ in the following security game is negligible in the security parameter k:

Setup: The adversary $\mathcal{A}$ sends a positive integer N to the challenger. The challenger runs $\mathsf{Setup}(1^k, N)$ to obtain (p, PK, SK), and gives p and PK to $\mathcal{A}$.

Queries: Proceeding adaptively, $\mathcal{A}$ specifies a sequence of vector subspaces $V_i \subset \mathbb{F}_p^N$. For each i, the challenger chooses id_i uniformly from $\{0,1\}^k$, and gives id_i and $\sigma_i \leftarrow \mathsf{Sign}(SK, \mathsf{id}_i, V_i)$ to $\mathcal{A}$.

Output: $\mathcal{A}$ outputs $\mathsf{id}^* \in \{0,1\}^k$, a *non-zero* vector $\mathbf{y}^* \in \mathbb{F}_p^N$, and a signature σ^*.

The adversary *wins* if $\mathsf{Verify}(PK, \mathsf{id}^*, \mathbf{y}^*, \sigma^*) = 1$, and either (1) $\mathsf{id}^* \neq \mathsf{id}_i$ for all i (a *type 1 forgery*), or (2) $\mathsf{id}^* = \mathsf{id}_i$ for some i but $\mathbf{y}^* \notin V_i$ (a *type 2 forgery*). The *advantage* NC-Adv$[\mathcal{A}, \mathcal{S}]$ of $\mathcal{A}$ is defined to be the probability that $\mathcal{A}$ wins the security game.

We require the adversary to output a *non-zero* vector $\mathbf{y}^*$ since the zero vector lies in every linear subspace. (Furthermore, by adding a dimension it is possible to rule out type-1 forgeries on the zero vector if desired.) Note also that it is not counted as a forgery if $\mathcal{A}$ obtains a signature σ on a vector space V and outputs a valid signature σ' on a vector space $V' \subset V$. Indeed, in the context of network coding this would not be problematic.

3.2 Bilinear Groups and Complexity Assumptions

We briefly review the framework of groups with bilinear maps.

Definition 5. A *bilinear group tuple* is a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \varphi)$ with the following properties:

1. $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, in which random sampling and group operations are efficiently computable.
2. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable map satisfying the following:
   (a) Bilinearity: for any $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}$, $e(g^a, h^b) = e(g,h)^{ab}$.

(b) Non-degeneracy: if g generates $\mathbb{G}_1$ and h generates $\mathbb{G}_2$, then $e(g,h)$ generates $\mathbb{G}_T$.

3. $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ is an efficiently computable isomorphism.[2]

For cryptographic applications, we require that the *discrete logarithm problem* — i.e., computing x given g and g^x — be infeasible in the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. Given an algorithm $\mathcal{A}$ that takes as input two elements g, h in a group $\mathbb{G}$ and outputs an integer x in $\{0, \ldots, |\mathbb{G}| - 1\}$, we define DL-Adv$[\mathcal{A}, \mathbb{G}]$ to be the probability that $h = g^x$, taken over inputs (g, h) and the random coins of $\mathcal{A}$.

Currently the only known bilinear group tuples in which the discrete logarithm problems are believed to be infeasible are those for which $\mathbb{G}_1, \mathbb{G}_2$ are (subgroups of) groups of rational points on elliptic curves or abelian varieties over finite fields; $\mathbb{G}_T$ is (a subgroup of) a multiplicative group of a finite field; and e is (a variant of) the Weil pairing or Tate pairing [9]. Elliptic curves and abelian varieties with the desired properties are called "pairing-friendly." Bilinear group tuples as described in Definition 5 exist on all pairing-friendly elliptic curves and abelian varieties with embedding degree $k > 1$.

The proof of security of our first signature scheme (Section 4) relies on the assumption that the *co-computational Diffie Hellman* (*co-CDH*) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ — i.e., computing $g^x \in \mathbb{G}_1$ given $g \in \mathbb{G}_1 \setminus \{1\}$ and $h, h^x \in \mathbb{G}_2 \setminus \{1\}$ — is infeasible. Given $\mathcal{A}$ that takes as input $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $z = h^x \in \mathbb{G}_2$, and outputs an element $\omega \in \mathbb{G}_1$, we define co-CDH-Adv$[\mathcal{A}, (\mathbb{G}_1, \mathbb{G}_2)]$ to be the probability that $\omega = g^x$, taken over inputs (g, h, z) and the random coins of $\mathcal{A}$. Note that if in addition to $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ there is an efficiently computable isomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$ then the co-CDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is equivalent to the standard computational Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$.

4 A Homomorphic Network Coding Signature Scheme

In this section we construct a homomorphic network coding signature scheme with constant-size public key and constant-size per-vector signatures. Our signatures are similar to the aggregate signatures of Boneh et al. [4].

Signature Scheme $\mathbf{NCS}_0$

$\mathsf{Setup}(1^k, N)$. Given a security parameter 1^k and a positive integer N, do:

1. Generate a bilinear group tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \varphi)$ such that $p > 2^k$. Choose a generator $h \leftarrow \mathbb{G}_2 \setminus \{1\}$.
2. Choose $\alpha \leftarrow \mathbb{F}_p^*$, and set $u := h^\alpha$.
3. Let $H : \{0,1\}^* \times \{0,1\}^* \to \mathbb{G}_1$ be a hash function, viewed as a random oracle.
4. Output p, the public key $PK := (\mathcal{G}, H, h, u)$ and the private key $SK := \alpha$.

[2] Existence of φ is not needed if we use a different cryptographic assumption to prove security of our first scheme. We omit further discussion.

$\mathsf{Sign}(SK, \mathsf{id}, \mathbf{v})$. Given a secret key $SK = \alpha$, an identifier $\mathsf{id} \in \{0,1\}^k$, and a vector $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{F}_p^N$, this algorithm outputs the signature

$$\sigma := \left(\prod_{i=1}^{N} H(\mathsf{id}, i)^{v_i} \right)^{\alpha} .$$

$\mathsf{Combine}(PK, \mathsf{id}, \{(\beta_i, \sigma_i)\}_{i=1}^{\ell})$. Given a public key PK, an identifier id, and $\{(\beta_i, \sigma_i)\}_{i=1}^{\ell}$ with $\beta_i \in \mathbb{F}_p$, this algorithm outputs $\sigma := \prod_{i=1}^{\ell} \sigma_i^{\beta_i}$.

$\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. Given a public key $PK = (\mathcal{G}, H, h, u)$, an identifier id, a signature σ, and a vector $\mathbf{y} \in \mathbb{F}_p^N$, define

$$\gamma_1(PK, \sigma) \stackrel{\text{def}}{=} e\,(\sigma, h) \qquad \text{and} \qquad \gamma_2(PK, \mathsf{id}, \mathbf{y}) \stackrel{\text{def}}{=} e\left(\prod_{i=1}^{N} H(\mathsf{id}, i)^{y_i},\ u \right).$$

If $\gamma_1(PK, \sigma) = \gamma_2(PK, \mathsf{id}, \mathbf{y})$ this algorithm outputs 1; otherwise it outputs 0.

Due to lack of space, we omit the (straightforward) proof of correctness.

A signature is just a single element of $\mathbb{G}_1$. Groups $\mathbb{G}_1$ whose elements can be represented using $\log_2 p$ bits can be obtained by using pairing-friendly elliptic curves of prime or near-prime order; see [10] for further details.

A variant of the above scheme is more efficient when only properly augmented vectors will be signed (as is the case for applications to network coding), and the dimension m of the resulting vector space is known at the time any vector is signed or verified. In this setting, the signer can choose random $g_1, \ldots, g_N \in \mathbb{G}_1$ at the time of key generation and publish these as part of the public key. To sign a vector $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{F}_p^N$ using the identifier id, the signer sets $n := N - m$ and computes

$$\sigma := \left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{v_{n+i}} \prod_{j=1}^{n} g_j^{v_j} \right)^{\alpha} .$$

(Verification is changed in the obvious way.) In this variant, signing the augmented vector $\mathbf{v}$ is dominated by computing a single hash into $\mathbb{G}_1$ (taking time similar to a full exponentiation) plus n additional exponentiations in $\mathbb{G}_1$. The public key in this variant can be compressed to size linear in the security parameter k by generating $g_1, \ldots, g_N$ as the output of an independent hash function H' (also modeled as a random oracle).

We now prove the security of signature scheme NCS_0. More precisely, we consider the variant scheme (call it NCS_1) suggested above, and prove security of the network coding signature scheme NCS_1' constructed from NCS_1 as described in the optimization following Lemma 3. Our security proof assumes that only properly augmented vector spaces are signed. (We stress that NCS_0 itself is also secure, even without this assumption.)

Theorem 6. *Let* NCS_1' *be the network coding signature scheme constructed from* NCS_1 *via the (optimized) method of Lemma 3. Then* NCS_1' *is secure in the random oracle model assuming that the co-CDH problem in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is infeasible.*

In particular, let $\mathcal{A}$ be a polynomial-time adversary as in Definition 4. Then there exists a polynomial-time algorithm $\mathcal{B}$ that computes co-CDH in $(\mathbb{G}_1, \mathbb{G}_2)$, and such that

$$\text{co-CDH-Adv}[\mathcal{B}, (\mathbb{G}_1, \mathbb{G}_2)] \geq \text{NC-Adv}[\mathcal{A}, \text{NCS}_1'] - \frac{1}{p} - \frac{q_s(q_s + q_h)}{2^k},$$

where q_s and q_h are the number of signature and hash queries made by $\mathcal{A}$.

Proof. Let $\mathcal{A}$ be an adversary as in Definition 4, though recall we make the assumption that $\mathcal{A}$ only requests signatures on properly augmented vector spaces. We construct $\mathcal{B}$ that takes as input parameters $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \varphi)$ and inputs $g \in \mathbb{G}_1$ and $h, z \in \mathbb{G}_2$, with $z = h^x$, and outputs an element $\omega \in \mathbb{G}_1$. Algorithm $\mathcal{B}$ simulates the hash function H and the Setup and Sign algorithms of NCS_1', and works as follows.

Setup. $\mathcal{A}$ chooses an integer N, and $\mathcal{B}$ does the following:

1. Choose random $s_1, t_1, \ldots, s_N, t_N \in \mathbb{F}_p$, and set $g_i := g^{s_i}\varphi(h)^{t_i}$ for all i.
2. Output the public key $PK := (\mathcal{G}, H, g_1, \ldots, g_N, h, z)$.

Hash query. When $\mathcal{A}$ requests the value of $H(\mathsf{id}, i)$, algorithm $\mathcal{B}$ does:

1. If (id, i) has already been queried, return $H(\mathsf{id}, i)$.
2. If (id, i) has not been queried, choose $\varsigma_i, \tau_i \leftarrow \mathbb{F}_p$ and set $H(\mathsf{id}, i) := g^{\varsigma_i}\varphi(h)^{\tau_i}$.

Sign. When $\mathcal{A}$ requests a signature on a vector space $V \subset \mathbb{F}_p^N$, described by properly augmented basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_p^N$ (where $\mathbf{v}_i = (v_{i1}, \ldots, v_{iN})$), algorithm $\mathcal{B}$ does the following:

1. Choose a random $\mathsf{id} \leftarrow \{0,1\}^k$. If $H(\mathsf{id}, i)$ has already been queried for some $i \in \{1, \ldots, N\}$ then abort. (The simulation has failed.)
2. Set $n := N - m$ and compute $\varsigma_i := -\sum_{j=1}^{n} s_j v_{ij}$ for $i = 1, \ldots, m$.
3. Choose $\tau_i \leftarrow \mathbb{F}_p$ and set $H(\mathsf{id}, i) := g^{\varsigma_i}\varphi(h)^{\tau_i}$ for $i = 1, \ldots, m$. Set $\mathbf{t} := (t_1, \ldots, t_n, \tau_1, \ldots, \tau_m)$.
4. Compute $\sigma_i := \varphi(z)^{\mathbf{v}_i \cdot \mathbf{t}}$ for $i = 1, \ldots, m$.
5. Output id and $\sigma := (\sigma_1, \ldots, \sigma_m)$.

Output. If $\mathcal{B}$ does not abort, then eventually $\mathcal{A}$ outputs a signature $\sigma = (\sigma_1, \ldots, \sigma_m)$, an identifier id, and a non-zero vector $\mathbf{y}$.

1. If id is not one of the identifiers used to answer a previous signature query, then compute $H(\mathsf{id}, i)$ as above for $i = 1, \ldots, m$. Thus, in any case, $H(\mathsf{id}, i) = g^{\varsigma_i}\varphi(h)^{\tau_i}$ for ς_i, τ_i known to $\mathcal{B}$.
2. Set $\mathbf{s} := (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ and $\mathbf{t} := (t_1, \ldots, t_n, \tau_1, \ldots, \tau_m)$. If $\mathbf{s} \cdot \mathbf{y} = 0$ then abort.
3. Set $n := N - m$ and output $\omega := \left(\dfrac{\prod_{i=1}^{m} \sigma_i^{y_{n+i}}}{\varphi(z)^{\mathbf{t} \cdot \mathbf{y}}} \right)^{1/(\mathbf{s} \cdot \mathbf{y})}$.

We first observe that the responses to all hash queries are independent and uniformly random in $\mathbb{G}_1$. We also observe that the $g_1, \ldots, g_N$ are random group elements, and thus the public key PK output by $\mathcal{B}$ is distributed identically to the public key produced by the real Setup algorithm.

Next we show that the signatures σ output by $\mathcal{B}$ are identical to the signatures that would be produced by the real Sign algorithm given the public key PK and the hash answers computed by $\mathcal{B}$. Since the secret key corresponding to PK is x, it suffices to show that for each vector $\mathbf{v}$ "signed" by $\mathcal{B}$, we have

$$\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{v_{n+i}} \prod_{j=1}^{n} g_j^{v_j} \right)^{x} = \varphi(z)^{\mathbf{v} \cdot \mathbf{t}}, \tag{3}$$

where the left-hand side is the "real" signature and the right-hand side is the signature computed by $\mathcal{B}$. The left-hand side is equal to

$$\left(\prod_{i=1}^{m} \left(g^{\varsigma_i} \varphi(h)^{\tau_i}\right)^{v_{n+i}} \prod_{j=1}^{n} \left(g^{s_j} \varphi(h)^{t_j}\right)^{v_j} \right)^{x} = \left(g^{\mathbf{s} \cdot \mathbf{v}} \varphi(h)^{\mathbf{t} \cdot \mathbf{v}}\right)^{x}. \tag{4}$$

Now observe that we constructed $\mathbf{s}$ so that $\mathbf{s} \in V^{\perp}$ (i.e., $\mathbf{s} \cdot \mathbf{v} = 0$ for all $\mathbf{v} \in V$), so the expression (4) is equal to $\varphi(h)^{x(\mathbf{t} \cdot \mathbf{v})}$. Equation (3) now follows from the fact that $\varphi(z) = \varphi(h)^{x}$.

We next analyze the probability that $\mathcal{B}$ aborts while interacting with $\mathcal{A}$. There are two scenarios in which this can happen: if $\mathcal{B}$ responds to two different signature queries by choosing the same identifier id, or if $\mathcal{B}$ responds to a signature query by choosing an identifier id such that $\mathcal{A}$ has already requested the value of $H(\mathsf{id}, i)$ for some i. The probability of the first event is at most $q_s^2/2^k$, while the probability of the second event is at most $q_s q_h/2^k$.

Suppose $\mathcal{B}$ does not abort and $\mathcal{A}$ outputs a signature σ, an identifier id, and a non-zero vector $\mathbf{y}$. Let $\sigma = (\sigma_1, \ldots, \sigma_m)$. If $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$ then

$$e\left(\prod_{i=1}^{m} \sigma_i^{y_{n+i}}, \, h \right) = e\left(\prod_{i=1}^{m} H(\mathsf{id}, i)^{y_{n+i}} \prod_{j=1}^{n} g_j^{y_j}, \, z \right).$$

By the same reasoning as above the right-hand side is equal to

$$e\left(g^{\mathbf{s} \cdot \mathbf{y}} \varphi(h)^{\mathbf{t} \cdot \mathbf{y}}, z\right) = e\left(g^{x(\mathbf{s} \cdot \mathbf{y})} \varphi(z)^{\mathbf{t} \cdot \mathbf{y}}, h\right),$$

where $\mathbf{s}$ and $\mathbf{t}$ are determined from id as in steps (1) and (2) of $\mathcal{B}$'s output procedure. The non-degeneracy of e then implies that

$$\prod_{i=1}^{m} \sigma_i^{y_{n+i}} = g^{x(\mathbf{s} \cdot \mathbf{y})} \varphi(z)^{\mathbf{t} \cdot \mathbf{y}}.$$

It follows that if $\mathbf{s} \cdot \mathbf{y} \neq 0$ then the element ω output by $\mathcal{B}$ is equal to g^{x}.

To complete the proof, we show that $\mathbf{s} \cdot \mathbf{y} = 0$ with probability $1/p$. In preparation, we first prove the following lemma:

Lemma 7. *Assume $\mathcal{B}$ does not abort. Then the variables $s_1, \ldots, s_N$ are each independently uniform in $\mathbb{F}_p$ even conditioned on the view of $\mathcal{A}$.*

Proof. In proving the lemma we can ignore any queries $H(\mathsf{id}, i)$ where id is not an identifier used to respond to a signing query, since (a) the variables $s_1, \ldots, s_N$ are not involved in these queries, and (b) the variables that are involved in these queries are not involved in any other interaction between $\mathcal{A}$ and $\mathcal{B}$.

We show that for any given view of $\mathcal{A}$ and any choice of values for $s_1, \ldots, s_N$, there is a unique choice of values for all of the other variables in the system that is consistent with the adversary's view. The adversary's view consists of the public key PK and the signatures on subspaces V_k for $k = 1, \ldots, q_s$. Let $m_k = \dim V_k$. Hence, the adversary's view is derived from $2N + \sum 2m_k$ random variables:

- The public key is derived from s_j, t_j for $j = 1, \ldots, N$.
- The kth signature is derived from the s_j, t_j and ς_i, τ_i for $i = 1, \ldots, m_k$. (Here we use the fact that $\mathcal{B}$ did not abort, and so no two signing queries use the same value of id.)

Moreover, the adversary has $N + \sum 3m_k$ linear relations on these variables:

- N relations derived from the public key,
- m_k relations derived from the values of $H(\mathsf{id}, i)$ for the kth query,
- m_k relations derived from the signature $(\sigma_1, \ldots, \sigma_{m_k})$ for the kth query,
- m_k relations derived from the fact that $\mathbf{s} \in V^\perp$ for the kth query.

We set the following notation:

$$\begin{aligned}
\mathbf{s}^L &= (s_1, \ldots, s_N) \\
\mathbf{s}^R_k &= (\varsigma_1, \ldots, \varsigma_{m_k}) \text{ for the } k\text{th signature query} \\
\mathbf{t}^L &= (t_1, \ldots, t_N) \\
\mathbf{t}^R_k &= (\tau_1, \ldots, \tau_{m_k}) \text{ for the } k\text{th signature query.}
\end{aligned}$$

Let $\bar{V}_k$ be the $m_k \times N$ matrix whose ith row consists of the first $N - m_k$ entries (i.e., the unaugmented part) of the basis vector $\mathbf{v}_i$ for the kth query, followed by m_k zeroes. Let $\varphi(h) = g^\alpha$. The view of $\mathcal{A}$ imposes the following constraints:

$$\begin{aligned}
\mathbf{s}^L + \alpha \mathbf{t}^L &= \mathbf{c}_1 && \text{(public key)} && (5) \\
\mathbf{s}^R_k + \alpha \mathbf{t}^R_k &= \mathbf{c}_{2,k} && \text{(values of } H(\mathsf{id}, i)) && (6) \\
\bar{V}_k \mathbf{t}^L + \mathbf{t}^R_k &= \mathbf{c}_{3,k} && \text{(signatures)} && (7) \\
\bar{V}_k \mathbf{s}^L + \mathbf{s}^R_k &= \mathbf{0} && (\mathbf{s} \in V^\perp) && (8)
\end{aligned}$$

for some vectors $\mathbf{c}_1 \in \mathbb{F}_p^N$, $\mathbf{c}_{2,k} \in \mathbb{F}_p^{m_k}$, $\mathbf{c}_{3,k} \in \mathbb{F}_p^{m_k}$ that are determined (in an information-theoretic sense) from the view of $\mathcal{A}$. We wish to show that the system has a unique solution for any value of $\mathbf{s}^L$.

Observe that equation (8) is linearly dependent on equations (5), (6), and (7). Specifically, for each k we have $(8) = \bar{V}_k(5) + (6) - \alpha(7)$. Since the system has at least one solution by construction, any choice of variables satisfying equations (5)–(7) must also satisfy (8).

Suppose $\mathbf{s}^L$ is fixed; then equation (5) determines a unique value for $\mathbf{t}^L$. For each k, equation (7) and this value of $\mathbf{t}^L$ determine a unique value for $\mathbf{t}_k^R$, from which equation (6) determines a unique value of $\mathbf{s}_k^R$. Thus for any value of $\mathbf{s}^L$ there is a unique solution to equations (5)–(8). We conclude that $\mathbf{s}^L = (s_1, \ldots, s_N)$ is uniform in $\mathbb{F}_p^N$ even conditioned on the view of $\mathcal{A}$. □

To complete the proof of Theorem 6, we now show that $\mathbf{s} \cdot \mathbf{y} = 0$ with probability $1/p$. If $\mathcal{A}$ outputs a type 1 forgery, then id was not used in response to any previous signing query. In this case the $\{\varsigma_i\}$ are independently uniform in $\mathbb{F}_p$ even conditioned on the view of $\mathcal{A}$. By Lemma 7 the $\{s_i\}$ are also independently uniform in $\mathbb{F}_p$ conditioned on $\mathcal{A}$'s view. Since $\mathbf{s} = (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ and $\mathbf{y}$ is non-zero, it follows that $\mathbf{s} \cdot \mathbf{y}$ is uniformly distributed in $\mathbb{F}_p$, and thus the probability that $\mathbf{s} \cdot \mathbf{y} = 0$ is $1/p$.

Now suppose that $\mathcal{A}$ outputs a type 2 forgery, so id was used in response to the signing query for some vector subspace V, and $\mathbf{y} \notin V$. (Note that id was used to answer only one signing query, otherwise $\mathcal{B}$ aborts.) By Lemma 7 the variables $\{s_i\}$ are independently uniform in $\mathbb{F}_p$ even conditioned on the adversary's view. This implies that, conditioned on the adversary's view, the vector $\mathbf{s} = (s_1, \ldots, s_n, \varsigma_1, \ldots, \varsigma_m)$ is uniformly random in $V^\perp$. So for any $\mathbf{y} \notin V$ we see that $\mathbf{s} \cdot \mathbf{y}$ is uniform in $\mathbb{F}_p$, and we conclude that $\mathbf{s} \cdot \mathbf{y} = 0$ with probability $1/p$. This completes the proof. □

5 Network Coding Signatures without Random Oracles

Krohn et al. [22] propose authenticating network coding data using a homomorphic hash function (see below). As in Definition 2, their scheme produces a signature σ_i on each basis vector of the subspace to be authenticated. Their system is not secure according to our definition, however, as there is no mechanism to ensure that basis vectors from different files cannot be combined. Our solution is to authenticate all the hash values (along with the file identifier) using a standard signature scheme, which we denote by $\mathcal{S}_0 = (\mathsf{Gen}_0, \mathsf{Sign}_0, \mathsf{Verify}_0)$. This modification produces a secure network coding signature scheme (as in Definition 1) but eliminates the homomorphic property. The scheme can thus be used to sign entire subspaces, but not individual vectors.

Signature Scheme NCS$_2$

$\mathsf{Setup}(1^k, N)$. Given a security parameter 1^k and a positive integer N do:

1. Choose a group $\mathbb{G}$ of prime order $p > 2^k$.
2. Run $\mathsf{Gen}_0(1^k)$ and let the public/private keys be PK_0, SK_0.
3. Choose generators $g_1, \ldots, g_N \leftarrow \mathbb{G} \setminus \{1\}$.
4. Output the prime p, the public key $PK := (\mathbb{G}, g_1, \ldots, g_N, PK_0)$, and the private key $SK := SK_0$.

$\mathsf{Sign}(SK, \mathsf{id}, V)$. Given a secret key SK, a file identifier id, and an m-dimensional subspace $V \subset \mathbb{F}_p^N$ described by a properly augmented basis $\mathbf{v}_1, \ldots, \mathbf{v}_m$, do:

1. Set $n := N - m$. Compute $\sigma_i := \prod_{j=1}^{n} g_j^{-v_{ij}}$ for $i = 1, \ldots, m$.
2. Compute $\tau \leftarrow \mathsf{Sign}_0(SK, (\mathsf{id}, \sigma_1, \ldots, \sigma_m))$.
3. Output $\sigma := (\sigma_1, \ldots, \sigma_m, \tau)$.

$\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. Given a public key $PK = (\mathbb{G}, g_1, \ldots, g_N, PK_0)$, an identifier id, a signature $\sigma = (\sigma_1, \ldots, \sigma_m, \tau)$, and a vector $\mathbf{y} \in \mathbb{F}_p^N$, do:

1. Run $\mathsf{Verify}_0(PK_0, (\mathsf{id}, \sigma_1, \ldots, \sigma_m), \tau)$. If the answer is 0, output 0.
2. If $\left(\prod_{j=1}^{n} g_j^{y_j}\right)\left(\prod_{i=1}^{m} \sigma_i^{y_{n+i}}\right) = 1$ then output 1; otherwise output 0.

We omit the (straightforward) proof of correctness.

If elements of $\mathbb{G}$ are represented using $\log_2 p$ bits and the signature scheme $\mathcal{S}_0$ produces signatures of size $\log_2 p$, then the size of the signature σ is $(m+1)\log_2 p$ bits. If one is willing to use the random oracle model, we can achieve a constant-size public key by letting the values $g_1, \ldots, g_N$ be computed as the output of a hash function H (viewed as a random oracle).

Theorem 8. *Assume $\mathcal{S}_0$ is a secure signature scheme. Then* NCS_2 *is secure assuming hardness of the discrete logarithm problem in $\mathbb{G}$.*

In particular, let $\mathcal{A}$ be a polynomial-time adversary as in Definition 4. Then there exists a polynomial-time adversary $\mathcal{B}_1$ that forges signatures for $\mathcal{S}_0$ and a polynomial-time algorithm $\mathcal{B}_2$ that computes discrete logarithms, such that

$$\text{Sig-Adv}[\mathcal{B}_1, \mathcal{S}_0] + 2 \cdot \text{DL-Adv}[\mathcal{B}_2, \mathbb{G}] \geq \text{NC-Adv}[\mathcal{A}, \mathrm{NCS}_2] - \frac{q_s^2}{2^k},$$

where q_s denotes the number of signature queries made by $\mathcal{A}$, and $\text{Sig-Adv}[\mathcal{B}_1, \mathcal{S}_0]$ *is the probability that $\mathcal{B}_1$ wins the security game for the standard signature scheme $\mathcal{S}_0$ (see [20, §12.2]).*

Proof. Suppose algorithm $\mathcal{A}$ produces a signature $\sigma = (\sigma_1, \ldots, \sigma_m, \tau)$, an identifier id, and a vector $\mathbf{y}$ such that $\mathsf{Verify}(PK, \mathsf{id}, \sigma, \mathbf{y}) = 1$. If id is not one of the identifiers returned on a signature query (type 1 forgery), then $\mathcal{A}$ has forged a signature relative to $\mathcal{S}_0$. In case of a type 2 forgery, say id was used in response to a unique signature query on the vector space V, and that $\mathbf{y} \notin V$. Define H via $H(\mathbf{v}) = \prod_{j=1}^{n} g_j^{v_j}$. Since $\mathbf{y} \notin V$ we have $\mathbf{y}' := \sum_{i=1}^{m} y_{n+i}\mathbf{v}_i \neq \mathbf{y}$. The fact that the signature verifies implies that $H(\mathbf{y}) = H(\mathbf{y}')$, and thus we have produced a collision for $H(\cdot)$. By standard arguments [5,2], an algorithm $\mathcal{A}$ that produces such a collision with probability ε can be used to compute discrete logarithms in $\mathbb{G}$ with probability at least $\varepsilon/2$. □

Relation with [25]. Signature Scheme NCS_2 can also be viewed as a secure instantiation of the signature scheme proposed by Zhao et al. [25]. Their scheme computes a signature on V by choosing a random vector $\mathbf{u} \in V^\perp$ and outputting $(h_1, \ldots, h_N) = (g^{u_1}, \ldots, g^{u_N})$ along with a signature on this tuple. Verification of $\mathbf{y}$ involves checking whether $\prod_{j=1}^{N} h_j^{y_j} = g^{\mathbf{u}\cdot\mathbf{y}} = 1$. Correctness follows from the fact that $\mathbf{u} \in V^{\perp}$, and security follows from the fact that computing $\mathbf{y} \notin V$ such that $\mathbf{u} \cdot \mathbf{y} = 0$ permits computation of discrete logarithms in $\mathbb{G}$ (see [25, Theorem 1] or [3, Lemma 3.2]).

To view the scheme NCS_2 from this perspective, fix a generator g of $\mathbb{G}$ and write the first $n = N - m$ elements of the public key as $g^{u_1}, \dots, g^{u_n}$. Letting $u_{n+i} = \sum_{j=1}^{n} -u_j v_{ij}$, we have $\sigma_i = g^{u_{n+i}}$. Furthermore, if $\{\mathbf{v}_i\}_{i=1}^{m}$ is a properly augmented basis of V then the vector $\mathbf{u} = (u_1, \dots, u_N)$ is in $V^{\perp}$. The verification step then computes $g^{\mathbf{u}\cdot\mathbf{y}}$ just as in the scheme of Zhao et al.

The signatures produced by scheme NCS_2 have length $\Theta(m)$ and are thus much shorter than the signatures produced by the scheme of Zhao et al., which have length $\Theta(N)$.

6 A Lower Bound on Signature Size

We now prove a lower bound on the length of signatures for linear subspaces. Our lower bound applies to network coding signature schemes (Definition 1) that have the following two properties:

Additivity: For any PK, id, σ, and vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}_p^N$, if $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{u}, \sigma) = \mathsf{Verify}(PK, \mathsf{id}, \mathbf{v}, \sigma) = 1$ then $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{u}+\mathbf{v}, \sigma) = 1$. Both our constructions are additive.

Fixed size: For a given $m > 0$ and a given SK, the size of the signature output by $\mathsf{Sign}(SK, \mathsf{id}, V)$ is the same for all identifiers id and m-dimensional spaces $V \subset \mathbb{F}_p^N$. Again, this holds for both of the systems in this paper. We make this assumption primarily to simplify the presentation; a version of our bound holds even if this property is not satisfied.

For a secret key SK and integers N, m let $\ell_{SK,N,m}$ be the length in bits of signatures $\mathsf{Sign}(SK, \mathsf{id}, V)$ where V is an m-dimensional subspace of $\mathbb{F}_p^N$.

For signatures that have these two properties, we show that the signature size must be at least (roughly) $m \log_2 p$ bits. In particular, we construct an attack algorithm that forges signatures whenever $\ell_{SK,N,m}$ is shorter than this bound. Hence, if the scheme is to be secure then for almost all secret keys the signature size must be greater than our bound.

The intuition behind our lower bound is that if signatures are short, then by the pigeonhole principle there is a large set $\mathcal{V}$ of linear subspaces that all have the same signature σ. If signatures are sufficiently short, then the direct sum of the spaces in $\mathcal{V}$ spans all of $\mathbb{F}_p^N$. Since the signature scheme is additive this implies that $\mathsf{Verify}(PK, \mathsf{id}, \sigma, \mathbf{y}) = 1$ for *any* $\mathbf{y} \in \mathbb{F}_p^N$; we will call a signature σ with this property (for a fixed identifier id) *trivial*. We conclude that there are many subspaces V with trivial signatures; the system can then be easily attacked by choosing a random subspace V, obtaining a signature on V, and producing a vector $\mathbf{y} \notin V$.

Theorem 9. *Let m, N be integers with $0 < m < N$, and let* $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ *be a network coding signature scheme satisfying the two properties above. Then there is a polynomial-time adversary $\mathcal{A}$ making a single signature query and such that the following holds: when the secret key SK used in the security game of Definition 4 satisfies*

$$\ell_{SK,N,m} \leq m \log_2 p - 4m/p - 1, \tag{9}$$

then $\mathcal{A}$ wins with probability at least $1/2$.

Proof. Fix a public/private key pair PK, SK. When the adversary queries a vector space V to the challenger, the challenger produces an identifier id uniformly at random from the space $\mathcal{I}$ of identifiers; in particular, id is independent of V. We may thus fix the randomness of the challenger in advance and let id_1 be the identifier produced on the first query. Although the Sign algorithm may be probabilistic, once we have fixed the randomness each m-dimensional subspace $V \subset \mathbb{F}_{p^N}$ is mapped to a *unique* signature $\sigma := \mathsf{Sign}(SK, \mathsf{id}_1, V)$.

We now proceed with a combinatorial argument. Let $n = N - m$. The number of m-dimensional subspaces $V \subset \mathbb{F}_p^{n+m}$ is the p-binomial coefficient [24, Proposition 1.3.18]

$$\binom{n+m}{m}_p = \frac{(p^{n+m}-1)(p^{n+m-1}-1)\cdots(p^{n+1}-1)}{(p^m-1)(p^{m-1}-1)\cdots(p-1)} > p^{mn}.$$

Let $\mathcal{U}$ be the set of vector spaces V such that the signature on V is nontrivial, and let β be the fraction of vector spaces V with nontrivial signatures; then the cardinality of $\mathcal{U}$ is at least $p^{mn}\beta$. Let α be the number of distinct nontrivial signatures produced by signing all vector spaces $V \in \mathcal{U}$ with identifier id_1. Then by the pigeonhole principle, there is a set of vector spaces $\mathcal{V} \subseteq \mathcal{U}$ of cardinality at least $p^{mn}\beta/\alpha$ such that the signatures $\mathsf{Sign}(SK, \mathsf{id}_1, V)$ are identical for all $V \in \mathcal{V}$. Call this signature σ.

Let $W \subseteq \mathbb{F}_p^{n+m}$ be the direct sum of all the spaces in $\mathcal{V}$. Since the signature system is additive, we know that $\mathsf{Verify}(PK, \mathsf{id}_1, \mathbf{w}, \sigma) = 1$ for all $\mathbf{w} \in W$. If $W = \mathbb{F}_p^{n+m}$ then σ is trivial, contradicting the assumption that $\mathcal{V} \subseteq \mathcal{U}$. Thus W is a subspace of $\mathbb{F}_p^{n+m}$ of dimension at most $n + m - 1$. Then the number of m-dimensional subspaces V contained in W is at most $\binom{n+m-1}{m}_p < p^{m(n-1)}(1 + 2/p)^m$, and we have

$$p^{mn}\left(\frac{\beta}{\alpha}\right) \leq \#\mathcal{V} < p^{m(n-1)}\left(1 + \frac{2}{p}\right)^m. \tag{10}$$

Now suppose that for the key pair PK, SK the quantity $\ell_{SK,N,m}$ satisfies (9). Then the number α of distinct nontrivial signatures satisfies

$$\alpha \leq p^m \cdot 2^{-4m/p}\left(\frac{1}{2}\right) < p^m\left(1 + \frac{2}{p}\right)^{-m}\left(\frac{1}{2}\right). \tag{11}$$

where the second inequality follows from $2^{2x} > 1 + x$ for $x > 0$. Combining inequalities (10) and (11), we see that the fraction β of subspaces with nontrivial signatures satisfies

$$\beta < \frac{\alpha}{p^m} \cdot \left(1 + \frac{2}{p}\right)^m < \frac{1}{2}. \tag{12}$$

Now adversary $\mathcal{A}$ works as follows: it chooses at random a vector space $V \subset \mathbb{F}_p^{n+m}$ and obtains id_1 and $\sigma := \mathsf{Sign}(SK, \mathsf{id}_1, V)$ from the signing oracle. The adversary then computes a vector $\mathbf{y} \notin V$ and outputs $(\mathsf{id}_1, \sigma, \mathbf{y})$ as the forgery. By (12) the probability that σ is trivial is at least 1/2, and if this is the case then $\mathsf{Verify}(PK, \mathsf{id}_1, \mathbf{y}, \sigma) = 1$. Hence, $\mathcal{A}$ has advantage (as in Definition 4) at least 1/2 while making a single signature query. □

7 Conclusion and Extensions

We studied the problem of signing a subspace $V \subset \mathbb{F}_p^N$ in a manner that authenticates all vectors in V. The question is motivated by the need to provide integrity when using network coding. We defined the problem and described two secure schemes. Our first scheme is homomorphic and has constant-size public keys and per-vector signatures; its security is based on the co-CDH assumption in the random oracle model. Our second scheme offers security based on the weaker discrete logarithm assumption without random oracles. In both schemes, a single public key can be used to sign many linear spaces (i.e., files). We also proved a lower bound on the length of signatures for linear subspaces, and observe that both our systems meet the lower bound.

In network coding applications, one may wish to vary the dimension of the ambient space $\mathbb{F}_p^N$. Our definitions assume that the dimension of the ambient space is fixed. However, our systems can easily be adapted to sign subspaces V_i contained in varying ambient spaces $\mathbb{F}_p^{N_i}$ with a single public key by incorporating the dimension N_i into the hash function (for scheme NCS_1) or the outer signature with respect to $\mathcal{S}_0$ (for scheme NCS_2).

Note: The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

1. Ahlswede, R., Cai, N., Li, S., Yeung, R.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
2. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The case of hashing and signing. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg (1994)
3. Boneh, D., Franklin, M.: An efficient public key traitor tracing scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)

4. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
5. Brands, S.: An efficient off-line electronic cash system based on the representation problem, CWI Technical Report CS-R9323 (1993)
6. Byers, J.W., Luby, M., Mitzenmacher, M., Rege, A.: A digital fountain approach to reliable distribution of bulk data. In: ACM SIGCOMM (1998)
7. Charles, D., Jain, K., Lauter, K.: Signatures for network coding. In: 40th Annual Conference on Information Sciences and Systems, CISS 2006 (2006)
8. Chou, P.A., Wu, Y., Jain, K.: Practical network coding. In: 41st Allerton Conference on Communication, Control, and Computing (2003)
9. Duquesne, S., Frey, G.: Background on pairings. In: Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman & Hall/CRC Press, Boca Raton (2006)
10. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372 (2006), http://eprint.iacr.org/
11. Gkantsidis, C., Rodriguez, P.: Network coding for large scale content distribution. In: IEEE INFOCOM (2005)
12. Gkantsidis, C., Rodriguez, P.: Cooperative security for network coding file distribution. In: IEEE INFOCOM (2006)
13. Han, K., Ho, T., Koetter, R., Médard, M., Zhao, F.: On network coding for security. In: IEEE MILCOM (2007)
14. Ho, T., Koetter, R., Médard, M., Karger, D., Effros, M.: The benefits of coding over routing in a randomized setting. In: Proc. International Symposium on Information Theory (ISIT) (2003)
15. Ho, T., Leong, B., Koetter, R., Médard, M., Effros, M., Karger, D.: Byzantine modification detection in multicast networks using randomized network coding. In: Proc. Intl. Symposium on Information Theory (ISIT), pp. 144–152 (2004)
16. Ho, T., Médard, M., Koetter, R., Karger, D.R., Effros, M., Shi, J., Leong, B.: A random linear network coding approach to multicast. IEEE Trans. Inform. Theory 52(10), 4413–4430 (2006)
17. Jaggi, S.: Design and Analysis of Network Codes. PhD thesis, California Institute of Technology (2006)
18. Jaggi, S., Langberg, M., Katti, S., Ho, T., Katabi, D., Médard, M., Effros, M.: Resilient network coding in the presence of Byzantine adversaries. IEEE Trans. on Information Theory 54(6), 2596–2603 (2008)
19. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
20. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC Press, Boca Raton (2007)
21. Kim, M., Médard, M., Barros, J.: Counteracting Byzantine adversaries with network coding: An overhead analysis (2008), http://arxiv.org/abs/0806.4451
22. Krohn, M., Freedman, M., Mazieres, D.: On the-fly verification of rateless erasure codes for efficient content distribution. In: Proc. of IEEE Symposium on Security and Privacy, pp. 226–240 (2004)
23. Li, S.-Y.R., Yeung, R.W., Cai, N.: Linear network coding. IEEE Trans. Info. Theory 49(2), 371–381 (2003)
24. Stanley, R.: Enumerative Combinatorics, vol. 1. Cambridge University Press, Cambridge (1997)
25. Zhao, F., Kalker, T., Médard, M., Han, K.: Signatures for content distribution with network coding. In: Proc. Intl. Symp. Info. Theory (ISIT) (2007)

Improving the Boneh-Franklin Traitor Tracing Scheme

Pascal Junod[1,2], Alexandre Karlov[1,3], and Arjen K. Lenstra[3,4]

[1] Nagravision SA, Cheseaux-sur-Lausanne, Switzerland
[2] University of Applied Sciences Western Switzerland, Yverdon-les-Bains, Switzerland
[3] EPFL IC LACAL, Station 14, Lausanne, Switzerland
[4] Alcatel-Lucent Bell Laboratories, USA

Abstract. Traitor tracing schemes are cryptographically secure broadcast methods that allow identification of conspirators: if a pirate key is generated by k traitors out of a static set of ℓ legitimate users, then all traitors can be identified given the pirate key. In this paper we address three practicality and security issues of the Boneh-Franklin traitor-tracing scheme. In the first place, without changing the original scheme, we modify its tracing procedure in the non-black-box model such that it allows identification of k traitors in time $\tilde{O}(k^2)$, as opposed to the original tracing complexity $\tilde{O}(\ell)$. This new tracing procedure works independently of the nature of the Reed-Solomon code used to watermark private keys. As a consequence, in applications with billions of users it takes just a few minutes on a common desktop computer to identify large collusions. Secondly, we exhibit the lack of practical value of list-decoding algorithms to identify more than k traitors. Finally, we show that $2k$ traitors can derive the keys of *all* legitimate users and we propose a fix to this security issue.

Keywords: Boneh-Franklin traitor tracing, Reed-Solomon codes, Berlekamp-Massey algorithm, Guruswami-Sudan algorithm.

1 Introduction

Consider the following scenario: a center broadcasts data to ℓ receivers where only authorized users (typically, those who have paid a fee) should have access to the data. A way to realize this, widely deployed in commercial Pay-TV systems, is to encrypt the data using a symmetric key and to securely transmit to each authorized receiver this key which will be stored in a tamper-proof piece of hardware, like a smart card.

Unfortunately, tamper-resistant hardware is very difficult and costly to design, since it is vulnerable to a wide variety of attacks (see [1,27] as two good starting points). As a result, a malicious user (hereafter called a *traitor*) can attempt to retrieve the decryption key from his receiver and, if successful, distribute it (sell or give away) to unauthorized users (the *pirates*). Depending on the nature of the encryption schemes in use, we can even imagine situations where a dishonest

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 88–104, 2009.

user will try to *mix* several legitimate keys in order to build a new one and embed it in a pirate receiver.

The problem of identifying which receivers were compromised or which secret keys have leaked is called *traitor tracing*. Usually, two modes of traitor tracing are considered: in the *black-box* model, the tracing algorithm sends crafty ciphertexts to the pirate receiver and aims at determining which keys it uses while observing its behavior; in the *non-black-box model*, we assume that the keys can be extracted from the pirate receiver and are known to the tracing algorithm. The black-box model is widely considered by the cryptographic community as being a standard security model for evaluating traitor-tracing schemes security. However, based on our practical experience, we know that it is reasonable to assume that a tracing authority has at least the same technological and financial resources to reverse-engineer a pirate receiver as a traitor had, to perform the same operation on a legitimate receiver.

1.1 Related Work

Fiat and Naor introduced the concept of *broadcast encryption* in [17]. In their model[1], there exists a set of ℓ authorized users and the broadcasting center can *dynamically* specify a privileged subset of authorized users that can decrypt selected ciphertexts (like high-value content, for instance). Later, Chor, Fiat, and Naor [12] introduced the concept of traitor-tracing to overcome decryption key piracy in broadcast encryption schemes. Their scheme (which was improved by Naor and Pinkas in [33, 13]) is *k-collusion resistant* (or k-resilient) in the sense that at least one traitor can be identified with high probability given a pirate key generated by up to k traitors. Naor, Naor and Lotspiech presented more efficient broadcast encryption schemes [32] with tracing capabilities; it was however demonstrated by Kiayias and Pehlivanoglu [21] that the iterative nature of the tracing procedure allows a pirate to significantly leverage the compromise of a few keys. Although broadcast encryption and traitor-tracing are orthogonal problems in nature, and thus frequently studied separately, they are in practice indivisible: some *trace-and-revoke* schemes have been proposed accordingly [15, 16], culminating in [9]. The latter scheme, though resistant to any collusion size, is geared towards small-scale systems and impractical for the systems of tens of millions of users that we are dealing with and that inspired this paper; this is mainly due to the $O(\sqrt{\ell})$ complexity of [9] in terms of key storage and bandwidth requirements. Additionally, the tracing costs are $O(\ell^2)$, which also severely limits its applicability.

Kurosawa and Desmedt [24] proposed a *public-key* traitor tracing scheme, which was later broken by Stinson and Wei [40]. Boneh and Shaw [8] discussed collusion-resistant schemes for fingerprinting digital data based on error-correcting codes. Boneh and Franklin [5] proposed a new public-key traitor-tracing scheme also

[1] Note that in this paper, we will only consider *stateless* receivers, i.e., receivers for which it is not possible to guarantee synchronism with the broadcast center and which are resettable. Broadcast encryption schemes for stateful receivers have been proposed in [44, 41].

based on error-correcting codes, more precisely on Reed-Solomon codes. Actually, the traitor-tracing problem can be interpreted as an application of watermarking to secret keys that are distributed among users. The Boneh-Franklin non-black-box traitor tracing scheme is k-collusion resistant and deterministic in the sense that all of the traitors are identified with probability 1 if at most k of them collude to derive new pirate keys. The fastest claimed running time of the non-black-box tracing algorithm is $O(\ell \log \ell \log \log \ell)$ while the best known black-box tracing method has an exponential complexity $O(\binom{\ell}{k} k^2)$. Kurosawa and Yoshida [25] have generalized the Kurosawa-Desmedt and Boneh-Franklin schemes. The technique used by Boneh and Franklin to watermark private keys has since been re-used by Kiayias and Yung [23] to design an *asymmetric*[2] public-key traitor tracing scheme; other examples of Reed-Solomon codes use include schemes designed by Dodis *et al.* [15, 16]. Recently, Boneh *et al.* [7] have presented a *fully-collusion resistant* traitor tracing scheme which has private keys of constant size and ciphertexts of size $O(\sqrt{\ell})$. Finally, the low efficiency of tracing procedures in traitor tracing schemes has been addressed by Silverberg *et al.* in [37,38]. The authors present several schemes based on algebraic codes which enable traitors to be traced in time polynomial in $k^2 \log \ell$. Recently, Billet and Phan [2] and Boneh and Naor [6] have independently proposed traitor-tracing schemes with constant size ciphertexts and having a black-box tracing complexity of $O(t^2 \ell \log \ell)$ and $O(t^4 \log \ell)$, respectively.

1.2 Our Contributions

While we agree that improving the exponential complexity of black-box tracing as cited above would be a very worthwhile cause to pursue, we choose to focus in this paper, in the light of the negative results obtained by Kiayias and Yung [22], on some security and efficiency issues that we encountered in practical applications of the Boneh-Franklin traitor-tracing scheme [5] in the *non-black-box model.* Although Boneh-Franklin traitor-tracing is one of the most elegant and efficient public-key traitor tracing schemes, it suffers from certain issues that limit its practical applicability in large-scale systems. We point out what the problems are and how they can be addressed. As usual, ℓ denotes the number of legitimate users and k the collusion threshold.

Complexity of Non-Black-Box Tracing. One of the issues is the complexity of the non-black-box traitor tracing procedure which depends on ℓ. This is a major drawback when applied to systems of many millions of users, since tracing would require large computational power, or could even be infeasible in practice. We dissect the way Reed-Solomon codes are used to watermark private keys, and we show that, contrary to what is suggested in [5], it is possible to trace in time[3] $\tilde{O}(k^2)$, i.e., with a complexity independent of ℓ, using the

[2] *Asymmetric* traitor tracing is a variant introduced by Pfitzmann [35] where the broadcasting center is not necessarily trusted, thus the tracing procedure must produce *undeniable* evidence of the implication of the traitor subscribers.

[3] Here, the $\tilde{O}(n)$ notation hides the terms which are poly-logarithmic in n.

Berlekamp-Massey algorithm instead of the Berlekamp-Welch algorithm. Although both algorithms require the same complexity to fully recover a noisy Reed-Solomon codeword, the complexity of the Berlekamp-Massey algorithm can be reduced if used for tracing only. The resulting new tracing procedure does not require any modification of the original Boneh-Franklin scheme. In practice, it takes us just a few minutes on a common desktop PC to trace large coalitions in systems having hundreds of millions of users. Our result improves the results obtained by Silverberg *et al.* [37, 38]. Our finding also applies to schemes using the same watermarking technique, such as the ones described in [23, 15, 16]. Another immediate benefit we identify is the possibility to use Reed-Solomon codes optimized specifically to allow faster decryption. In practice, for large systems and coalitions of medium size, we speed up the decryption by almost an order of magnitude.

Above-Threshold Tracing. Secondly, we raise an issue concerning the above-threshold security of the Boneh-Franklin scheme and its variants. We show that the list-decoding techniques, such as the Guruswami-Sudan algorithm, as advocated by Boneh-Franklin to trace more than k traitors, detect only a few additional traitors, and this at a high cost.

Beyond-Threshold Tracing. Finally, we show that if an adversary is able to recover $2k$ secret keys, then she is able to compute *any* other secret key, including the uncompromised ones. Thus, in this case the security of the system completely collapses. This somewhat embarrassing property is primarily due to the fact that the linear tracing code is public. We show how this issue can be addressed at the cost of keeping more than a single secret value in the receivers.

This paper is organized as follows. In §2 we review the Boneh-Franklin scheme [5]. Then, in §3, we speed up both its codeword generation and tracing procedures. In §4 we discuss the above-threshold tracing based on the Guruswami-Sudan list-decoding algorithm, while in §5 we study the security of the Boneh-Franklin scheme when the number of recovered secret keys is at least twice the allowed threshold.

2 Boneh-Franklin Scheme

This section describes the Boneh-Franklin traitor tracing scheme [5] by first defining its encryption and decryption procedures, then by explaining the codeword generation mechanism and finally by describing the underlying non-black-box tracing mechanism. We adopt the notation used in [5] denoting by ℓ the number of users in the system and by k the maximal coalition size. Hence, the described scheme is supposed to be secure against any collusion of at most k users.

2.1 Encryption/Decryption

Let $\mathbb{G}_q$ denote a group of prime order q in which the Decision Diffie-Hellman problem [4] is hard. Typically, $\mathbb{G}_q$ is a subgroup of order q of $\mathbb{Z}_p^*$, where p is prime and $q|p-1$; alternatively, $\mathbb{G}_q$ can be a group of points of an elliptic curve over a finite field.

The key generation process proceeds as follows. Let g be a generator of $\mathbb{G}_q$. For $1 \leq j \leq 2k$, let $r_j \in_R \mathbb{Z}/q\mathbb{Z}$ and compute $h_j = g^{r_j}$. The *public key* is defined as $\langle y, h_1, \ldots, h_{2k} \rangle \in \mathbb{G}_q^{2k+1}$ where $y = \prod_{j=1}^{2k} h_j^{\alpha_j} \in \mathbb{G}_q$ for random $\alpha_j \in_R \mathbb{Z}/q\mathbb{Z}$. Here, we say that the vector $\boldsymbol{\alpha} = \langle \alpha_1, \ldots, \alpha_{2k} \rangle$ is a *representation* of y with respect to the base $\langle h_1, \ldots, h_{2k} \rangle$. Note that if $\boldsymbol{\rho}^{(1)}, \ldots, \boldsymbol{\rho}^{(n)}$ are n representations of the same element of $\mathbb{G}_q$ with respect to the same base, then so is any convex combination $\sum_{i=1}^{n} \eta_i \boldsymbol{\rho}^{(i)}$ of the representations, where $\eta_i \in \mathbb{Z}/q\mathbb{Z}$ are scalars such that $\sum_{i=1}^{n} \eta_i = 1$.

Let $\Gamma = \{\boldsymbol{\gamma}^{(1)}, \ldots, \boldsymbol{\gamma}^{(\ell)}\}$ be a *linear space tracing code*, i.e., a collection of ℓ codewords $\boldsymbol{\gamma}^{(i)}$, for $1 \leq i \leq \ell$, where each $\boldsymbol{\gamma}^{(i)} = \langle \gamma_j^{(1)}, \ldots, \gamma_j^{(2k)} \rangle$ is a $2k$-dimensional vector over $\mathbb{Z}/q\mathbb{Z}$. The set Γ is fixed in advance and not secret, and can thus be considered as being a public parameter of the Boneh-Franklin traitor tracing scheme. We detail in §2.2 the codeword generation process from [5]. In §3.2 we propose a slightly different way to define Γ that has interesting practical consequences.

A private key is an element $\theta_i \in \mathbb{Z}/q\mathbb{Z}$ such that $\theta_i \cdot \boldsymbol{\gamma}^{(i)}$ is a representation of y with respect to the base $\langle h_1, \ldots, h_{2k} \rangle$. Thus, the i-th private key θ_i can be derived from the i-th codeword $\boldsymbol{\gamma}^{(i)}$ as

$$\theta_i = \frac{\sum_{j=1}^{2k} r_j \alpha_j}{\sum_{j=1}^{2k} r_j \gamma_j^{(i)}}, \tag{1}$$

where, obviously, the calculation takes place in $\mathbb{Z}/q\mathbb{Z}$. To encrypt a message $m \in \mathbb{G}_q$, one picks a random $a \in_R \mathbb{Z}/q\mathbb{Z}$ and calculates the ciphertext as $\langle m \cdot y^a, h_1^a, \ldots, h_{2k}^a \rangle$. Given a ciphertext $\langle s, p_1, \ldots, p_{2k} \rangle$, and the i-th secret key θ_i, the message m can be recovered as:

$$m = \frac{s}{\left(\prod_{j=1}^{2k} p_j^{\gamma_j^{(i)}} \right)^{\theta_i}}. \tag{2}$$

The correctness follows in a straightforward way from the fact that $\theta_i \cdot \boldsymbol{\gamma}^{(i)}$ is a representation of y with respect to the base $\langle h_1, \ldots, h_{2k} \rangle$. It follows that it is possible to decrypt a ciphertext given *any* representation $\langle \delta_1, \ldots, \delta_{2k} \rangle$ of y with respect to the base $\langle h_1, \ldots, h_{2k} \rangle$, since $\prod_{j=1}^{2k} (h_j^a)^{\delta_j} = y^a$; in other words, to decrypt it suffices to have a representation of y with respect to the base $\langle h_1, \ldots, h_{2k} \rangle$. Interestingly, Boneh and Franklin show in [5, Lemma 1] that if it is infeasible to compute discrete logarithms in $\mathbb{G}_q$, then convex combinations of $n < 2k$ given representations $\boldsymbol{\rho}^{(1)}, \ldots, \boldsymbol{\rho}^{(n)}$ of y are the *only* representations of y that can efficiently be constructed.

2.2 Codewords Generation

We describe the codewords $\boldsymbol{\gamma}^{(i)}$ generation process from [5] which is based on the use of Reed-Solomon codes [36]. Given the $(\ell - 2k) \times \ell$ matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & \ell \\ 1^2 & 2^2 & 3^2 & \dots & \ell^2 \\ \vdots & \vdots & \vdots & & \vdots \\ 1^{\ell-2k-1} & 2^{\ell-2k-1} & 3^{\ell-2k-1} & \dots & \ell^{\ell-2k-1} \end{pmatrix} \bmod q \tag{3}$$

over $\mathbb{Z}/q\mathbb{Z}$, let $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{2k}$ be a basis of the nullspace of $\boldsymbol{A}$. Boneh and Franklin define Γ as the rows of the $\ell \times 2k$ matrix

$$\boldsymbol{B} = \begin{pmatrix} | & | & | & & | \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 & \dots & \boldsymbol{b}_{2k} \\ | & | & | & & | \end{pmatrix}, \tag{4}$$

also over $\mathbb{Z}/q\mathbb{Z}$. Thus, Γ contains ℓ codewords each of length $2k$. By observing that any vector in the span of the rows of $\boldsymbol{A}$ corresponds to a polynomial of degree at most $\ell - 2k - 1$ evaluated at the points $1, \dots, \ell$, one can construct a basis of the nullspace of $\boldsymbol{A}$ using Lagrange interpolation. Using this construction the i-th codeword becomes $\langle u_i, iu_i, i^2u_i, \dots, i^{2k-1}u_i \rangle$ where $u_i^{-1} = \prod_{j \neq i}(i-j)$ and all computations are in $\mathbb{Z}/q\mathbb{Z}$. Naive computation of the ℓ codewords requires $\Omega(\ell^2)$ operations in $\mathbb{Z}/q\mathbb{Z}$. This can easily be turned into $O(\ell)$ operations using the following recursive formula:

$$u_1^{-1} = \prod_{j=1}^{\ell-1}(-j) \text{ and } u_{i+1}^{-1} = \frac{u_i(i-1)}{i-\ell} \text{ for } 1 \leq i \leq \ell - 1. \tag{5}$$

2.3 Tracing Procedure

We briefly recall the non-black-box tracing procedure [5]. Let $\boldsymbol{d} \in (\mathbb{Z}/q\mathbb{Z})^{2k}$ be a vector formed by taking a linear combination of at most k vectors in Γ. In practice $\boldsymbol{d}$ will be a convex combination, but we do not need that here. Since the vectors in Γ form the rows of the matrix $\boldsymbol{B}$, we know there exists a vector $\boldsymbol{w} \in (\mathbb{Z}/q\mathbb{Z})^{\ell}$ (having Hamming weight at most k) such that $\boldsymbol{wB} = \boldsymbol{d}$. The tracing procedure then works as follows. First, we determine a vector[4] $\boldsymbol{v} \in (\mathbb{Z}/q\mathbb{Z})^{\ell}$ such that $\boldsymbol{vB} = \boldsymbol{d}$. Since $(\boldsymbol{v} - \boldsymbol{w})\boldsymbol{B} = \boldsymbol{0}$, we know that $\boldsymbol{v} - \boldsymbol{w}$ lies in the linear span of the rows of $\boldsymbol{A}$ (recall that the rows of $\boldsymbol{A}$ span the vector space orthogonal to the one spanned by the columns of $\boldsymbol{B}$). In other words, there exists a unique polynomial f of degree at most $\ell - 2k - 1$ over $\mathbb{Z}/q\mathbb{Z}$ such that $\boldsymbol{v} - \boldsymbol{w} = \langle f(1), \dots, f(\ell) \rangle$. Taking into account that $\boldsymbol{w}$ has Hamming weight of at most k, we know that $\langle f(1), \dots, f(\ell) \rangle$ equals $\boldsymbol{v}$ in all but at most k components. Hence, it is possible to use Berlekamp-Welch algorithm [42] to find f from $\boldsymbol{v}$, after which f gives us $\boldsymbol{v} - \boldsymbol{w}$, from which we recover $\boldsymbol{w}$. The Berlekamp-Welch algorithm, published in a patent [42] granted in 1986, runs in $O(\ell^2)$. Asymptotically faster variants exist (see [3]), the fastest known being the one described by Pan [34] which runs in $O(\ell \log \ell \log \log \ell)$.

[4] Note that several such vectors exist.

As mentioned in §1.1, the best known black-box tracing procedure for the Boneh-Franklin scheme is not efficient since it has a $O(\binom{\ell}{k}k^2)$ complexity. We refer the reader to [5] for its description since it is out of the scope of this paper. Furthermore, we note that the black-box tracing procedure is vulnerable to the attacks described by Kiayias and Yung [22] which demonstrate that the Boneh-Franklin scheme is essentially incapable of black-box tracing super-logarithmic self-protecting traitor collusions unless the ciphertext size is linear in the number of users. Those two facts considerably limit the application of black-box tracing with the Boneh-Franklin scheme.

3 Revisiting the Tracing Mechanism

We recall several notions from coding theory. A *linear code* $\mathcal{C}$ over the vector space $(\mathbb{Z}/q\mathbb{Z})^\ell$ is a *subspace* of $(\mathbb{Z}/q\mathbb{Z})^\ell$. For our purposes we may assume that $\mathcal{C}$ has dimension $2k$ with $0 \leq 2k \leq \ell$. It follows that $\mathcal{C}$ contains q^{2k} *codewords.* The *minimal distance* d *of* $\mathcal{C}$ is the minimum Hamming weight of its non-zero codewords. A code $\mathcal{C}$ is called *maximum-distance separable* (*MDS*) if its minimal distance reaches the Singleton bound, i.e., if $d = \ell - 2k + 1$. A $2k \times \ell$ matrix $\boldsymbol{G}$ over $\mathbb{Z}/q\mathbb{Z}$ is called a *generator matrix* or *encoding matrix for* $\mathcal{C}$ if its rows form a linearly independent basis for $\mathcal{C}$. Thus, $\mathcal{C} = \{\boldsymbol{x} \in (\mathbb{Z}/q\mathbb{Z})^\ell : \boldsymbol{x} = \boldsymbol{z}\boldsymbol{G}$ where $\boldsymbol{z} \in (\mathbb{Z}/q\mathbb{Z})^{2k}\}$ and $\mathcal{C}$ is the code *associated* to G. The *dual code* $\mathcal{C}^\perp$ of a linear code $\mathcal{C}$ is the linear code $\mathcal{C}^\perp = \{\boldsymbol{x} \in (\mathbb{Z}/q\mathbb{Z})^\ell : \boldsymbol{x}\boldsymbol{c}^T = \boldsymbol{0}$ for all $\boldsymbol{c} \in \mathcal{C}\}$. A *reduced parity-check matrix* for the code $\mathcal{C}$ is an $(\ell - 2k) \times \ell$ matrix $\boldsymbol{H}$ over $\mathbb{Z}/q\mathbb{Z}$ such that $\mathcal{C} = \{\boldsymbol{x} \in (\mathbb{Z}/q\mathbb{Z})^\ell : \boldsymbol{x}\boldsymbol{H}^T = \boldsymbol{0}\}$. Receiving a noisy version $\tilde{\boldsymbol{x}}$ of a codeword $\boldsymbol{x}$, the vector $\boldsymbol{s} = \tilde{\boldsymbol{x}}\boldsymbol{H}^T$ is called the *syndrome*. Writing $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{e}$ is called the *error pattern*, we see that the syndrome $\boldsymbol{s} = (\boldsymbol{x} + \boldsymbol{e})\boldsymbol{H}^T = \boldsymbol{0} + \boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{e}\boldsymbol{H}^T$ depends only on the error pattern, and not on the codeword itself. Finally, the following lemma makes clear the link between a reduced parity-check matrix of a linear code and its dual code.

Lemma 1. *$\boldsymbol{H}$ is a parity-check matrix for the linear code $\mathcal{C}$ if and only if $\mathcal{C}$ spans the subspace orthogonal to the row space of $\boldsymbol{H}$.*

Therefore, a reduced parity-check matrix for $\mathcal{C}$ is an encoding matrix for the dual code $\mathcal{C}^\perp$ and conversely.

3.1 Generalized Reed-Solomon Codes

Given vectors $\boldsymbol{\pi} = (\pi_i)_{i=1}^\ell, \boldsymbol{c} = (c_i)_{i=1}^\ell \in (\mathbb{Z}/q\mathbb{Z})^\ell$, a *Generalized Reed-Solomon code* $\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{c})$ is defined as follows:

$$\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{c}) = \left\{(c_i f(\pi_i))_{i=1}^\ell : f(x) \in (\mathbb{Z}/q\mathbb{Z})[x] \text{ and } \deg(f) < 2k\right\}. \qquad (6)$$

Thus, a codeword in $\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{c})$ is a vector consisting of a polynomial of degree less than $2k$ over $\mathbb{Z}/q\mathbb{Z}$ evaluated at the ℓ points $\pi_1, \ldots, \pi_\ell$ scaled by $c_1, \ldots, c_\ell$. It is well-known that GRS codes are MDS codes, i.e., $d = \ell - 2k + 1$. When $\boldsymbol{c} = (1, 1, \ldots, 1)$, we speak of Reed-Solomon codes. The following theorem states that the dual of a GRS code is a GRS code (see [20, page 66] for a proof).

Theorem 1. *The dual of a* $\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{c})$ *code is*

$$\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{c})^{\perp} = \mathrm{GRS}_{\ell,\ell-2k}(\boldsymbol{\pi}, \boldsymbol{d}) \tag{7}$$

where $\boldsymbol{d} = (d_1, \ldots, d_\ell)$ *with* $d_i^{-1} = c_i \prod_{j \neq i}(\pi_i - \pi_j)$.

The above allows us to rephrase the Boneh-Franklin codeword generation mechanism described in §2.2 as follows: the matrix $\boldsymbol{A}$ defined in (3) is the generator matrix of a $\mathrm{GRS}_{\ell,\ell-2k}(\boldsymbol{\pi}, \boldsymbol{c})$ code over $\mathbb{Z}/q\mathbb{Z}$ with $\boldsymbol{\pi} = (1, \ldots, \ell)$ and $\boldsymbol{c} = (1, 1, \ldots, 1)$ (this fact was already recognized by [23], for instance), while the matrix $\boldsymbol{B}$ defined in (4) is a (transposed) reduced parity-check matrix for the same code. Conversely, in the light of Lemma 1 and Theorem 1, the matrix $\boldsymbol{B}^T$ can be seen as a generator matrix of the dual $\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{d})$ of $\mathrm{GRS}_{\ell,\ell-2k}(\boldsymbol{\pi}, \boldsymbol{c})$, where $\boldsymbol{d}$ is as in Theorem 1. Thus, Γ consists of vectors forming a basis of the $2k$-dimensional vector space which contains the syndromes of $\mathrm{GRS}_{\ell,\ell-2k}(\boldsymbol{\pi}, \boldsymbol{c})$.

3.2 More Efficient Codewords

The above more general framework allows us to define the code Γ in such a way that both the codeword generation and decryption become faster without affecting the security of the Boneh-Franklin scheme.

Using Theorem 1, we observe that in order to compute the codewords we can avoid Lagrange interpolation and recursive formula (5): let $\boldsymbol{B}$ be the generator matrix of a $\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{d})$ code with $\boldsymbol{\pi} = (1, 2, \ldots, \ell)$ and $\boldsymbol{d} = (1, 1, \ldots, 1)$, then the i-th codeword can simply be defined as $\boldsymbol{\gamma}^{(i)} = \langle 1, i, i^2, \ldots, i^{2k-1} \rangle$, for $i = 1, 2, \ldots, \ell$. This in turn allows us to rewrite the decryption operation (2) as

$$m = \frac{s}{\left(\prod_{j=1}^{2k} p_j^{i^{j-1}}\right)^{\theta_i}} = \frac{s}{\left(\left(\left(\left(p_{2k}^{i} p_{2k-1}\right)^{i} \cdots\right)^{i} p_2\right)^{i} p_1\right)^{\theta_i}}. \tag{8}$$

Compared to (2), this replaces $2k$ of the $2k+1$ $\log_2 q$-bit modular exponentiation exponents by $\log_2 \ell$-bit ones. With $\ell \approx 2^{20}$ and assuming 80-bit security with 160-bit q, this results in a speedup by a factor of 7, which is much more effective than using multi-exponentiations (cf. [30, page 617]) as suggested in [5]. In practice, the efficiency of our decryption is comparable to [29]. Furthermore, provided each receiver knows its identity number i, it can directly compute codeword $\boldsymbol{\gamma}^{(i)}$ without requiring knowledge of the Lagrange coefficients attached to the receiver with identity $i - 1$.

We note that the semantic security of the Boneh-Franklin scheme is not impacted by the nature of the code, while its tracing capabilities only depend on the minimal distance of the code. In our case, we use Generalized Reed-Solomon codes with the same minimal distance as the one used by Boneh and Franklin.

3.3 An Efficient Tracing Procedure

In this section, we present in two steps a new and efficient non-black-box tracing procedure for the Boneh-Franklin scheme. We stress that this new tracing procedure can be used for any type of Reed-Solomon and generalized Reed-Solomon codes, being the original code described in [5], the faster code discussed in §3.2 or the variant we will discuss in §5. First, we reduce the complexity from $O(\ell \log \ell \log \log \ell)$ to $O(\ell)$, using a technique based on the Berlekamp-Massey algorithm [28] and Chien search [11]. Then, we improve it to expected complexity $\tilde{O}(k^2)$ by replacing Chien search by the Cantor-Zassenhaus factorization algorithm [10].

As outlined in §2.3, the Boneh-Franklin tracing procedure based on Berlekamp-Welch algorithm consists in finding a noisy codeword which results in the syndrome discovered in the pirate receiver, and in decoding this codeword. More precisely, let $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ denote a codeword belonging to $\mathrm{GRS}_{\ell,\ell-2k}(\boldsymbol{\pi}, \boldsymbol{c})$ with $\boldsymbol{c} = (1, 1, \ldots, 1)$ and its noisy version, respectively. We can interpret both $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ as polynomials $f(x)$ and $\tilde{f}(x)$ in $(\mathbb{Z}/q\mathbb{Z})[x]$. If no error is introduced in the codeword, then $d_i f(\pi_i) = \tilde{f}_i$ for $1 \leq i \leq \ell$, where $\tilde{f}(x) = \sum_{i=1}^{\ell} \tilde{f}_i x^{i-1}$. Let $g(x) \in (\mathbb{Z}/q\mathbb{Z})[x]$ be a polynomial (hereafter called an error-locator polynomial) of degree at most k with $g(\pi_i) = 0$ for those π_i's for which $d_i f(\pi_i) \neq \tilde{f}_i$. This leads to the following system of ℓ linear equations in ℓ unknowns: $d_i f(\pi_i) g(\pi) = g(\pi) \tilde{f}_i$. Solving the system, one obtains the polynomial $g(x)$, from which the error locations can be derived. Along with $g(x)$, one also gets $d_i f(x) g(x)$ and thus $f(x)$. Straightforward implementation using Gaussian reduction would lead to $O(\ell^3)$ complexity. Faster approaches would be to use the Berlekamp-Welch algorithm in $O(\ell^2)$ or others of complexity $\tilde{O}(\ell)$ (see [42, 3, 34]).

The key observation to derive a faster tracing algorithm is to note that computing a (noisy) codeword from the syndrome retrieved from a pirate receiver and then decoding this codeword, as done above, is *not* necessary: actually, the pirate syndrome itself suffices to trace the legitimate syndromes used to derive it. Indeed, as pointed out by Massey [28], the *Berlekamp-Massey algorithm* allows reconstruction of the error-locator polynomial from the syndrome only. This key property permits us to stop the decoding process earlier for the purpose of tracing and thus reduce the complexity, since we are interested in the error-locator polynomial only and we do not need the amplitudes of the errors.

We now clarify the link between the error-locator polynomial and the syndrome, following [43, page 214]. Let $\tilde{f}(x) = f(x) + e(x)$, where $\tilde{f}(x)$, $f(x)$ and $e(x)$ are the received codeword, the original codeword, and the error polynomial, respectively. Let $s(x) = s_0 + s_1 x + \cdots + s_{2k-1} x^{2k-1}$ denote the syndrome vector interpreted as a polynomial. Let $g(x)$ denote an error-locator polynomial whose zeroes are the *inverses* of the error locations $\sigma_j = \pi_i$ with $1 \leq j \leq k$ and with $i \in \mathcal{I}$ for a cardinality k subset $\mathcal{I}$ of $\{1, 2, \ldots, \ell\}$:

$$g(x) = \prod_{j=1}^{k} (1 - \sigma_j x) = g_0 + g_1 x + \cdots + g_k x^k. \tag{9}$$

Let $t_1, t_2, \ldots, t_k$ be the indices of the non-zero coefficients of $e(x)$. Because $g(\sigma_m^{-1}) = 0$ for all error locations σ_m with $1 \leq m \leq k$, it follows that

$$e_{t_m} \sigma_m^j g(\sigma_m^{-1}) = 0,$$

and thus

$$e_{t_m}(g_k\sigma_m^{-k+j} + g_{k-1}\sigma_m^{-k+j+1} + \cdots + g_1\sigma_m^{j-1} + g_0\sigma_m^j) = 0 \tag{10}$$

for any j. Summing (10) over $m = 1, 2, \ldots k$ gives an expression from which Newton's identities can be constructed:

$$\begin{aligned}
& \textstyle\sum_{m=1}^{k} e_{t_m}(g_k\sigma_m^{-k+j} + g_{k-1}\sigma_m^{-k+j+1} + \cdots + g_1\sigma_m^{j-1} + g_0\sigma_m^j) \\
&= g_k \textstyle\sum_{m=1}^{k} e_{t_m}\sigma_m^{j-k} + g_{k-1}\textstyle\sum_{m=1}^{k} e_{t_m}\sigma_m^{j-k+1} + \cdots + g_0\textstyle\sum_{m=1}^{k} e_{t_m}\sigma_m^{j} \\
&= \qquad g_k s_{j-k} + g_{k-1}s_{j-k+1} + \cdots + g_1 s_{j-1} + g_0 s_j = 0.
\end{aligned}$$

The last equality comes from the fact that the following system of equations can be written using the parity-check matrix:

$$\begin{aligned}
s_0 &= e_{t_1} + e_{t_2} + \cdots + e_{t_k} \\
s_1 &= e_{t_1}\sigma_1 + e_{t_2}\sigma_2 + \cdots + e_{t_k}\sigma_k \\
s_2 &= e_{t_1}\sigma_1^2 + e_{t_2}\sigma_2^2 + \cdots + e_{t_k}\sigma_k^2 \\
&\cdots \\
s_{2k-1} &= e_{t_1}\sigma_1^{2k-1} + e_{t_2}\sigma_2^{2k-1} + \cdots + e_{t_k}\sigma_k^{2k-1}.
\end{aligned}$$

From (9) it follows that $g_0 = 1$, which leads to the order k linear recurrence relation

$$g_k s_{j-k} + \cdots + g_1 s_{j-1} = -s_j. \tag{11}$$

Given $2k$ consecutive terms of an order k linear recurrence, the Berlekamp-Massey algorithm computes the coefficients of the recurrence in time $O(k^2)$. Because the s_i for $i = 0, 1, \ldots, 2k-1$ are known, the g_i can thus be computed directly in time $O(k^2)$.

After the error-locator polynomial $g(x)$ has been computed, the remaining task consists in finding its roots, which are the inverses of identities of the traitors. Traditionally, Reed-Solomon decoders rely on the Chien search algorithm [11] which searches over the possible roots. In our case, this results in a complexity of $O(\ell)$. The roots can, however, be located faster by *factorizing* $g(1/x)$ using the Cantor-Zassenhaus algorithm [10] within expected time $O(k^2 \log k \log\log k(\log q + \log k)) = \tilde{O}(k^2)$. This algorithm works recursively on the squarefree polynomial $g(x)$ whose irreducible factors[5] are all of degree 1. It is based on the fact[6] that $g(x) = \gcd(g(x), r(x)) \cdot \gcd(g(x), r(x)^{(p^d-1)/2} + 1) \cdot \gcd(g(x), r(x)^{(p^d+1)/2} - 1)$ for any polynomial $r(x) \in (\mathbb{Z}/q\mathbb{Z})[x]$.

Then, the obtained roots directly reveal the identities of the traitors. The overall complexity of our tracing procedures is $\tilde{O}(k^2)$ which is independent of ℓ. The latter is not the case for the schemes based on algebraic codes described by Silverberg *et al.* in [37,38].

Our method makes it possible to trace large coalitions in Boneh-Franklin systems with a virtually *unlimited* number of users, and this without requiring

[5] $g(x)$ in fact fulfills these conditions if $g(x)$ has at most k roots.

[6] The interested reader will find more details about the Cantor-Zassenhaus algorithm in [14, page 128].

any modification of the encryption scheme. Our implementation, based on the `GMP` [31] and `LiDIA` [26] software libraries and working over the group of points of an elliptic curve over a finite field of cardinality approximately 2^{160}, allows tracing of a coalition of $k = 1024$ traitors in a system of $\ell = 200'000'000$ users in less than *two minutes* on a common desktop PC. These parameter values cannot realistically be handled using the Berlekamp-Welch algorithm as described in [5].

4 Above-Threshold Tracing

In [5] Boneh and Franklin emphasize an interesting property of their scheme, namely the possibility to trace a collusion of more than k traitors using list-decoding techniques like the Guruswami-Sudan algorithm [18, 19]. This would correspond to finding more than k errors in a codeword. In such cases, the Berlekamp-Welch algorithm fails to find the polynomial $f(x)$. The Berlekamp-Massey approach fails as well, since it outputs a polynomial of degree k that does not have k roots over $\mathbb{Z}/q\mathbb{Z}$. The algorithm of Guruswami and Sudan allows, under certain circumstances, to find a candidate for the polynomial $f(x)$. In this section we investigate under which circumstances tracing is possible and how it will influence system parameters. We finally show that the Guruswami-Sudan algorithm can detect only a few additional traitors, and this at high cost.

4.1 Guruswami-Sudan Algorithm for Reed-Solomon Codes

This algorithm attempts to find the message polynomial $f(x)$ given a received codeword when more than k errors occurred. It can be thought of as a generalization of the Berlekamp-Welch algorithm. Let ℓ and k be as above. Given ℓ pairs $(\pi_i, c_i) \in (\mathbb{Z}/q\mathbb{Z})^2$ for $1 \leq i \leq \ell$, message length $\ell - 2k$, and an error parameter $k' \leq \ell - 1 - \sqrt{\ell(\ell - 2k - 1)}$, the Guruswami-Sudan algorithm finds all univariate polynomials f of degree at most $\ell - 2k - 1$ such that $f(\pi_i) = c_i$ for at least $\ell - k'$ values of i. Thus, the algorithm allows correction of at most k' errors. It consists of two steps. In the first step a parameter r is selected and a system of $O(\ell r^2)$ linear equations is solved to find a non-zero bivariate polynomial $Q(x, y)$ of a certain weighted degree[7] such that $Q(\pi_i, c_i) = 0$ for $1 \leq i \leq \ell$. The parameter r, which is the multiplicity of the singularity of $Q(x, y)$, is chosen in such a way that as many errors as possible can be handled while keeping the system of equations tractable. In the second step, factors $(y - f(x))$ of $Q(x, y)$ are determined such that $\deg(f(x)) \leq \ell - 2k - 1$. For a complete description of the method see [18, 19]. Below we are interested in its practical feasibility (in particular of the first step) in the context of the traitor tracing problem.

4.2 List Decoding and Traitor Tracing

In this section we have a closer look at the various parameters of of the Guruswami-Sudan algorithm. We will see that this leads to the unavoidable conclusion that it is of little practical significance for our type of applications.

[7] $\deg_x(Q(x, y))m + \deg_y(Q(x, y))n$ is called the (m, n)-weighted degree of $Q(x, y)$.

Since the traditional algorithms (such as Berlekamp-Welch) can trace up to k traitors, the only case of interest is $k' > k$. Let $\delta = k' - k$ be the number of additional traitors we wish to be able to trace, and let $\phi = \ell - 2k - 1$. Because at most $\ell - 1 - \sqrt{\ell\phi}$ traitors can be traced, only k's need to be considered for which

$$k + \delta \leq \ell - 1 - \sqrt{\ell\phi} \tag{12}$$

for a $\delta \geq 1$.

With $\omega = r(\ell - k - \delta) - 1$, in the first step of the Guruswami-Sudan algorithm a system needs to be solved over $\mathbb{Z}/q\mathbb{Z}$ involving $\ell r(r+1)/2$ constraints and

$$\left(\omega + 1 - \frac{\phi}{2}\left\lfloor\frac{\omega}{\phi}\right\rfloor\right)\left(\left\lfloor\frac{\omega}{\phi}\right\rfloor + 1\right)$$

unknowns [18, 19]. It follows that

$$\left(\omega + 1 - \frac{\phi}{2}\left\lfloor\frac{\omega}{\phi}\right\rfloor\right)\left(\left\lfloor\frac{\omega}{\phi}\right\rfloor + 1\right) \geq \frac{\ell r(r+1)}{2}. \tag{13}$$

Furthermore, since in practice q will have at least 160 bits, it is reasonable to limit the number of constraints to 10000 if we want to be able to store the matrix in 2GB of memory. This leads to

$$\frac{\ell(r+1)r}{2} < 10000. \tag{14}$$

Note that this immediately limits the practical applicability of the Guruswami-Sudan algorithm to tracing in systems of at most a few thousand users. This is in sharp contrast with our syndrome-only tracing which allows millions of users.

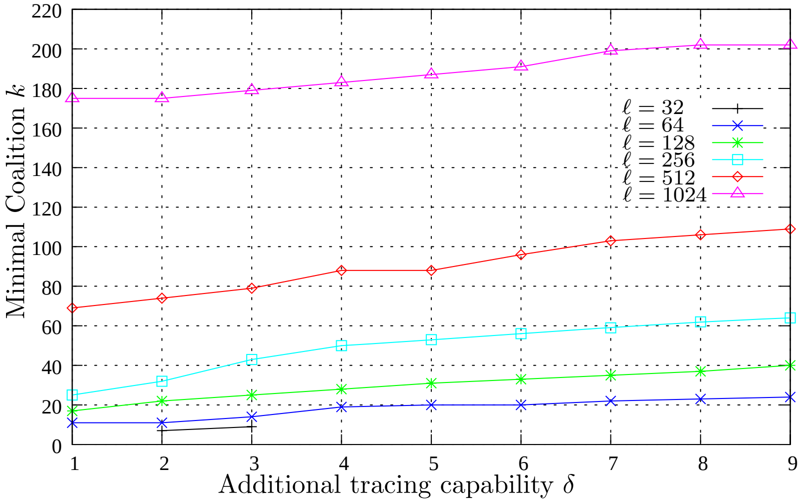


Fig. 1. Minimal coalition with respect to a given above-threshold tracing capacity

Define the *minimal coalition size* as the smallest k such that (12), (13), and (14) are satisfied. For any ℓ and δ, this k follows from a simple search, as illustrated in Fig. 1 for several (small) numbers of users. For example, in a system with $\ell = 512$ users the minimal initial coalition size is 69 in order to be able to trace a single additional key if 70 pirates collude. In many applications, this results in an overkill, because the ciphertext and private key, which are dependent on the coalition size, become too large. As illustration, let us consider the following case: for $\ell = 1024$ and $k = 500$, we get $k + \delta = 855$, which may seem fairly good. However, the required bandwidth to transmit the ciphertext is equal to 1001 group elements. This is only 2.24% less than a trivial scheme involving an individual encryption based on El-Gamal which additionally would bring natural revocation capabilities. Besides that, a system of size $\ell = 1024$ is not far from the limit capacity of the original Berlekamp-Welch algorithm. Hence this method is not applicable for systems with large number of users, constrained bandwidth and key-space storage capability.

5 Beyond-Threshold Security

In practical scenarios, there are three distinct cases for the number of compromised keys in a coalition, namely: at most k, between $k+1$ and $2k-1$, and $2k$ keys or more. The first case corresponds to the situation for which the Boneh-Franklin scheme has been designed and security guarantees have been derived, while the second case corresponds to the above-threshold tracing scenario described in §4. In this section we discuss the third case.

Suppose that an adversary has managed to get $2k$ private elements θ_{i_s}, for $1 \leq s \leq 2k$ and assume, as before, that the vectors in $\mathit{\Gamma}$ are public. Because Eq. (1) over $\mathbb{Z}/q\mathbb{Z}$ can be rewritten as

$$\theta_{i_s}^{-1} = \frac{\sum_{j=1}^{2k} r_j \gamma_j^{(i_s)}}{\sum_{j=1}^{2k} r_j \alpha_j} = \sum_{j=1}^{2k} \omega_j \gamma_j^{(i_s)} \tag{15}$$

with $\omega_j = r_j / \sum_{j=1}^{2k} r_j \alpha_j$, knowledge of the $2k$ private keys θ_{i_s} leads to a system of $2k$ linear equations in the $2k$ unknowns ω_j, for $1 \leq j \leq 2k$. After determining the ω_j's using for instance Gaussian reduction, the adversary can compute *any* other private key θ_i in the system:

$$\theta_i = \left(\sum_{j=1}^{2k} \omega_j \gamma_j^{(i)} \right)^{-1} .$$

Not only will the adversary be able to create any number of untraceable combinations of keys, but he will also be able to distribute newly derived keys so that *innocent* users (whose keys were *a priori* never compromised) may be accused of treachery. We note that this observation applies not only to the Boneh-Franklin scheme, but to many tracing schemes that are based on a publicly-known linear code such as the generalizations described by Kurosawa and Yoshida [25].

An obvious way to repair this annoying property of the Boneh-Franklin scheme would require keeping the tracing code matrix secret, while making sure that the vectors $\boldsymbol{\gamma}^{(i)} = \langle \gamma_1^{(i)}, \ldots, \gamma_{2k}^{(i)} \rangle$ are statistically decorrelated. In that case acquiring $2k$ representations should give an adversary no information about other representations. This idea was already used by Kiayias and Yung in [23] for the different goal of obtaining an asymmetric traitor-tracing scheme. A way to achieve this would be to choose the i-th codeword $\gamma^{(i)}$ as $\boldsymbol{\gamma}^{(i)} = \langle 1, \zeta_i, \ldots, \zeta_i^{2k-1} \rangle$ where $\zeta_i \in_R \mathbb{Z}/q\mathbb{Z}$ with $1 \leq i \leq \ell$ is drawn independently and uniformly[8] at random for each $\gamma^{(i)}$. Here, a $\mathrm{GRS}_{\ell,2k}(\boldsymbol{\pi}, \boldsymbol{d})$ code is used, with $\boldsymbol{\pi} = (\zeta_1, \zeta_2, \ldots, \zeta_\ell)$ and $\boldsymbol{d} = (1, 1, \ldots, 1)$. The ith receiver has to protect the entire representation $\langle \theta_i \gamma_1^{(i)}, \ldots, \theta_i \gamma_{2k}^{(i)} \rangle$, and thus, to store at least θ_i and ζ_i in tamper-proof memory. Furthermore, the fast codeword generation method from §3.2 can no longer be used.

By applying the above codeword distribution method, an adversary who acquires $2k$ or more keys will be unable to derive any information about the tracing codewords that are used in the representations. She will only be capable of creating combinations of the representations. If there are fewer than $k + 1$ keys in a combination, we are back to a standard tracing scenario. Otherwise, combinations of $k + 1$ or more keys will be detected, but not traceable, since our tracing algorithm will be unable to factorize the error-locator polynomial nor the original approaches will reveal the traitors.

6 Conclusion

In this paper, we have presented new insights as well as several improvements to the Boneh-Franklin traitor tracing scheme [5]. First of all, we revisited the private key watermarking scheme based on Reed-Solomon codes; based on this, we describe a new non-black-box tracing algorithm whose complexity only depends on the square of the maximal coalition size k and is independent of the total number ℓ of users. Our new tracing algorithm does not require any change in the encryption scheme and can be used with any generalized Reed-Solomon codes.

This allows us to implement the scheme in a system with a virtually unlimited number of users; in other words, the maximal coalition size is only constrained by the channel bandwidth and the computational capacity of the receivers. This new tracing algorithm can also be applied with any other scheme relying on (generalized) Reed-Solomon codes to watermark the distributed private keys.

Additionally, we discussed the application of the Guruswami-Sudan list-decoding algorithm, whose use was proposed in [5], and showed that, in practice, it brings only a marginal improvement in tracing capabilities, and this at high cost.

As a final step, we studied the above-threshold security of the Boneh-Franklin scheme, i.e., the malicious capabilities of an adversary having access to many

[8] Note that for practical values of ℓ, a collision between two codewords has a negligible probability to occur.

more than k keys. We showed that, given a coalition size of k, an adversary who has recovered $2k$ private keys or more can derive *any* other private key, provided the code Γ is publicly known, as advocated in [5]. To the best of our knowledge, this 'feature' has not been reported in the literature. To deal with this problem, we suggest to keep the tracing code matrix secret and to distribute *statistically independent* codewords to the receivers.

Even though the Boneh-Franklin scheme can encrypt only small messages (basically, one group element), and even though using it in a hybrid fashion by encrypting a symmetric session key is prone to a trivial untraceable strategy[9], we believe based on our results that, in order to fight illegitimate clones of tamper-proof modules, the Boneh-Franklin scheme is now really worth to be considered in scenarios where trivial untraceable strategies are unavoidable[10] by design. Of course, this statement is based on the assumption that it is possible to revoke a traced clone by some other mechanism.

Acknowledgments

We would like to thank Olivier Billet, Olivier Brique, Nicolas Fischer, Jim Fuller, Michael Hill, Corinne Le Buhan Jordan, André Nicoulin, Karl Osen, Martijn Stam as well as the anonymous reviewers of PKC'09 for interesting discussions and comments about this paper.

References

1. Anderson, R.: Security engineering – a guide to building dependable distributed systems. Wiley, Chichester (2001)
2. Billet, O., Phan, D.: Efficient traitor tracing from collusion secure codes. In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 171–182. Springer, Heidelberg (2008)
3. Bini, D., Pan, V.: Polynomial and matrix computations: fundamental algorithms. Progress in Theoretical Computer Science Series, vol. 1. Birkhauser Verlag, Basel (1994)
4. Boneh, D.: The decision Diffie-Hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
5. Boneh, D., Franklin, M.: An efficient public key traitor tracing scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)
6. Boneh, D., Naor, M.: Traitor tracing with constant size ciphertext (manuscript 2008), `http://crypto.stanford.edu/~dabo/papers/const-tt.pdf`
7. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
8. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. IEEE Transactions on Information Theory 44(5), 1897–1905 (1998)

[9] This strategy is simply to share the session key.

[10] Like in Pay-TV systems using the DVB-CSA [39] standard encryption, for instance.

9. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace and revoke system. In: Juels, A., Wright, R., De Capitani de Vimercati, S. (eds.) Proceedings of the 13th ACM Conference on Computer and Communication Security, CCS 2006, Alexandria, USA, October 30 - November 3, pp. 211–220. ACM Press, New York (2006)
10. Cantor, D., Zassenhaus, H.: A new algorithm for factoring polynomials over finite fields. Mathematics of Computation 36(154), 587–592 (1981)
11. Chien, R.: Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. IEEE Transactions on Information Theory 10(4), 357–363 (1964)
12. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
13. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing traitors. IEEE Transactions on Information Theory 46(3), 893–910 (2000)
14. Cohen, H.: A course in computational algebraic number theory. Springer, Heidelberg (2000)
15. Dodis, Y., Fazio, N., Kiayias, A., Yung, M.: Scalable public-key tracing and revoking. In: Rajsbaum, S. (ed.) PODC 2003, Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, July 13-16, 2003, pp. 190–199. ACM Press, Boston (2003)
16. Dodis, Y., Fazio, N., Kiayias, A., Yung, M.: Scalable public-key tracing and revoking. Distributed Computing 17(4), 323–347 (2005)
17. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
18. Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometric codes. In: 39th Annual Symposium on Foundations of Computer Science (FOCS 1998), California, USA, November 8-11, 1998, pp. 28–39. IEEE Computer Society, Los Alamitos (1998)
19. Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometry codes. IEEE Transactions on Information Theory 45(6), 1757–1767 (1999)
20. Hall, J.: Notes on coding theory – Generalized Reed-Solomon codes (2003), `http://www.mth.msu.edu/~jhall/classes/codenotes/GRS.pdf`
21. Kiayias, A., Pehlivanoglu, S.: Pirate evolution: how to make most of your traitor keys. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 448–465. Springer, Heidelberg (2007)
22. Kiayias, A., Yung, M.: Self protecting pirates and black-box traitor tracing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 63–79. Springer, Heidelberg (2001)
23. Kiayias, A., Yung, M.: Breaking and repairing asymmetric public-key traitor tracing. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 32–50. Springer, Heidelberg (2003)
24. Kurosawa, K., Desmedt, Y.: Optimum traitor tracing and asymmetric schemes with arbiter. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 145–157. Springer, Heidelberg (1998)
25. Kurosawa, K., Yoshida, T.: Linear code implies public-key traitor tracing. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 172–187. Springer, Heidelberg (2002)
26. LiDIA A C++ Library for Computational Number Theory. Software, `http://www.cdc.informatik.tu-darmstadt.de/TI/LiDIA/`
27. Mangard, S., Oswald, E., Popp, T.: Power analysis – revealing the secrets of smart cards. Springer, Heidelberg (2007)

28. Massey, J.: Shift-register synthesis and BCH decoding. IEEE Transactions on Information Theory 15(1), 122–127 (1969)
29. McGregor, J., Yin, Y., Lee, R.: A traitor tracing scheme based on RSA for fast decryption. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 56–74. Springer, Heidelberg (2005)
30. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of applied cryptography. The CRC Press series on discrete mathematics and its applications. CRC Press, Boca Raton (1997)
31. GNU Multiple Precision Arithmetic Library, `http://gmplib.org`.
32. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
33. Naor, M., Pinkas, B.: Threshold traitor tracing. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 502–517. Springer, Heidelberg (1998)
34. Pan, V.: Faster solution of the key equation for decoding BCH error-correcting codes. In: Leighton, F., Shor, P. (eds.) Proceedings, 29th Annual ACM Symposium on the Theory of Computing (STOC), pp. 168–175. ACM Press, New York (1997)
35. Pfitzmann, B.: Trials of traced traitors. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 49–64. Springer, Heidelberg (1996)
36. Reed, I., Solomon, G.: Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics (SIAM) 8(2), 300–304 (1960)
37. Silverberg, A., Staddon, J., Walker, J.: Efficient traitor tracing algorithms using list decoding. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 175–192. Springer, Heidelberg (2001)
38. Silverberg, A., Staddon, J., Walker, J.: Applications of list decoding to traitor tracing. IEEE Transactions on Information Theory 49(5), 1312–1318 (2003)
39. Digital Video Broadcasting (DVB) Conditional Access Standards, `http://www.dvb.org/technology/standards/index.xml#conditional`
40. Stinson, D., Wei, R.: Key preassigned traceability schemes for broadcast encryption. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 144–156. Springer, Heidelberg (1999)
41. Wallner, D., Harder, E., Agee, R.: Key management for multicast: issues and architectures. RFC 2627 (1999), `http://www.ietf.org`
42. Welch, L., Berlekamp, E.: Error correction for algebraic block codes. US Patent 4’633’470 (1986)
43. Wicker, S.: Error control systems for digital communications and storage. Prentice-Hall, Englewood Cliffs (1995)
44. Wong, C., Gouda, M., Lam, S.: Secure group communications using key graphs. In: Proceedings of the ACM SIGCOMM 1998 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Vancouver, British Columbia, Canada, August 31 - September 4, 1998, pp. 68–79. ACM Press, New York (1998)

Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols

M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto

Information Security Institute, Faculty of IT, Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia
mc.gorantla@isi.qut.edu.au, {c.boyd,j.gonzaleznieto}@qut.edu.au

Abstract. A key exchange protocol allows a set of parties to agree upon a secret session key over a public network. Two-party key exchange (2PKE) protocols have been rigorously analyzed under various models considering different adversarial actions. However, the analysis of group key exchange (GKE) protocols has not been as extensive as that of 2PKE protocols. Particularly, the security attribute of key compromise impersonation (KCI) resilience has so far been ignored for the case of GKE protocols. We first model the security of GKE protocols addressing KCI attacks by both outsider and insider adversaries. We then show that a few existing protocols are not secure even against outsider KCI attacks. The attacks on these protocols demonstrate the necessity of considering KCI resilience. Finally, we give a new proof of security for an existing GKE protocol under the revised model assuming random oracles.

Keywords: Group Key Exchange, Key Compromise Impersonation, Insider Attacks.

1 Introduction

A group key exchange (GKE) protocol allows a group of parties to agree upon a secret common session key over a public network. Although there had been GKE protocols earlier, Bresson et al. [1,2,3] were the first to analyze the security of GKE protocols under formal security models. These models define authenticated key exchange (AKE) security and mutual authentication as the desired notions of security against an outsider adversary i.e. assuming that the adversary is not part of the group. The notion of AKE-security demands that an outsider adversary should not learn the session key while the notion of mutual authentication requires that parties who complete the protocol execution should output identical session keys and that each party should be ensured of the identity of the other participating parties.

Katz and Shin [4] define insider security for GKE protocols by separating the requirements of mutual authentication in the presence of insiders into *agreement* and *security against insider impersonation attacks*. Their definition has been revisited by Bohli et al. [5] and Bresson and Manulis [6] under different corruption models. Bohli et al. also define the notion of *contributiveness* which requires that

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 105–123, 2009.

a proper subset of insiders should not predetermine the resulting session key. Bresson and Manulis strengthen this notion by considering strong corruptions where the ephemeral session state of an instance may also be revealed in addition to the long-term private key of the party.

All the models above (for both outsider and insider security) assume that a party will be fully under the control of the adversary once the party's long-term private key is compromised. These models however consider forward secrecy to limit the damage done by compromise of long-term private key after session completion. Another equally important security attribute related to compromise of the long-term private key of parties is key compromise impersonation (KCI) resilience. Informally, an adversary is said to impersonate a party B to another party A if B is honest and the protocol instance at A accepts the session with B as one of the session peers but there exists no such partnered instance at B [4]. In a successful KCI attack, an adversary with the knowledge of the long-term private key of a party A can impersonate B to A. Resilience to KCI attacks is often seen as a desired security attribute for two-party key exchange (2PKE) [7,8]. However, it has so far been ignored for the case of group key exchange.

We argue that KCI resilience for GKE protocols is at least as important as it is for 2PKE protocols. For this purpose we illustrate two scenarios with different setup assumptions, where a KCI attack is a threat. We first extend the peer-to-peer file sharing system scenario given for the two-party case by Ng [9, Section 4.2.2], to the group case. In these systems each user stores some data and allows access only to users with whom it wants to share the data. This can be achieved by executing a GKE protocol with the peers who have read access to the data and sending them the data encrypted using the established session key. Let A be the party who has some sensitive data. The goal of an adversary $\mathcal{A}$ is to access the data at A which is to be shared only with the users who have read access. Although the compromise of A's long-term private key helps $\mathcal{A}$ to impersonate A, $\mathcal{A}$ may not be able to access the data locally stored at A. However, if the GKE protocol used is vulnerable to KCI attacks, $\mathcal{A}$ can impersonate a party who has read access and decrypt the data using the session key. Note that in this scenario, the GKE protocol needs to have forward secrecy. Otherwise, compromising A's private key will enable the adversary to obtain the session key. On the other hand, having forward secrecy alone is not sufficient as $\mathcal{A}$ will be able to spoof the presence of an honest party if the protocol is not KCI resilient as discussed above.

The second one is a server-client scenario for the application given by Bresson et al. [10]. They propose a GKE protocol which allows a cluster of mobile devices (acting as clients) to agree upon a session key with a wireless gateway (acting as a server). The authors suggest that the established session key can be used along with a suitable protocol to secure IEEE 802.11 wireless networks. If the long-term private key of the gateway is compromised, an adversary can easily impersonate the gateway and allow any mobile device to access the wireless network. However, impersonating the gateway may be recognized by observing the logs, erasing which may require additional administrative rights depending on how the gateway is configured. On the other hand, if the GKE protocol is vulnerable to KCI attacks, the adversary can impersonate a legitimate mobile device and gain access to the wireless network without being detected. We indeed present a KCI attack on the protocol of Bresson et al. [10].

Outsider KCI Resilience. We call a *party* corrupted if the long-term private key of the party is compromised, while a *protocol instance* is called corrupted if the ephemeral session state of that instance is revealed. In an outsider KCI attack scenario for GKE protocol, an adversary $\mathcal{A}$ is allowed to compromise the long-term private keys of up to all parties except one. But, it is allowed neither to corrupt the protocol instances at any of parties nor to participate in the protocol on behalf of the corrupted parties. $\mathcal{A}$ is an outsider to the specific protocol execution in consideration as no session specific information is revealed. $\mathcal{A}$ is considered successful in mounting a KCI attack if it can impersonate any uncorrupted party to an uncorrupted instance at any of the corrupted parties. We consider the goal of $\mathcal{A}$ as an outsider is to break the confidentiality of the session key established. Hence, we modify the existing definition of AKE-security accordingly.

Insider KCI Resilience. A party is called an insider if the adversary corrupts the party and actively participates in the protocol on behalf of that party. In an insider KCI attack scenario, the goal of an adversary $\mathcal{A}$ is to impersonate an uncorrupted party B to an uncorrupted instance of a party A. Note that $\mathcal{A}$ is allowed to compromise the long-term private key of A, while all the parties except A and B can be insiders. We revise the existing definition of mutual authentication accordingly. It is easy to see that the revised definition implies mutual authentication with KCI resilience in the presence of no insiders.

Boyd and González Nieto (BG) proposed a one-round GKE protocol and proved the protocol secure under the AKE-security notion. Choo et al. [11] later presented unknown key share attacks on the BG protocol and suggested an improvement. We show that the improved protocol is not outsider KCI resilient. We also show that the tripartite key agreement protocol TAK-3 of Al-Riyami and Paterson [12] and the GKE protocol of Bresson et al. [10] are not secure against outsider KCI attacks.

One way to modify the BG protocol to make it secure against KCI attacks is by applying the compiler of Katz and Shin (KS) [4]. A KS-compiled protocol is shown to guarantee mutual authentication in the presence of insiders. If one applies the KS-compiler to the improved BG protocol it is easy to show that the resulting protocol will be both outsider and insider KCI resilient in the random oracle model. However the resulting two-round protocol will not provide forward secrecy as the BG protocol itself does not have this property. Hence, we show that the two-round protocol of Bohli et al. [5], which already has forward secrecy, satisfies our new definitions. For the sake of completeness, we also show that this protocol is secure under the notion of contributiveness proposed by Bresson and Manulis [6]. The contributions of this paper are:

1. Modeling KCI attacks on GKE protocols by presenting new outsider and insider security notions

2. KCI attacks on the protocols of Boyd and González Nieto [13], Al-Riyami and Paterson [12] and Bresson et al. [10]
3. A new proof of security for the protocol of Bohli et al. [5] in the random oracle model

Organization. In Section 2 we present new notions of AKE-security and mutual authentication considering KCI attacks by outsiders and insiders respectively. Section 3 presents outsider KCI attacks on the improved Boyd and González Nieto's protocol, Al-Riyami and Paterson's protocol and Bresson et al.'s protocol. In Section 4, we show that the protocol of Bohli et al. [5] is insider secure i.e. that it satisfies the new notions AKE-security and mutual authentication in addition to existing notion of contributiveness. We conclude our paper in Section 5 with a comparison among existing GKE protocols.

2 Model

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of n parties. The protocol may be run among any subset of these parties. Each party is assumed to have a pair of long-term public and private keys, (PK_U, SK_U) generated during an initialization phase prior to the protocol run. A GKE protocol π executed among n users is modeled as a collection of n programs running at the n different parties in $\mathcal{U}$. Each instance of π within a party is defined as a session and each party may have multiple such sessions running concurrently.

Let π_U^i be the i-th run of the protocol π at party $U \in \mathcal{U}$. Each protocol instance at a party is identified by a unique session ID. We assume that the session ID is derived during the run of the protocol. The session ID of an instance π_U^i is denoted by sid_U^i. We assume that each party knows who the other participants are for each protocol instance. The partner ID pid_U^i of an instance π_U^i, is a set of identities of the parties with whom π_U^i wishes to establish a common group key. Note that pid_U^i includes the identity of U itself.

An instance π_U^i enters an *accepted* state when it computes a session key sk_U^i. Note that an instance may terminate without ever entering into an accepted state. The information of whether an instance has terminated with acceptance or without acceptance is assumed to be public. Two instances π_U^i and $\pi_{U'}^j$ at two different parties U and U' respectively are considered *partnered* iff (1) both the instances have accepted, (2) $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$ and (3) $\mathsf{pid}_U^i = \mathsf{pid}_{U'}^j$.

The communication network is assumed to be fully controlled by an adversary $\mathcal{A}$, which schedules and mediates the sessions among all the parties. $\mathcal{A}$ is allowed to insert, delete or modify the protocol messages. If the adversary honestly forwards the protocol messages among all the participants, then all the instances are partnered and output identical session keys. Such a protocol is called a correct GKE protocol. In addition to controlling the message transmission, $\mathcal{A}$ is allowed to ask the following queries.

- $\mathsf{Execute}(\mathsf{sid},\mathsf{pid})$ prompts a complete execution of the protocol among the parties in pid using the unique session ID sid. $\mathcal{A}$ is given all the protocol messages, modeling passive attacks.

- $\mathsf{Send}(\pi_U^i, m)$ sends a message m to the instance π_U^i. If the message is $(\mathsf{sid}, \mathsf{pid})$, the instance π_U^i is initiated with session ID sid and partner ID pid. The response of π_U^i to any Send query is returned to $\mathcal{A}$.
- $\mathsf{RevealKey}(\pi_U^i)$ If π_U^i has accepted, $\mathcal{A}$ is given the session key sk_U^i established at π_U^i.
- Corrupt(U) The long-term secret key SK_U of U is returned to $\mathcal{A}$. Note that this query returns neither the session key (if computed) nor the internal state.
- $\mathsf{RevealState}(\pi_U^i)$ The internal state of U is returned to $\mathcal{A}$. We assume that the internal state is erased once π_U^i has accepted. Hence, a RevealState query to an accepted instance returns nothing.
- $\mathsf{Test}(\pi_U^i)$ A random bit b is secretly chosen. If $b = 1$, $\mathcal{A}$ is given sk_U^i established at π_U^i. Otherwise, a random value chosen from the session key probability distribution is given. Note that a Test query is allowed only on an accepted instance.

2.1 AKE Security

We present a revised notion of AKE-security by taking KCI attacks into account. As this is a notion of outsider security, we assume that all the participants are honest i.e. all the parties execute the protocol honestly.

The notion of freshness is central to the definition of AKE-security. We define the notion of freshness by considering KCI attack scenarios based on a corresponding notion for two-party key exchange given by Krawczyk [8]. Informally, a session is considered fresh if the session key is not trivially compromised.

Freshness. An instance π_U^i is fresh if the following conditions hold:

1. the instance π_U^i or any of its partners has not been asked a RevealKey after their acceptance
2. the instance π_U^i or any of its partners has not been asked a RevealState before their acceptance
3. If $\pi_{U'}^j$ is a partner of π_U^i and $\mathcal{A}$ asked $\mathsf{Corrupt}(U')$, then any message that $\mathcal{A}$ sends to π_U^i on behalf of $\pi_{U'}^j$ must come from $\pi_{U'}^j$ intended to π_U^i .

The last condition requires that the adversary be an outsider, i.e. it must be passive for any partner that it corrupts.

Definition 1 (AKE-Security with KCI resilience)**.** An adversary $\mathcal{A}_{ake}$ against the AKE-security notion is allowed to make Execute, Send, RevealState, RevealKey and Corrupt queries in Stage 1. $\mathcal{A}_{ake}$ makes a Test query to an instance π_U^i at the end of Stage 1 and is given a challenge key K_b as described above. It can continue asking queries in Stage 2. Finally, $\mathcal{A}_{ake}$ outputs a bit b' and wins the AKE security game if (1) $b' = b$ **and** (2) the instance π_U^i that was asked Test query remained fresh till the end of $\mathcal{A}_{ake}$'s execution. Let $\mathsf{Succ}_{\mathcal{A}_{ake}}$ be the success probability of $\mathcal{A}_{ake}$ in winning the AKE security game. The advantage of $\mathcal{A}_{ake}$ in winning this game is $\mathsf{Adv}_{\mathcal{A}_{ake}} = |2 \cdot \Pr[\mathsf{Succ}_{\mathcal{A}_{ake}}] - 1|$. A protocol is called AKE-secure if $\mathsf{Adv}_{\mathcal{A}_{ake}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{ake}$.

The definition of freshness takes care of the KCI attacks as it does allow $\mathcal{A}_{ake}$ to corrupt the owner of the test protocol instance. Note that if the adversary is active with respect to a partner to the test instance π_U^i , then that party cannot have been corrupted, otherwise π_U^i is not fresh. The definition also takes forward secrecy into account as it allows $\mathcal{A}_{ake}$ to obtain the long-term private keys of all the parties. In this case, $\mathcal{A}_{ake}$ must be passive with respect to all partners of π_U^i.

2.2 Mutual Authentication

Katz and Shin [4] first presented a definition of insider security that models impersonation attacks and ensures agreement on the session key in the presence of insiders. Bohli et al. [5] revisited this notion in weak corruption model, where session state is not revealed. They also presented insider attacks on the protocols of Katz and Yung [14] and Kim et al. [15] that violate integrity of the protocols. Later, Bresson and Manulis [6] unified the insider security notions of Katz and Shin into their definition of mutual authentication. They also considered session state reveal queries separately from the corrupt queries. We strengthen the definition of Bresson and Manulis by considering KCI attacks by insiders.

Definition 2 (Mutual Authentication with KCI resilience)**.** An adversary $\mathcal{A}_{ma}$ against the mutual authentication of a correct GKE protocol π is allowed to ask $\mathsf{Execute}$, Send, $\mathsf{RevealState}$, $\mathsf{RevealKey}$ and $\mathsf{Corrupt}$ queries. $\mathcal{A}_{ma}$ violates the mutual authentication property of the GKE protocol if at some point during the protocol run, there exists an uncorrupted instance π_U^i (although the party U may be corrupted) that has accepted with a key sk_U^i and another party $U' \in \mathsf{pid}_U^i$ that is uncorrupted at the time π_U^i accepts such that

1. there is no instance $\pi_{U'}^j$ with $(\mathsf{pid}_{U'}^j, \mathsf{sid}_{U'}^j) = (\mathsf{pid}_U^i, \mathsf{sid}_U^i)$ **or**
2. there is an instance $\pi_{U'}^j$ with $(\mathsf{pid}_{U'}^j, \mathsf{sid}_{U'}^j) = (\mathsf{pid}_U^i, \mathsf{sid}_U^i)$ that has accepted with $sk_{U'}^j \neq sk_U^i$.

Let $\mathsf{Succ}_{\mathcal{A}_{ma}}$ be the success probability of $\mathcal{A}_{ma}$ in winning the mutual authentication game. A protocol is said to provide mutual authentication in the presence of insiders if $\mathsf{Succ}_{\mathcal{A}_{ma}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{ma}$.

The difference between the above definition and that of Bresson and Manulis is that we allow $\mathcal{A}_{ma}$ to obtain the long-term private key of U_i, but $\mathcal{A}_{ma}$ is not allowed to execute the protocol on U_i's behalf. $\mathcal{A}_{ma}$ is considered successful in an insider KCI attack against U_i if it violates above definition of mutual authentication.

It is easy to see that if a protocol does not satisfy earlier definitions [4,5,6], it also does not satisfy Definition 2. Many existing protocols [5,6] which are proven secure under the definitions in the corresponding papers also seem to satisfy our definition. Note that this is not the general case as the adversary in our definition clearly has additional power.

2.3 Contributiveness

To ensure complete covering of insider security notions we present below the notion of contributiveness by Bresson and Manulis [6]. A GKE protocol secure under this notion resists the key control attacks by Pieprzyk and Wang [16] where a proper subset of insiders tries to predetermine the resulting session key.

Definition 3 (Contributiveness)**.** An adversary $\mathcal{A}_{con}$ against the contributiveness of correct GKE protocol π is allowed ask Execute, Send, RevealKey, RevealState and Corrupt queries. It operates in two stages prepare and attack as follows:

prepare. $\mathcal{A}_{con}$ queries the instances of π and outputs some state information ζ along with a key $\tilde{k}$.

At the end of prepare stage, a set Π is built such that Π consists of uncorrupted instances which have been asked either Execute or Send queries.

attack. On input (ζ, Π), $\mathcal{A}_{con}$ interacts with the instances of π as in the prepare stage.

At the end of this stage $\mathcal{A}_{con}$ outputs (U, i) and wins the game if an instance π_U^i at an uncorrupted party U has terminated accepting $\tilde{k}$ with $\pi_U^i \notin \Pi$.

Let $\mathsf{Succ}_{\mathcal{A}_{con}}$ be the success probability of $\mathcal{A}_{con}$ in winning the above game. A protocol is said to provide contributiveness in the presence of insiders, if $\mathsf{Succ}_{\mathcal{A}_{con}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{con}$.

3 KCI Attacks on Existing Protocols

We present KCI attacks on the protocols of Boyd and González Nieto [13], Al-Riyami and Paterson [12] and Bresson et al. [10]. We speculate that there are many GKE protocols in the literature which are not secure against KCI attacks. By selecting these three protocols, we are able to demonstrate the importance of considering resilience to KCI attacks for GKE protocols under different setup assumptions. Note that the Boyd and González Nieto protocol is a contributory GKE protocol where each party is assumed to have equal resources. The Al-Riyami and Paterson protocol is a GKE protocol with the group size three, while the protocol of Bresson et al. assumes a server with high computational resources and many computationally restricted clients.

3.1 Boyd and González Nieto's Protocol [13]

Boyd and González Nieto [13] (BG) proposed a one-round GKE protocol and proved it AKE-secure in the Bellare-Rogaway model [17] adapted to the group setting. Later, Choo et al. [11] presented an unknown key share attack on the BG protocol in a multi-user setting. They also presented an improved BG protocol that resists unknown key share attacks but do not give any formal security proof. We now briefly describe the protocol.

Round 1

$U_1 \rightarrow * : \mathcal{U} = \{U_1, U_2, \ldots, U_n\}, \sigma_{sk_{s_1}}(\mathcal{U}, \{N_1, U_1\}_{pk_{e_2}}, \ldots, \{N_1, U_1\}_{pk_{e_n}})$

$U_1 \rightarrow * : \{N_1, U_1\}_{pk_{e_i}}$ for $1 < i \leq n$

$U_i \rightarrow * : U_i, N_i$ for $1 < i \leq n$

Key Computation

$sid = \mathcal{U} \| \sigma_{sk_{s_1}}(\mathcal{U}, \{N_1, U_1\}_{pk_{e_2}}, \ldots, N_1, U_1\}_{pk_{e_n}}) \| \{N_1, U_1\}_{pk_{e_i}} \| U_i \| N_i$

The session key is $SK_i^U = \mathcal{H}(N_1 \| sid)$

Fig. 1. Improved Boyd-González Nieto Protocol [11]

Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participants. All the users agree upon a distinguished user for each execution of the protocol. Without loss of generality let U_1 be the distinguished user. The protocol uses a public key encryption scheme $\mathcal{PE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$, where $\mathcal{K}_e$, $\mathcal{E}$ and $\mathcal{D}$ are the key generation, encryption and decryption algorithms. It also uses a signature scheme $\Sigma = (\mathcal{K}_s, \mathcal{S}, \mathcal{V})$, where $\mathcal{K}_s$, $\mathcal{S}$ and $\mathcal{V}$ are the key generation, signature and verification algorithms. Each user is issued with a key pair for each of the schemes. Let (sk_e, pk_e) and (sk_s, pk_s) be the private-public key pairs for the encryption and signature schemes.

In the protocol, the distinguished user U_1 chooses a nonce $N_1 \overset{R}{\leftarrow} \{0,1\}^k$ and encrypts it along with its identity for each of the other parties. U_1 signs all these ciphertexts together with the set of identities of all the users $\mathcal{U}$. The set $\mathcal{U}$, the signature computed and the ciphertexts are then broadcast. All the parties $U_i \in \mathcal{U}, U_i \neq U_1$ broadcast their nonces $N_i \overset{R}{\leftarrow} \{0,1\}^k$ along with their identities. A user computes the session ID as the concatenation of all the outgoing and incoming protocol messages. A key derivation function $\mathcal{H}$ is used to compute the session key with the nonce N_1 and the session ID as input. As there is no restriction on who should send a protocol message first, the protocol can be completed in one round. The protocol message transmission and session key computation are presented in Figure 1. The users U_i, $i \neq 1$, verify the signature of U_1 and decrypt k_1 before computing the session key.

We show that the improved BG protocol in Figure 1 is not secure against KCI attacks. An attack can be mounted by corrupting any user except the distinguished user U_1. Let us assume that U_2 is corrupted. An adversary $\mathcal{A}$ can impersonate U_1 just by replaying a message from a previous successful execution of the protocol. The nonce selected by U_1 in the replayed message can be decrypted using the private key of U_2. Thus $\mathcal{A}$ can easily win the AKE-security game by selecting the test session at U_2.

A straightforward improvement to the protocol in Figure 1 could be by asking all users $U_i \neq U_1$ to encrypt their nonces with the public keys of other users and broadcast the messages. Although this thwarts the above KCI attack on the protocol, the improved protocol cannot be proven secure under the AKE-security notion. To see why, note that in the above attack scenario we have considered the simple case where the long-term private key of only one user other than U_1 is compromised. If we assume that more than one user (other than U_1) is corrupted, then the adversary can impersonate U_1 in the same way as described above and successfully mount a KCI attack. We leave open the task of constructing a one-round GKE protocol that resists KCI attacks.

3.2 Al-Riyami and Paterson's Protocol [12]

Al-Riyami and Paterson [12] proposed a series of tripartite key agreement (TAK) protocols based on Joux's protocol [18]. While the authors do not provide a definition of KCI resilience for a TAK protocol, they claim that the protocol TAK-3 is secure against KCI attacks. However we below present a KCI attack on TAK-3.

The system parameters are $(q, \mathbb{G}_1, \mathbb{G}_T, P, e, H)$, where q is a prime number, $\mathbb{G}_1$ and $\mathbb{G}_T$ are groups of order q, P is a generator of $\mathbb{G}_1$, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is an admissible bilinear map and H is a hash function that maps to the key space. Let (x, xP), (y, yP) and (z, zP) be the private-public key pairs of three users A, B and C respectively, where $x, y, z \in \mathbb{Z}_q^*$. The parties are issued certificates for their public key, which bind an identity to the corresponding public key. Let $Cert_A$, $Cert_B$ and $Cert_C$ be the certificates issued for the public keys of A, B and C respectively.

As the part of the protocol the users A, B and C select the ephemeral secret keys $a, b, c \overset{R}{\leftarrow} \mathbb{Z}_q^*$ respectively. The protocol message transmission and key computation is shown in Figure 2.

Round 1

$A \rightarrow B, C : aP \| Cert_A$
$B \rightarrow A, C : bP \| Cert_B$
$C \rightarrow A, B : cP \| Cert_C$

Key Computation:

$A : K_A = e(yP, cP)^x \cdot e(bP, zP)^x \cdot e(yP, zP)^a \cdot e(bP, cP)^a$
$B : K_B = e(aP, zP)^y \cdot e(xP, cP)^y \cdot e(xP, zP)^b \cdot e(aP, cP)^b$
$C : K_C = e(aP, yP)^z \cdot e(xP, bP)^z \cdot e(xP, yP)^c \cdot e(aP, bP)^c$

The session key is $K_{ABC} = K_A = K_B = K_C = e(P, P)^{(xy)c+(xz)b+(yz)a+abc}$

Fig. 2. TAK-3 protocol of Al-Riyami and Paterson with forward secrecy [12]

We now show that the protocol in Figure 2 is not KCI resilient as per our AKE-security definition. Let us assume that the adversary $\mathcal{A}$ has compromised the long-term private keys x and y of the parties A and B respectively. $\mathcal{A}$ can impersonate an honest user C by sending a message $c'P \| Cert_C$ for a known $c' \in \mathbb{Z}_q^*$. It can compute the same key that A and B computes with its knowledge of x, y and c' as $K' = e(yP, c'P)^x \cdot e(bP, zP)^x \cdot e(aP, zP)^y \cdot e(aP, bP)^{c'}$. It can now easily win the AKE-security game by selecting a test session at either A or B.

The key derivation of TAK-3 protocol is similar to the MTI/A0 protocol [19]. Al-Riyami and Paterson also proposed another tripartite variant TAK-4 whose key derivation is based on the MQV [7]. The two-party protocols MTI/A0 and the MQV are secure against KCI attacks. It is interesting to see that TAK-3 protocol is vulnerable to KCI attacks while TAK-4 protocol appears to resist them.

3.3 Bresson et al.'s Protocol [10]

In the protocol of Bresson et al. [10], a group of n parties computes a common session key with a mobile gateway S acting as a server. The system parameters are $(q, \mathbb{G}, g, \mathcal{H}, \mathcal{H}_0, \mathcal{H}_1, k_0, k_1)$, where q is a prime number chosen based on a security parameter k, $\mathbb{G}$ is a finite cyclic group of order q, g is an arbitrary generator of $\mathbb{G}$. The hash functions $\mathcal{H}, \mathcal{H}_0$ and $\mathcal{H}_1$ map to bit strings of length k, k_0 and k_1 respectively. The server is assumed to have a private-public key pair $(x, y = g^x)$ where $x \overset{R}{\leftarrow} \mathbb{Z}_q^*$. It also assumed to know the group of parties $\mathcal{G}_c$ with whom it communicates. Each party U_i is issued a private-public key pair (SK_i, PK_i) for a signature scheme $\mathsf{SIGN} = (\mathsf{SIGN.KGen}, \mathsf{SIGN.Sig}, \mathsf{SIGN.Ver})$. The protocol execution is described in Figure 3.

Round 1:

1. Each U_i selects $x_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$, computes $y_i = g_i^x$, $\alpha_i = y^{x_i}$ and $\sigma_1 = \mathsf{SIGN.Sig}(SK_i, y_i)$
2. Each U_i sends (y_i, σ_i) to the server S

Round 2:

1. The base station first verifies all the incoming signatures.
2. It then computes $\alpha_i = y_i^x$, initializes a counter $c \in \{0,1\}^{k_1}$ and the shared secret key $K = \mathcal{H}_0(c\|\alpha_1\|\ldots\|\alpha_n)$
3. It also computes for each party U_i, $K_i = K \oplus \mathcal{H}_1(c\|\alpha_i)$
4. S sends (c, K_i) to the user U_i for all $i \in [1, n]$

Key Computation:

1. Each user computes $K = K_i \oplus \mathcal{H}_1(c\|\alpha_i)$
2. The session key is computed by the server and the parties as $sk = \mathcal{H}(K\|\mathcal{G}_c\|S)$

Fig. 3. Bresson et al.'s GKE protocol [10]

We now show that the protocol in Figure 3 is not secure against KCI attacks as per Definition 1. If an adversary $\mathcal{A}$ obtains the long-term private key x of the server S, it can impersonate any honest user in $\mathcal{G}_c$ to S as follows: $\mathcal{A}$ simply replays a message (y_i, σ_i) of party U_i from an earlier successful execution of the protocol. The server sends back (c, K_i). $\mathcal{A}$ can compute the shared secret K as $K = K_i \oplus \mathcal{H}_1(c\|\alpha_i)$ where α_i is computed as y_i^x with its knowledge of the private key x. Thus it can win the AKE-security game by choosing a test session at S.

4 An Insider Secure GKE Protocol

Bohli et al. [5] showed that the protocol of Kim et al. [15] was insecure in the presence of insiders and then modified the protocol as shown in Figure 4. The improved protocol was shown to satisfy their definitions of outsider and insider security. We briefly review the protocol here.

Let $\{U_1, \ldots, U_n\}$ be the set of parties who wish to establish a common group key. It is assumed that the parties are ordered in a logical ring with U_{i-1} and

U_{i+1} being the left and right neighbors of U_i for $1 \leq i \leq n$, $U_0 = U_n$ and $U_{n+1} = U_1$. During the initialization phase, a cyclic group $\mathbb{G}$ of prime order q, an arbitrary generator g of $\mathbb{G}$ and the description of a hash function H that maps to $\{0,1\}^k$ are chosen. Each party is assumed to have a long-term private and public key pair for a public key signature scheme. Figure 4 outlines the execution of the protocol after initialization.

At a high level, the protocol in Figure 4 embeds the protocol of Boyd and González [13] in the first round of Burmester and Desmedt [20] (BD) protocol with the Katz and Yung [14] signature-based compiler applied. However, there are non-trivial and crucial changes done to the resulting protocol to enable it to achieve forward secrecy and contributiveness. As in the Boyd and González [13] (BG) protocol the parties choose their shares k_i's in the first round and all except one party send their shares in plain with the message broadcast in Round 1. Unlike the BG protocol, the nth user (or the distinguished user) sends only a commitment to its share instead of encrypting it with the long-term public keys of other users. The parties compute pair-wise CDH components using the y_i's sent in the first round similar to the BD protocol. These *ephemeral* values are used to encrypt the share of the distinguished user in the second round, which can be decrypted by the other users using the pair-wise CDH components they have computed. This enables the protocol to achieve forward secrecy unlike the BG protocol. The session key is finally computed in a way similar to the BG protocol using the shares from all the users, which guarantees contributiveness unlike the BD protocol [16]. The signature based authentication ensures security against impersonation attacks.

We now show that the protocol in Figure 4 is KCI resilient as per our new definitions and also contributory as per the definition of Bresson and Manulis [6].

Theorem 1. *The protocol in Figure 4 is AKE-secure as per Definition 1 assuming that the CDH assumption holds in $\mathbb{G}$, the signature scheme is UF-CMA secure and that H is a random oracle. The advantage of $\mathcal{A}_{ake}$ is upper bounded by*

$$2\left(n^2\mathsf{AdvCMA}_{\Sigma} + \frac{(3q_s+q_r)^2}{2^k} + \frac{q_s^2}{2^k} + nq_sq_r\mathsf{SuccCDH} + \frac{q_sq_r}{2^k}\right)$$

where n is the number of participants, AdvCMA_{Σ} *is the advantage of a polynomial adversary against the UF-CMA security of the signature scheme,* $\mathsf{SuccCDH}$ *is the probability of solving CDH in $\mathbb{G}$ and k is the security parameter. q_s and q_r are the upper bounds on the number of* Send *and random oracle queries respectively that $\mathcal{A}_{ake}$ can ask.*

Proof. We give the proof in a sequence of games. Let S_i be the event that $\mathcal{A}_{ake}$ wins the AKE-security game in Game i and τ_i be the advantage of $\mathcal{A}_{ake}$ in Game i i.e. $\tau_i = |2 \cdot \Pr[S_i] - 1|$.

We use the following game hopping technique suggested by Dent [21] for indistinguishability games and recently used by Boyd et al. [22]. Consider an event E that may occur during $\mathcal{A}_{ake}$'s execution such that E is detectable by

Round 1:

Computation

1. Each U_i chooses $k_i \xleftarrow{R} \{0,1\}^k$, $x_i \xleftarrow{R} \mathbb{Z}_q$ and computes $y_i = g^{x_i}$. U_n additionally computes $H(k_n)$
2. Each U_i except U_n sets $M_i^I = k_i \| y_i$, while U_n sets $M_n^I = H(k_n) \| y_n$
3. Each U_i computes a signature σ_i^I on $M_i^I \| \mathsf{pid}_i$.

Broadcast Each U_i broadcasts $M_i^I \| \sigma_i^I$.

Check Each U_i checks all signatures σ_j^I of incoming messages $M_j^I \| \sigma_j^I$ for $j \neq i$

Round 2:

Computation

1. Each U_i computes $t_i^L = H(y_{i-1}^{x_i})$, $t_i^R = H(y_{i+1}^{x_i})$, $T_i = t_i^L \oplus t_i^R$ and $\mathsf{sid}_i = H(\mathsf{pid} \| k_1 \| \ldots \| k_{n-1} \| H(k_n))$. U_n additionally computes $\mathsf{mask}_n = k_n \oplus t_n^R$.
2. Each U_i except U_n sets $M_i^{II} = T_i \| \mathsf{sid_i}$ while U_n sets $M_n^{II} = \mathsf{mask}_n \| T_n \| \mathsf{sid_n}$
3. Each U_i computes a signature σ_i^{II} on M_i^{II}.

Broadcast Each U_i broadcasts $M_i^{II} \| \sigma_i^{II}$.

Check

1. Each U_i verifies the incoming the signatures σ_j^{II} on the corresponding message M_j^{II} for each $j \in [1,n]$ and $j \neq i$ also checks that $T_1 \oplus \cdots \oplus T_n \stackrel{?}{=} 0$ and $\mathsf{sid}_i \stackrel{?}{=} \mathsf{sid}_j$
2. Each U_i for $i < n$, extracts $k_n = \mathsf{mask}_n \oplus T_1 \oplus \cdots \oplus T_{i-1} \oplus t_i^L$ and checks the commitment $H(k_n)$ sent in Round 1 for the k_n extracted.

Key Computation

Each U_i computes the session key $sk_i = H(\mathsf{pid}_i \| k_1 \| \ldots \| k_n)$

Fig. 4. GKE protocol of Bohli et al. [5]

the simulator, E is independent of S_i, Game i and Game $i+1$ are identical unless E occurs, and $Pr[S_{i+1}|E] = \frac{1}{2}$. Then we have:

$$\Pr[S_{i+1}] = \Pr[S_{i+1}|E]\Pr[E] + \Pr[S_{i+1}|\neg E]\Pr[\neg E] \tag{1}$$

$$= \frac{1}{2}\Pr[E] + \Pr[S_i|\neg E]\Pr[\neg E] \tag{2}$$

$$= \frac{1}{2}(1 - \Pr[\neg E]) + \Pr[S_i]\Pr[\neg E] \tag{3}$$

$$= \frac{1}{2} + \Pr[\neg E]\left(\Pr[S_i] - \frac{1}{2}\right) \tag{4}$$

$$\text{Hence } \tau_{i+1} = 2|\Pr[S_{i+1}] - \frac{1}{2}| = 2|\Pr[\neg E]\left(\Pr[S_i] - \frac{1}{2}\right)| \tag{5}$$

$$= \Pr[\neg E]\tau_i \tag{6}$$

Game 0. This is the original AKE-security game as per the Definition 1. By definition we have

$$Adv_{\mathcal{A}_{ake}} = |2 \cdot \Pr[S_0] - 1| = \tau_0 \tag{7}$$

Game 1. This is the same as the previous game except that the simulation fails if an event Forge occurs. Hence

$$|\Pr[S_1] - \Pr[S_0]| \le \Pr[\mathsf{Forge}] \tag{8}$$

$$\begin{aligned} \tau_0 &= |2 \cdot \Pr[S_0] - 1| \le |2 \cdot \Pr[S_0] - 2 \cdot \Pr[S_1]| + |2 \cdot \Pr[S_1] - 1| && (9)\\ &\le 2\Pr[\mathsf{Forge}] + \tau_1 && (10) \end{aligned}$$

The event Forge occurs when $\mathcal{A}_{ake}$ issues a Send query with a message of the form (M_i, σ_i) such that U_i is not corrupted and the message has previously not been an output of an instance at U_i. Note that in a KCI attack, $\mathcal{A}_{ake}$ corrupts up to $n-1$ parties but it has to remain passive on behalf of the corrupted users. Hence Forge represents successful forgery of honest users' signatures.

If this event occurs we can use $\mathcal{A}_{ake}$ to forge a signature for a given public key in a chosen message attack as follows: The given public key is assigned to one of the n parties. All other parties are initialized as normal according to the protocol. All queries to the parties can be easily answered by following the protocol specification since all secret keys are known, except for the private key corresponding to the public key of the forgery attack game. In the latter case the signing oracle that is available as part of the chosen message attack can be used to simulate the answers.

The probability of $\mathcal{A}_{ake}$ not corrupting this party is $\ge \frac{1}{n}$. The probability of $\mathcal{A}_{ake}$ outputting a valid forgery on behalf of this user is also $\ge \frac{1}{n}$. Hence $\mathsf{AdvCMA}_{\Sigma} \ge \frac{1}{n^2} \cdot \Pr[\mathsf{Forge}]$. Rewriting the equation we have

$$\Pr[\mathsf{Forge}] \le n^2 \cdot \mathsf{AdvCMA}_{\Sigma} \tag{11}$$

Game 2. This game is the same as the previous game except that the simulation fails if an event Collision occurs.

$$|\Pr[S_2] - \Pr[S_1]| \le \Pr[\mathsf{Collision}] \tag{12}$$

$$\begin{aligned} \tau_1 &= |2 \cdot \Pr[S_1] - 1| \le |2 \cdot \Pr[S_1] - 2 \cdot \Pr[S_2]| + |2 \cdot \Pr[S_2] - 1| && (13)\\ &\le 2\Pr[\mathsf{Collision}] + \tau_2 && (14) \end{aligned}$$

The event Collision occurs when the random oracle H produces a collision for any of its inputs. Each Send query requires at most 3 queries to the random oracle. Hence the total number of random oracle queries are bounded by $(3q_s + q_r)$. The probability of Collision is

$$\Pr[\mathsf{Collision}] \le \frac{(3q_s + q_r)^2}{2^k} \tag{15}$$

Game 3. This game is the same as the previous game except that the simulation fails if an event Repeat occurs. Hence

$$|\Pr[S_3] - \Pr[S_2]| \le \Pr[\mathsf{Repeat}] \tag{16}$$

$$\tau_2 = |2 \cdot \Pr[S_2] - 1| \leq |2 \cdot \Pr[S_2] - 2 \cdot \Pr[S_3]| + |2 \cdot \Pr[S_3] - 1| \quad (17)$$

$$\leq 2 \Pr[\mathsf{Repeat}] + \tau_3 \quad (18)$$

The event Repeat occurs when an instance at a party U_i chooses a nonce k_i that was chosen by another instance at U_i. As there are a maximum q_s instances that may have chosen a nonce k_i, we have

$$\Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{2^k} \quad (19)$$

Game 4. This game is the same as the previous game except that at the beginning of the game a value $\bar{s}$ is chosen at random in $\{1, \ldots, q_s\}$, where q_s is an upper bound to the maximum number of protocol sessions activated by the adversary. $\bar{s}$ represents a guess as to the protocol session in which the adversary is going to be tested. If the adversary does not choose the $\bar{s}^{th}$ session to ask the Test query, then the guess it wrong and the game is aborted.

The probability of aborting due to an incorrect choice of $\bar{s}$ is $1 - 1/q_s$. This event could be detected in the previous game if it also chose $\bar{s}$ in the same way. Therefore from Equation 6 we have

$$\tau_4 = \frac{1}{q_s}\tau_3 \implies \tau_3 = q_s\tau_4 \quad (20)$$

Game 5. This game differs from the previous game in how the Send queries are answered in the test session. Note that in this test session, the adversary is an outsider and moreover is passive with respect to all the parties. An active adversary producing valid signatures on behalf of uncorrupted parties would have caused Game 1 to halt.

In round 1 of the test session in Game 5, all messages y_i are chosen at random from $\mathbb{G}$. In round 2, all $t_i^R (= t_{i+1}^L)$ are assigned random values from $\{0,1\}^k$. All other computations are performed as in Game 4.

Since $H(\cdot)$ is modeled as a random oracle, the only way that any adversary can distinguish between Game 4 and 5 is if for at least one value of i it queries $y_i^{x_{i+1}}$ $(= y_{i+1}^{x_i})$ to the random oracle, where x_i and x_{i+1} are discrete logs of y_i and y_{i+1} respectively. Let Ask be such an event.

$$|\Pr[S_5] - \Pr[S_4]| \leq \Pr[\mathsf{Ask}] \quad (21)$$

$$\tau_4 = |2 \cdot \Pr[S_4] - 1| \leq |2 \cdot \Pr[S_4] - 2 \cdot \Pr[S_5]| + |2 \cdot \Pr[S_5] - 1| \quad (22)$$

$$\leq 2 \Pr[\mathsf{Ask}] + \tau_5 \quad (23)$$

If Ask occurs, we can use $\mathcal{A}_{ake}$ to solve the CDH problem in $\mathbb{G}$. Given a CDH instance $(g, A = g^a, B = g^b)$, this can be plugged into a simulation of Game 5 as follows. Firstly, choose at random a party in the test session U_i. Then for the test session assign $y_i = A$ and $y_{i+1} = B$. If the event Ask occurs, the probability that a randomly chosen entry Z, from the random oracle table is a pair-wise CDH is at least $\frac{1}{q_r}$. Further, the probability of Z being the correct solution to the given instance (g, g^a, g^b) is at least $\frac{1}{n}$. Hence, $\mathsf{SuccCDH} \geq \frac{1}{nq_r} \Pr[\mathsf{Ask}]$. Rewriting the equation we have

$$\Pr[\mathsf{Ask}] \leq nq_r \mathsf{SuccCDH} \tag{24}$$

Game 6. This game is the same as the previous game except that in the test session the game halts if $\mathcal{A}_{ake}$ asks a H-query with the corresponding input $(\mathsf{pid}_i \| k_1 \| \ldots \| k_n)$.

Because the protocol messages in round 2 of the test session carry no information about k_n, the best any adversary can do is to guess k_n with a probability $\frac{1}{2^k}$. Hence, the probability that $\mathcal{A}_{ake}$ asks the right H-query for the test session is at most $\frac{q_r}{2^k}$.

$$|\Pr[S_6] - \Pr[S_5]| \leq \frac{q_r}{2^k} \tag{25}$$

$$\tau_5 = |2 \cdot \Pr[S_5] - 1| \leq |2 \cdot \Pr[S_6] - 2 \cdot \Pr[S_6]| + |2 \cdot \Pr[S_6] - 1| \tag{26}$$

$$\leq 2\frac{q_r}{2^k} + \tau_6 \tag{27}$$

If the adversary does not query the random oracle H on the correct input, then the adversary has no advantage in distinguishing the real session key from a random one and so

$$\tau_6 = 0.$$

By combining equations 7 to 25, we have the claimed advantage of $\mathcal{A}_{ake}$, which is negligible in k. □

Theorem 2. *The protocol in Figure 4 satisfies mutual authentication as per Definition 2 assuming that the signature scheme is UF-CMA secure and that H is a random oracle. The advantage of $\mathcal{A}_{ma}$ is upper bounded by*

$$n^2 \cdot \mathsf{AdvCMA}_{\Sigma} + \frac{(3q_s + q_r)^2}{2^k} + \frac{q_s^2}{2^k}$$

where n is the number of participants, AdvCMA_{Σ} is the advantage of a polynomial adversary against the UF-CMA security of the signature scheme and k is the security parameter. q_s and q_r are the upper bounds on the number of Send *and random oracle queries respectively that $\mathcal{A}_{ma}$ can ask.*

Proof. We give the proof in a sequence of games. Let S_i be the event that $\mathcal{A}_{ma}$ violates the mutual authentication definition in Game i.

Game 0. This is the original mutual authentication game as per the Definition 2. By definition we have

$$Adv_{\mathcal{A}_{ma}} = \Pr[S_0] \tag{28}$$

Game 1. This game is the same as the previous game except that the simulation fails if an event Forge occurs, where Forge is the same event described in Game 1 of Theorem 1.

$$|\Pr[S_1] - \Pr[S_0] \leq \Pr[\mathsf{Forge}] \leq n^2 \cdot \mathsf{AdvCMA}_{\Sigma} \tag{29}$$

Game 2. This game is the same as the previous game except that the simulation fails if an event $\mathsf{Collision}$ occurs, where $\mathsf{Collision}$ is the same event described in Game 2 of Theorem 1.

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{Collision}] \leq \frac{(3q_s + q_r)^2}{2^k} \tag{30}$$

Game 3. This game is the same as the previous game except that the game now aborts if an event Repeat occurs, where Repeat is the same event described in Game 3 of Theorem 1.

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{2^k} \tag{31}$$

If Game 3 does not abort, all the honest partnered parties compute the same key. Hence, $\Pr[S_3] = 0$. By combining the Equations 28 to 31, we have the claimed advantage of $\mathcal{A}_{ma}$, which is negligible in k. □

Theorem 3. *The protocol in Figure 4 satisfies contributiveness as per Definition 3 assuming that H is a random oracle. The advantage of $\mathcal{A}_{con}$ is upper bounded by*

$$\frac{q_s^2 + 2 \cdot q_r}{2^k}$$

where n is the number of participants and k is the security parameter. q_s and q_r are the upper bounds on the number of Send *and random oracle queries respectively that $\mathcal{A}_{ake}$ can ask.*

Proof. We give the proof in a sequence of games. Let S_i be the event that $\mathcal{A}_{con}$ violates the definition of contributiveness in Game i.

Game 0. This is the original game of contributiveness and as per the Definition 3. By definition we have

$$Adv_{\mathcal{A}_{con}} = \Pr[S_0] \tag{32}$$

Game 1. This game is the same as the previous game except that the simulation fails if an event Repeat occurs, where Repeat is the same event described in Game 3 of Theorem 1.

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{2^k} \tag{33}$$

Game 2. This game is the same as the previous game except that the simulation fails if $\mathcal{A}_{con}$ can find a collision for the input k_n.

$$|\Pr[S_2] - \Pr[S_1]| \leq \frac{q_r}{2^k} \tag{34}$$

Game 3. This game is the same as the previous game except that the simulation fails if $\mathcal{A}_{con}$ finds a collision for the keying material input $(\mathsf{pid}_i \| k_1 \| \ldots \| k_n)$.

$$|\Pr[S_3] - \Pr[S_1]| \leq \frac{q_r}{2^k} \tag{35}$$

If Game 3 does not abort, the output of the random oracle is uniformly distributed. Hence, $\Pr[S_3] = 0$. By combining the equations 32 to 35, we have the claimed advantage of $\mathcal{A}_{con}$, which is negligible in k. □

5 Conclusion

Table 1 gives a comparison of the security of some of the existing GKE protocols. The terms "AKE" refers to AKE-security, "AKE-FS" refers to AKE-security with forward secrecy and "AKE-KCIR" refers to AKE-security with KCI resilience. Similarly "MA" refers to mutual authentication and "MA-KCIR" refers to mutual authentication with KCI resilience. The entry "Yes*" says that the corresponding protocol appears to be secure under the notion but there is no formal proof. The last column in the table says whether the protocol is proven in the random oracle model or in the standard model.

It can be observed from the table that only the two-round protocol of Bohli et al. is proven to satisfy all the desired notions of security in the random oracle model. The two-round protocol obtained after applying the KC-compiler to the

Table 1. Security comparison among existing GKE protocols

	AKE	AKE-FS	AKE-KCIR	MA	MA-KCIR	Contributiveness	Model
Boyd and González Nieto [13] (BG)	Yes	No	No	No	No	honest	ROM
Katz and Yung [14]	Yes	Yes	Yes*	honest	honest	No	Std.
Bresson et al. [10]	Yes	Yes	No	honest	honest	unknown	ROM
BG Protocol + KS-compiler [4]	Yes	No	Yes*	Yes*	Yes*	Yes*	ROM
Bohli et al. [5]	Yes	Yes	Yes	Yes	Yes	Yes	ROM
Bresson and Manulis [6]	Yes	Yes	Yes*	Yes	Yes*	Yes	Std.
Furukawa et al. [23]	Yes	Yes	Yes*	Yes	Yes*	unknown	Std.

improved Boyd and González Nieto protocol appears to satisfy all the desired notions except forward secrecy. Another two-round protocol that appears to resist KCI attacks is that of Furukawa et al. Their protocol is proven in the universal composability framework without assuming random oracles. It is not known whether the protocol of Furukawa et al. satisfies contributiveness in the presence of insiders. The protocol of Bresson and Manulis appears to resist KCI attacks and their protocol is also proven secure in standard model. However, it has three rounds of communication.

Our work models KCI attacks on GKE protocols in the presence of both outsiders and insiders. We have shown that there exist protocols which are not secure against KCI attacks. We have then shown that an existing protocol satisfies the new definitions. We hope that our work helps future protocols to be analyzed for KCI resilience.

We have not considered GKE protocols with special properties like robustness [24] and deniability [25]. Modeling KCI attacks on these types of protocols is an interesting open task. An open question is whether we can construct one-round AKE-secure GKE protocols that can have KCI resilience or forward secrecy. Constructing GKE protocols which do not use signature based authenticators seems to be another interesting problem.

Acknowledgement

This work has been supported by the Australian Research Council Discovery Project grant DP0773348.

References

1. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably authenticated group Diffie-Hellman key exchange. In: CCS 2001: Proceedings of the 8th ACM conference on Computer and Communications Security, pp. 255–264. ACM, New York (2001)
2. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290–309. Springer, Heidelberg (2001)
3. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
4. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security–CCS 2005, pp. 180–189. ACM, New York (2005)
5. Bohli, J.M., Gonzalez Vasco, M.I., Steinwandt, R.: Secure group key establishment revisited. Int. J. Inf. Sec. 6(4), 243–254 (2007)
6. Bresson, E., Manulis, M.: Securing Group Key Exchange against Strong Corruptions. In: Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS 2008), pp. 249–260. ACM Press, New York (2008)
7. Law, L., Menezes, A., Qu, M., Solinas, J.A., Vanstone, S.A.: An Efficient Protocol for Authenticated Key Agreement. Des. Codes Cryptography 28(2), 119–134 (2003)

8. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
9. Ng, E.M.: Security Models and Proofs for Key Establishment Protocols. Master's thesis, University of Waterloo (2005)
10. Bresson, E., Chevassut, O., Essiari, A., Pointcheval, D.: Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In: Proc. of MWCN 2003, pp. 59–62. World Scientific Publishing, Singapore (2003)
11. Choo, K.K.R., Boyd, C., Hitchcock, Y.: Errors in computational complexity proofs for protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)
12. Al-Riyami, S.S., Paterson, K.G.: Tripartite Authenticated Key Agreement Protocols from Pairings. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 332–359. Springer, Heidelberg (2003)
13. Boyd, C., González Nieto, J.M.: Round-Optimal Contributory Conference Key Agreement. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 161–174. Springer, Heidelberg (2002)
14. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
15. Kim, H.J., Lee, S.M., Lee, D.H.: Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 245–259. Springer, Heidelberg (2004)
16. Pieprzyk, J., Wang, H.: Key Control in Multi-party Key Agreement Protocols. In: Workshop on Coding, Cryptography and Combinatorics (CCC 2003), vol. 23. Progress in Computer Science and Applied Logic (PCS), pp. 277–288 (2003)
17. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
18. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
19. Matsumoto, T., Takashima, Y., Imai, H.: On seeking smart public-keydistribution systems. Trans. IECE of Japan E69, 99–106 (1986)
20. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System (Extended Abstract). In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
21. Dent, A.W.: A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260 (2006), http://eprint.iacr.org/
22. Boyd, C., Cliff, Y., González Nieto, J.M., Paterson, K.G.: Efficient One-Round Key Exchange in the Standard Model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 69–83. Springer, Heidelberg (2008)
23. Furukawa, J., Armknecht, F., Kurosawa, K.: A Universally Composable Group Key Exchange Protocol with Minimum Communication Effort. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 392–408. Springer, Heidelberg (2008)
24. Desmedt, Y., Pieprzyk, J., Steinfeld, R., Wang, H.: A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 459–475. Springer, Heidelberg (2006)
25. Bohli, J.M., Steinwandt, R.: Deniable Group Key Agreement. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 298–311. Springer, Heidelberg (2006)

Zero-Knowledge Proofs with Witness Elimination

Aggelos Kiayias[⋆] and Hong-Sheng Zhou[⋆]

Computer Science and Engineering
University of Connecticut
Storrs, CT, USA
{aggelos,hszhou}@cse.uconn.edu

Abstract. Zero-knowledge proofs with witness elimination are protocols that enable a prover to demonstrate knowledge of a witness to the verifier that accepts the interaction provided that the witness is valid for a given statement and additionally the witness **does not** belong to a set of *eliminated witnesses*. This set is determined by a public relation Q (that parameterizes the primitive) and the *private* input of the verifier. Zero-knowledge proofs with witness elimination thus call for a relaxation of the zero-knowledge property and are relevant in settings where a statement has a multitude of witnesses that may attest to its validity. A number of interesting issues arise in the design of such protocols that include whether a protocol transcript enables the verifier to test for witness after termination (something akin to an "offline dictionary attack") and whether the prover should be capable of understanding whether her witness is eliminated. The primitive is motivated by the setting of identification schemes where a user wishes to authenticate herself to an access point while preserving her anonymity and the access point needs to certify that the user is eligible while at the same time making sure she does not match the identity of a suspect user that is tracked by the authorities. We call such primitives anonymous identification schemes with *suspect tracking*.

In this work we formalize zero-knowledge proofs with witness elimination in the universal composability setting and we provide a general construction based on smooth projective hashing that is suitable for designing efficient schemes. As an illustration of our general construction we then present an explicit efficient scheme for proving knowledge of a Boneh-Boyen signature with witness elimination. Our scheme requires the design of a smooth projective hash function for the language of linear ElGamal ciphertexts. Along the way we demonstrate how zero-knowledge proofs with witness elimination naturally relate to the primitives of password-based key exchange and private equality testing.

1 Introduction

Zero-knowledge proofs were introduced in [23] and constitute a most useful cryptographic primitive. Given the wide applicability of the primitive several

[⋆] Research partly supported by NSF CAREER Award CNS-0447808.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 124–138, 2009.

generalizations of its basic formulation have been considered, typically by extending the basic set of security properties that are captured by a protocol implementation. These include parallel and concurrent composition of zero-knowledge [22,18], non-malleable zero-knowledge [17], resettable zero-knowledge protocols [9], monotone closure of zero-knowledge proofs [16], non-interactive zero-knowledge [2] and more recently isolated zero-knowledge proofs [15].

The security of the prover in a zero-knowledge proof is intended to be captured by demonstrating the existence of a simulator that is capable of generating transcripts indistinguishable from regular honest prover verifier interactions. A relaxation of the zero-knowledge property is the notion of witness indistinguishability [19] that requires instead that a malicious verifier cannot distinguish which one amongst two different possible witnesses the prover is using. Note that witness indistinguishability is useful as a notion only in cases where a number of distinct witnesses are associated with the same statement (while in cases when only a single witness exists the property is vacuous).

In this work we consider a different relaxation of the zero-knowledge property that we call *witness elimination.* A zero-knowledge proof with witness elimination with respect to a relation Q, enables the verifier to make sure that the witness employed by the prover does not satisfy $(w, w') \in Q$ where w' is of the choice of the verifier. Our typical example for the relation Q would be the equality relation: in this case, the verifier given a string w' of her choice can make sure that $w \neq w'$. Note that in case w' is public and the relation R for which the proof is executed is in NP, witness elimination can be resolved generically by having the prover engage in a proof that $((x, w) \in R) \wedge (w \neq w')$. Nevertheless witness elimination is interesting as a property in the case that the verifier does not wish to disclose to the prover which witness w' is she checking against.

It follows from the above discussion that zero-knowledge proofs with witness elimination as a primitive introduces a private input for the verifier for which, ideally, no information should be disclosed to the prover about it. A number of interesting questions arise when one seeks to construct protocols that solve the witness elimination problem for zero-knowledge proofs, in particular: (i) how many candidate witnesses should the verifier be able to test given a single protocol execution? (ii) should the verifier be able to test whether the prover possesses a valid witness? (this is relevant in the case where the witness has been eliminated) (iii) should the prover be allowed to extract the information that the verifier is eliminating her witness based on the protocol interaction? (note that the prover may realize this from a signal that is external to the protocol but this may or may not be provided depending on the application scenario – see below).

Our motivation for studying this primitive is its application to the identification protocol setting. Indeed, a notable application of zero-knowledge proofs is the efficient design of identification schemes that was exemplified in the work of Fiat and Shamir [20] and Schnorr [30]. Given such protocols, the ability to perform efficient conjunctions and disjunctions of zero-knowledge proof protocols [16,12] lead to the design of anonymous identification schemes [6]. Indeed, disjunction of identification protocols leads to a natural one-to-many relationship between a valid statement and the witnesses that attest to its validity something that in turn lends itself to the design of anonymous identification schemes. In particular, a prover may show that he belongs to a group of users provided that such group is identified by a directory of public-keys.

Witness elimination in this domain translates naturally to the following problem: the verifier wishes to make sure that the prover's identity does not match a suspect that is currently being tracked by authorities. Given that anonymity must still be preserved the verifier should have some way to make sure that the witness used by the prover is not equal to a suspect witness while not learning any further information about the witness of the prover. At the same time the protocol should not leak any information about the suspect identity which may be protected for the course of the investigation and potentially should never be revealed if sufficient evidence is not gathered. This is related to but at the same time different from user revocation in identification schemes where typically the identity of revoked users can be made public in the form of a CRL.

Our contributions are summarized as follows:

- We formalize the primitive of zero-knowledge with witness elimination in the Universal Composability setting [7,8]. In our security formalization we capture the following properties related to the three questions raised above: (i) A verifier can only test a single witness for each live protocol execution with the prover; this also ensures the prover that transcripts of her protocol executions at the present time cannot be tracked if she becomes a suspect in the future. This protects the privacy of individuals prior to the time when authorities receive a court order that enables their tracking. As a cryptographic property it also relates to protection against *offline dictionary attacks* in password based authenticated key exchange [25,21,10] as well as to the notion of *backwards unlinkability* of identification schemes [31,1,27] in the context of user revocation. (ii) In our protocols we may allow the verifier to have the power to test whether the witness of the prover is valid even in case that such witness is eliminated – this is natural as we view zero-knowledge proofs with witness elimination as an extension of the standard zero-knowledge proof functionality. (iii) We do not allow any information about the input of the verifier to be leaked to the prover from a protocol execution, i.e., the prover will be oblivious to what witnesses are eliminated by the verifier (independently of whether she is revoked or not). This protects the identity of the suspect in case no case is ever made against her by the authorities.
- We provide a general construction of zero-knowledge proofs with witness elimination that is suitable for building efficient implementations and is based on the notion of smooth projective hash functions, [13,21,24].
- We present an explicit practical construction of a zero-knowledge proof with witness elimination for the language of all non-adaptive Boneh-Boyen signatures [3]. This result immediately implies an anonymous identification scheme with suspect tracking. As part of our construction we also design a smooth projective hash function family for the language of Linear

ElGamal ciphertexts that is suitable in the bilinear group setting when the DDH assumption may not necessary hold.

Comparison with previous primitives. We note first that the problem of zero-knowledge with witness elimination can be viewed within the general context of secure two party computation (as it is also the case for regular zero-knowledge proofs). This means that one can apply generic techniques to obtain a solution for the functionality (e.g., using the protocol compiler of [11]). Still such generic techniques do not yield efficient protocols or much insight about the primitive something that further motivates the present investigation. Second, depending on the relation Q, zero-knowledge proofs with witness elimination may provide an inequality test for the private inputs of the prover and the verifier. It follows that this makes the resulting primitive similar to private equality tests [28] and password-based key exchange [25,21,10]. Note though that in the latter primitive it is also required to incorporate a private coin flipping component (that will produce the shared key) something that is not necessary in the witness elimination setting. On the other hand, the property that protocol transcripts of a password-based key exchange should not facilitate offline dictionary attacks corresponds to the property that the verifier should not be able to extract any new information from old witness elimination transcripts.

Looking at the primitive from the identification point of view, one can draw parallels to the primitive of traceable signatures [26] and verifier-local revocation group signatures [31,1,5]. The difference of these previous schemes compared to anonymous identification with suspect tracking is that in our case prior identification transcripts need to conceal the identity of the users even after the suspect witness is made known to the access points. This relates to the the property of backwards unlinkability that was introduced in [31,1,27]. We note that in these papers backwards unlinkability is only partially achieved as time is divided into epochs and within an epoch all user transcripts would be traceable. In contrast in our setting of anonymous identification with suspect tracking we explicitly require only live checking of suspects, i.e., an access point can check exactly one suspicion per protocol invocation and past protocol transcripts reveal no tracking information.

2 Preliminaries

Notation. We write $s \overset{\mathtt{r}}{\leftarrow} S$ to denote randomly selecting s from a finite set S, and $s \overset{\mathtt{d}}{\leftarrow} S$ to denote selecting s from a finite set S according to distribution $D(S)$.

Smooth Projective Hashing for Hard Partitioned Subset Membership Problems. Smooth projective hashing was introduced by Cramer and Shoup for constructing encryption schemes [13]. Later it was used to obtain efficient password-based authenticated key-exchange protocols [21] and further to achieve stronger security in the UC framework [10]. It is also used to build efficient oblivious transfer protocols [24]. Loosely speaking, a projective hashing family is a family of hash functions that can be computed in two ways: either use the (secret) hashing key to compute the function on every point in its domain, or use the (public) projected key to compute the function only on a specified subset of the domain. Further we say such a family is "smooth" if the value of the function on any point outside the specified subset is independent of the projected key. Here we use a variant adopted by Gennaro and Lindell [21].

In our setting, a hash family is defined as a tuple $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ that satisfies the following:

- The parameter generation algorithm $\mathsf{paragen}(\cdot)$ takes as input a security parameter and returns the parameter $\Lambda \leftarrow \mathsf{paragen}(1^\lambda)$, where $\Lambda = (X, L, W, R)$, and X, L, W are finite non-empty sets such that $L \subset X$. $R \subseteq X \times W$ is a binary relation, where for all $x \in X$ there exists $w \in W$ such that $(x, w) \in R$ iff $x \in L$.
- Further we assume that the set X can be written as a Cartesian product $C \times I$ and it is partitioned into disjoint subsets $X(i)$ as follows: each element $x \in X$ is in the form (c, i), where $i \in I$ will be called the index and $c \in C$ will be called the content. We define $X(i) \stackrel{\mathsf{def}}{=} C \times \{i\}$, i.e., the subset of pairs in X of the form (c, i) with i is fixed. Accordingly, we define $L(i)$ the subset of pairs in the language L of the form (c, i).
- Given Λ as a parameter, the random variable $\mathsf{sampler}(\Lambda, i)$ is a triple (x, x', w) such that $x \in L(i), x' \in X(i) \setminus L(i)$, $(x, w) \in R$. We will also use the notation $(x, w) \stackrel{\mathsf{d}}{\leftarrow} L(i)$ and $x' \stackrel{\mathsf{d}}{\leftarrow} X(i) \setminus L(i)$ respectively to denote the operation of the $\mathsf{sampler}(\cdot)$ procedure.
- G is a finite non-empty set, $H = \{H_k\}_{k \in K}$ is a collection of functions indexed by K where $H_k : X \to G$; S be a finite non-empty set, $\alpha : K \times C \to S$ and $J = \{J_s\}_{s \in S}$ is a collection of functions where $J_s : X \times W \to G$.

Definition 1. *The hash family* $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ *is a* hard smooth projective hash family *if the following properties hold:*

Projection: *For any* $((c, i), w) \in R$
$\Pr[\Lambda \leftarrow \mathsf{paragen}(1^\lambda); k \stackrel{\mathsf{r}}{\leftarrow} K; s \leftarrow \alpha(k, c) : H_k(c, i) = J_s(c, i, w)] \geq 1 - \mathsf{negl}(\lambda)$

Smoothness: *For any* $(c, i) \in X \setminus L$*, the following two distributions are statistically indistinguishable:*
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); k \stackrel{\mathsf{r}}{\leftarrow} K; s \leftarrow \alpha(k, c); g \leftarrow H_k(x) : (\Lambda, c, i, s, g)\}$
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); k \stackrel{\mathsf{r}}{\leftarrow} K; s \leftarrow \alpha(k, c); g \stackrel{\mathsf{r}}{\leftarrow} G : (\Lambda, c, i, s, g)\}$

Hard partitioned subset membership: *For every* $i \in I$*, the following two distributions are computationally indistinguishable:*
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); ((c, i), w) \stackrel{\mathsf{d}}{\leftarrow} L(i) : (\Lambda, c, i)\}$
$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); (c, i) \stackrel{\mathsf{d}}{\leftarrow} X(i) \setminus L(i) : (\Lambda, c, i)\}$

Gennaro and Lindell show that the properties above have the following implication:

Lemma 1 (Corollary 3.6 in [21]). *Given a hard smooth projective hash family* $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$*, for any* $i \in I$*, the following two distributions are computationally indistinguishable:*

$$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); ((c,i),w) \xleftarrow{\mathsf{d}} L(i); k \xleftarrow{\mathsf{r}} K; s \leftarrow \alpha(k,c); g \leftarrow H_k(c,i) : (\Lambda, c, i, s, g)\}$$

$$\{\Lambda \leftarrow \mathsf{paragen}(1^\lambda); ((c,i),w) \xleftarrow{\mathsf{d}} L(i); k \xleftarrow{\mathsf{r}} K; s \leftarrow \alpha(k,c); g \xleftarrow{\mathsf{r}} G : (\Lambda, c, i, s, g)\}$$

Hard Smooth Projective Hashing in the context of Public-Key Encryption. In our protocol design, we consider hard smooth projective hashing in the context of encryption schemes. Given a CPA-secure public key encryption scheme $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$, we define $\mathsf{paragen}$ and $\mathsf{sampler}$ algorithms for a hash family $\mathcal{H}$ as follows:

- The parameter generation algorithm $\mathsf{paragen}()$ operates as follows: first run a key pair $(pk, sk) \leftarrow \mathsf{gen}(1^\lambda)$; a ciphertext for a plaintext $m \in M$ can be computed by $c = \mathsf{enc}(pk; m; r)$ where $r \xleftarrow{\mathsf{r}} U$, and M is the plaintext space and U the random coins space; Let C_{pk} be the space of all ciphertexts based on public key pk; define $X \overset{\mathsf{def}}{=} C_{pk} \times M$, and $L \overset{\mathsf{def}}{=} \{(c,m) \in X \mid \exists r \text{ s.t. } c = \mathsf{enc}(pk; m; r)\}$. Note that the witness space $W = U$.

  Furthermore, define the partitioning of X to be by index $m \in M$, i.e., $X(m) \overset{\mathsf{def}}{=} C_{pk} \times \{m\}$, and $L(m) \overset{\mathsf{def}}{=} \{(c,m) \in X(m) \mid \exists r \text{ s.t. } c = \mathsf{enc}(pk; m; r)\}$.
- The sample procedure $\mathsf{sampler}()$ operates as follows: sample $((c,m),r) \in L(m)$ by computing $c \leftarrow \mathsf{enc}(pk; m; r)$ where $r \xleftarrow{\mathsf{r}} U$ is the witness; sample $(c,m) \in X(m) \backslash L(m)$ by computing $c \leftarrow \mathsf{enc}(pk; \tilde{m}; r)$ where $r \xleftarrow{\mathsf{r}} U$, $\tilde{m} \in M$, and $\tilde{m} \neq m$.

If we further can define (H, K, G, α, S, J) where $H_k : C_{pk} \times M \to G$, $J_s : C_{pk} \times M \times U \to G$, and $\alpha : K \times C_{pk} \to S$, to satisfy the properties of projection and smoothness, then we have a hard smooth projective hashing $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$. In [21], several concrete hard smooth projective hashing in the context of ElGamal encryption, Cramer-Shoup encryptions, and others are constructed. In section 3.3, we give a concrete construction in the context of Linear ElGamal encryption.

3 Zero-Knowledge with Witness Elimination

3.1 Functionality $\mathcal{F}_{\mathbf{ZKWE}}$

Here we introduce our new ZK functionality, *zero-knowledge with witness elimination*, $\mathcal{F}_{\text{ZKWE}}^{R,Q}$, in figure 1, where R is the ZK relation and Q is the elimination relation. The functionality interacts with an ideal adversary $\mathcal{S}$ and two parties, the prover P and the verifier T; the identities of both parties are encoded in session identifier *sid*. Intuitively $\mathcal{F}_{\text{ZKWE}}^{R,Q}$ can be viewed as an augmented ZK functionality with a "$\neg Q$-test" based on the private input of the verifier. Compared to the standard $\mathcal{F}_{\text{ZK}}$ functionality, here the functionality will receive from

Functionality $\mathcal{F}_{\mathrm{ZKWE}}^{R,Q}$

$\mathcal{F}_{\mathrm{ZKWE}}^{R,Q}$ is parameterized with the ZK relation R and the elimination relation Q.

- Upon receiving $(\textsc{Prove}, \mathit{sid}, \langle x, w\rangle)$ from party P where $\mathit{sid} = (P, T, \mathit{sid}')$, record $\langle P, x, w\rangle$, send $(\textsc{LeakProve}, \mathit{sid}, \langle x, \phi\rangle, P)$ to the adversary, where $\phi = 1$ if $(x, w) \in R$, and $\phi = 0$ otherwise. Ignore future $(\textsc{Prove}, \ldots)$ inputs.
- Upon receiving $(\textsc{Verify}, \mathit{sid}, w')$ from party T where $\mathit{sid} = (P, T, \mathit{sid}')$, record $\langle T, w'\rangle$, send $(\textsc{LeakVerify}, \mathit{sid}, T)$ to the adversary. Ignore future $(\textsc{Verify}, \ldots)$ inputs.
- Upon receiving $(\textsc{InflVerify}, \mathit{sid})$ from the adversary, if $(x, w) \in R$ and $(w, w') \notin Q$, then send $(\textsc{VerifyReturn}, \mathit{sid}, 1)$ to party T. Else if $(x, w) \notin R$ or $(w, w') \in Q$, then send $(\textsc{VerifyReturn}, \mathit{sid}, 0)$ to party T.

Fig. 1. Ideal functionality of zero-knowledge for the relation R with witness elimination based on the relation Q

prover P an input (x, w), and will receive from verifier T a secret input w'. Once both inputs are given, T is supposed to eventually receive a bit φ, where $\varphi = 1$ if and only if both $(x, w) \in R$ and $(w, w') \notin Q$ hold. The functionality blocks the secret inputs w and w' as well as the outcome of the test $(w, w') \in Q$ from the ideal world adversary $\mathcal{S}$; it leaks though a bit ϕ to $\mathcal{S}$ where $\phi = 1$ if and only if $(x, w) \in R$ (cf. the standard ZK functionality, [7,11]).

Some comments are in place:

- While one may use arbitrary relations Q for witness elimination, here we will be focusing on relations Q that capture some sort of equality test, either of the whole witness or a substring of the witness. This is consistent with our motivation in applying witness elimination to solve the suspect tracking problem in identification schemes. Nevertheless, more general elimination relations may be defined and could be potentially useful in other settings.
- Under the assumption that there is a way for any w to sample w' such that the event $(w, w') \in Q$ happens with negligible probability, one can simulate the $\mathcal{F}_{\mathrm{ZK}}$ ideal functionality using an ideal $\mathcal{F}_{\mathrm{ZKWE}}$ functionality box. To see this observe that the verifier can play the role of T and select T's input w' at random. The assumption on Q needed for the simulation can be seen to be easily satisfied by the substring equality relations Q we consider here.
- Our formalization of $\mathcal{F}_{\mathrm{ZKWE}}$ ensures that as long as no party is corrupted the protocol transcript by itself should not leak any information about the relation of w, w'. This as an essential property as witness elimination should not be applied to protocols transcripts that are not "live." Moreover, even if the party T is corrupted the protocol transcript should enable a corrupted T to test the predicate $Q(w, \cdot)$ for **no more** than a single candidate witness. This property is similar to the property of resistance against offline dictionary attacks in password-based key exchange schemes where an

eavesdropper should not be able to use a protocol transcript to scan for the correct password.

- In our formalization of $\mathcal{F}_{\mathrm{ZKWE}}$ the adversary $\mathcal{S}$ learns whether the prover uses a valid witness or not. Intuitively this suggests that we allow the protocols that realize $\mathcal{F}_{\mathrm{ZKWE}}$ to have a method for a (dishonest) party T to extract this fact from a protocol interaction with an honest prover. This is consistent with the fact that we will only consider elimination relations Q for which the $\mathcal{F}^{R,Q}_{\mathrm{ZKWE}}$ functionality simulates the $\mathcal{F}^{R}_{\mathrm{ZK}}$ ideal functionality and thus this does not affect the intended security properties of the protocol (the prover P is aware that the verifier T can easily learn whether she possesses a valid witness or not, cf. the second bullet above).

3.2 Generic Construction Based on Smooth Projective Hashing

In this section we present a construction for $\mathcal{F}^{R,Q}_{\mathrm{ZKWE}}$ that applies to the setting when the elimination relation Q is either the equality relation or it contains all elements of the form $\langle (w_1, w_2), (w'_1, w'_2)\rangle$ such that $w_2 = w'_2$ (i.e., Q is a "substring equality" relation). Based on this, in the remaining of the section we assume that the NP relation R contains pairs of the form $(x, (w_1, w_2))$ and the elimination relation Q tests the "w_2-part" of the witness (while it should be noted that the "w_1-part" could be empty).

Our construction is based on a CPA-secure encryption scheme$\mathcal{E}$=(gen,enc,dec), a zero-knowledge proof of membership (ZKPM) scheme $\mathcal{P} = (\mathsf{setup}, \mathsf{P}, \mathsf{V}, \mathsf{S})$, and a hard smooth projective hashing $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ in the context of a CPA-secure encryption scheme $\mathcal{E}' = (\mathsf{gen}', \mathsf{enc}', \mathsf{dec}')$. Recall that both P and T are supposed to keep their inputs private. P's input is the witness to the ZK relation that she is supposed to demonstrate knowledge of while T's input determines the elimination relation. In accordance to the above our construction includes: *(1)* an encryption of the prover's input under $\mathcal{E}$ and the verifier's input under $\mathcal{E}'$, *(2)* an inequality test subprotocol using the smooth projective hash $\mathcal{H}$, and *(3)* a ZKPM for consistency proof that the above *(1)* and *(2)* components are consistent and at the same time the input of the prover satisfies the ZK-relation R. As mentioned, to implement the (one-sided) inequality test, we use the hard smooth projective hashing in the context of the public-key encryption scheme $\mathcal{E}'$. If the party P's input witness is indeed matched by the verifier T, i.e., $w_2 = w'_2$, then the projection property of the hashing will guarantee that $\kappa = \kappa'$ holds (where κ, κ' are the respective hash values with κ being transmitted by the prover to the verifier), and otherwise if $w_2 \neq w'_2$ then $\kappa \neq \kappa'$. A corrupted P cannot learn T's input given the underlying public-key encryption $\mathcal{E}'$ is CPA-secure. A corrupted T on the other hand cannot learn P's witness in the case that $w_2 \neq w'_2$ given the smoothness property of the hashing as well as the CPA property of $\mathcal{E}$. Of course T will learn (part of) P's witness in the case $w_2 = w'_2$; nevertheless, any polynomial-time bounded eavesdropper will not be able to extract any information about the exchanged witnesses (cf. lemma 1). In figure 2 we present the protocol in details.

Regarding the security of the construction we have the following:

Theorem 1. *Given that $\mathcal{E}$ is a CPA-secure encryption scheme, $\mathcal{H}$ is a family of hard smooth projective hash functions in the context of a CPA-secure encryption scheme $\mathcal{E}'$, and $\mathcal{P}$ is a zero-knowledge proof of membership, then the construction in figure 2 realizes functionality $\mathcal{F}_{\text{ZKWE}}^{R,Q}$ against non-adaptive adversaries in the $\mathcal{F}_{\text{CRS}}$-hybrid world.*

In order to prove the theorem, we describe first an ideal world simulator $\mathcal{S}$ as below; then we show for this simulator that for any PPT environment $\mathcal{Z}$, $\text{EXEC}_{\pi,\mathcal{A}_d,\mathcal{Z}}^{\mathcal{F}_{\text{CRS}}} \approx \text{EXEC}_{\pi_d,\mathcal{S},\mathcal{Z}}^{\mathcal{F}_{\text{ZKWE}}^{R,Q}}$ (the proof can be found in the full version).

Protocol π for zero-knowledge with witness elimination for relations R and Q

Common reference string: $crs = (pk', pk, \rho)$, where pk' and pk are public keys of encryption schemes $\mathcal{E}'$ and $\mathcal{E}$ respectively, and ρ is a reference string of ZKPM scheme $\mathcal{P}$.

Protocol steps:

Upon receiving $(\textsc{Verify}, sid, w_2')$ from the environment, where $sid = (P, T, sid')$, party T selects $r' \xleftarrow{r} U'$ and computes $c' \leftarrow \mathsf{enc}'(pk'; w_2'; r')$, and sends $(\texttt{move1}, c')$ to party P.

Upon receiving $(\textsc{Prove}, sid, \langle x, w_1, w_2 \rangle)$ from the environment, party P first checks if $(x, (w_1, w_2)) \in^? R$ and waits for a move1 message from party T; after receiving move1 message,

- if $(x, (w_1, w_2)) \notin R$ then party P sends party T a message $(\texttt{move2}, \text{“no valid proof”})$;
- else if $(x, (w_1, w_2)) \in R$ then party P sends party T a message $(\texttt{move2}, s, \kappa)$, where $\kappa \leftarrow H_k(c', w_2)$, $s \leftarrow \alpha(k, c')$, $k \xleftarrow{r} K$; then party P computes $c \leftarrow \mathsf{enc}(pk; w_1, w_2; r)$ where $r \xleftarrow{r} U$; now parties P and T play the roles of prover and verifier respectively to run a ZKPM subprotocol

$$\mathsf{P}(w_1, w_2, k) \xleftrightarrow{(x,c',c,\kappa)\in L_{R'}} \mathsf{V}()$$

i.e, party P proves to party T that based on crs, the statement $(x, c', c, \kappa) \in L_{R'}$ where $L_{R'} = \{(x, c', c, \kappa) | \exists (w_1, w_2, k, r) \text{ s.t. } (x, (w_1, w_2)) \in R \wedge \kappa = H_k(c', w_2) \wedge\ c = \mathsf{enc}(pk; w_1, w_2; r)\}$.

Upon receiving $(\texttt{move2}, \text{“no valid proof”})$ from party P, party T returns $(\textsc{VerifyReturn}, sid, 0)$ to the environment. Else if receiving $(\texttt{move2}, s, \kappa)$, party T computes $\kappa' \leftarrow J_s(c', w_2'; r')$; and if $\kappa \neq \kappa'$ and also party T accepts the proof in the subprotocol above, then party T returns $(\textsc{VerifyReturn}, sid, 1)$ to the environment; otherwise returns $(\textsc{VerifyReturn}, sid, 0)$ to the environment.

Fig. 2. Generic construction of zero-knowledge with witness elimination based on a CPA-secure encryption scheme $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$, and a zero-knowledge proof of membership scheme $\mathcal{P} = (\mathsf{setup}, \mathsf{P}, \mathsf{V}, \mathsf{S})$, as well as a hard smooth projective hashing $\mathcal{H} = (\mathsf{paragen}, \mathsf{sampler}, G, H, K, \alpha, S, J)$ in the context of a CPA-secure encryption scheme $\mathcal{E}' = (\mathsf{gen}', \mathsf{enc}', \mathsf{dec}')$

The construction of simulator. The simulator $\mathcal{S}$ first runs the key generation algorithms of both the encryption schemes $\mathcal{E}'$, $\mathcal{E}$ and the proof system $\mathcal{P}$ to obtain key pairs, i.e., $(pk', sk') \leftarrow \mathsf{gen}'(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{gen}(1^\lambda)$, and $(\rho, \tau) \leftarrow \mathsf{setup}(1^\lambda)$; then $\mathcal{S}$ sets $crs \leftarrow (pk', pk, \rho)$ and the corresponding trapdoor $td \leftarrow (sk', sk, \tau)$. Then $\mathcal{S}$ initializes $\mathcal{A}_d$ by giving it crs as the common reference string. Now the simulator $\mathcal{S}$ interacts with the environment $\mathcal{Z}$, the functionality $\mathcal{F}_{\text{ZKWE}}^{R,Q}$ and its subroutine $\mathcal{A}_d$.

The simulator $\mathcal{S}$ simulates the protocol transcripts by following the protocol π on behalf of the honest parties. Since the secret inputs of the honest parties are not known, the simulator uses randomly selected $\tilde{w}_2'$ and $\tilde{w}$ as the witnesses to compute c' and κ, c, and further, the simulator runs S with V to simulate the ZKPM subprotocol transcripts. We give details below. Note that we only consider non-adaptive corruptions.

After receiving $(\textsc{LeakProve}, sid, \langle x, 1\rangle, P)$ from the functionality (party P is honest), and the `move1` message from party T including c', the simulator $\mathcal{S}$ uses a randomly selected $\tilde{w} = (\tilde{w}_1, \tilde{w}_2)$ (since the real witness $w = (w_1, w_2)$ is blocked by the functionality) to compute κ and c, and runs S with V (instead of P with V) to simulate the ZKPM subprotocol proof transcripts. If on the other hand it receives $(\textsc{LeakProve}, sid, \langle x, 0\rangle, P)$ from the functionality and the `move1` message from T, $\mathcal{S}$ responds T with a `move2` message "no valid proof".

After receiving $(\textsc{LeakVerify}, sid, T)$ from the functionality (party T is honest), the simulator $\mathcal{S}$ uses a randomly selected $\tilde{w}_2'$ (since the real witness w_2' is blocked by the functionality) to compute c', and sends c' as the `move1` message to party P. Later when it receives the `move2` message from party P, $\mathcal{S}$ sends $(\textsc{InflVerify}, sid)$ to the functionality.

We next consider the case that party T is corrupted. When $\mathcal{S}$ receives $(\textsc{LeakProve}, sid, \langle x, 1\rangle, P)$ from the functionality and a `move1` message including c' from the corrupted party T, the simulator uses sk' to decrypt c' into w_2', and then sends the input $(\textsc{Verify}, sid, w_2')$ in the name of T to the functionality; after receiving $(\textsc{LeakVerify}, sid, T)$ from the functionality, $\mathcal{S}$ sends $(\textsc{InflVerify}, sid)$ to the functionality; if the functionality returns $(\textsc{VerifyReturn}, sid, 1)$ which means $w_2' \neq w_2$ where w_2 is from the actual witness of the prover, the simulator randomly selects $\breve{w} = (\breve{w}_1, \breve{w}_2)$ to finish the simulation of the P protocol; if the functionality returns $(\textsc{VerifyReturn}, sid, 0)$ which means $w_2' = w_2$ where w_2 is from the real witness, the simulator randomly selects $\breve{w}_1$, and uses $\breve{w} = (\breve{w}_1, w_2)$ to finish the simulation. On the other hand, if it receives $(\textsc{LeakProve}, sid, \langle x, 0\rangle, P)$ from the functionality and a `move1` message, $\mathcal{S}$ simulates the P by responding with "no valid proof" as the `move2` message.

We now consider the case that party P is corrupted. If $\mathcal{S}$ receives the message $(\mathtt{move2}, s, \kappa)$ from the corrupted party P, and after executing the ZKPM subprotocol the party V returns 1 which means party T accepts the subprotocol proof that $(x, c', c, \kappa) \in L_{R'}$, the simulator uses sk to decrypt c into w and sends input $(\textsc{Prove}, sid, \langle x, w\rangle)$ in the name of P to the functionality; else if V returns 0 which means party T rejects the subprotocol proof, then the simulator sends input $(\textsc{Prove}, sid, \langle x, \perp\rangle)$ where $(x, \perp) \notin R$. Further, if $\mathcal{S}$ receives (move2, “no valid proof”) from the corrupted P, the simulator also sends input (Prove, $sid, \langle x, \perp\rangle$) to functionality. After it receives (LeakProve, $sid, \langle x, \phi\rangle$) from the functionality, $\mathcal{S}$ sends (InflVerify, sid) to the functionality, and the functionality returns (VerifyReturn, sid, φ) to the dummy T according to its program. This completes the description of the simulator $\mathcal{S}$.

3.3 Illustrative Efficient Construction

In this subsection, we instantiate the generic construction to obtain an explicit practical protocol for a zero-knowledge proof with witness elimination. The zero-knowledge proof we develop is for proving possession of a non-adaptive Boneh-Boyen digital signature [3]. We first introduce a building block, a hard smooth projective hashing in the context of linear ElGamal encryption [4], and then give the protocol.

Our construction yields immediately an anonymous identification scheme with suspect tracking. This can be seen as follows: each user in the system will hold a BB signature on a random message while the public-key of the BB signature will be publicly available. To authenticate itself to a verifier, a prover will engage in a proof of knowledge of a signature. Note that based on the non-adaptive unforgeability of BB signatures it is impossible for any coalition of users to obtain an additional signature. Observe now that if the system manager wishes to track a suspect it may disclose the message-signature pair that was assigned to the particular user. This will enable any verifier to employ witness elimination and thus check that any prover that engages in the interaction does not possess the suspect message-signature pair.

Bilinear Groups. Let $\mathbb{G} = \langle g\rangle$ be a cyclic group of prime order p such that $\check{\mathsf{e}} : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_t$ is a bilinear map, i.e., for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, it holds that $\check{\mathsf{e}}(u^a, v^b) = \check{\mathsf{e}}(u, v)^{ab}$ and $\check{\mathsf{e}}$ is non-trivial, i.e., $\check{\mathsf{e}}(g, g) \neq 1$. Note that $|\mathbb{G}_t| = p$.

Linear Encryption. Boneh et al. [4] proposed a variant of ElGamal encryption, called, Linear Encryption that is suitable for groups over which the DDH assumption fails.

Key Generation: $(pk, sk) \leftarrow \mathsf{gen}(1^\lambda)$, where $pk = (u, v, w)$ and $sk = (y, z)$, and $u, v, w \in \mathbb{G}$ and $y, z \in \mathbb{Z}_p$ such that $u^y = v^z = w$.

Encryption: $c \leftarrow \mathsf{enc}(pk; m; r)$, where $m \in \mathbb{G}$, $r = (\eta, \theta)$ with $\eta, \theta \overset{\mathrm{r}}{\leftarrow} \mathbb{Z}_p$, and $c = (u^\eta, v^\theta, w^{\eta+\theta}m)$.

Decryption: $m \leftarrow \mathsf{dec}(pk, sk; c)$, where $c = (U, V, W)$, $sk = (y, z)$, and $m = \frac{W}{U^y\cdot V^z}$.

The Linear encryption is CPA-secure under the Decision Linear Diffie-Hellman assumption [4].

Definition 2 (Decision Linear Diffie-Hellman Assumption). *We say that the Decision Linear Diffie-Hellman assumption holds in $\mathbb{G}$ if for all PPT distinguisher $\mathcal{A}$ we have*

$$\left|\begin{array}{l}\Pr[u, v, w \overset{\mathrm{r}}{\leftarrow} \mathbb{G}; \beta, \gamma \overset{\mathrm{r}}{\leftarrow} \mathbb{Z}_p : \mathcal{A}(u, v, w, u^\beta, v^\gamma, w^{\beta+\gamma}) = 1] \\ \quad - \Pr[u, v, w, \chi \overset{\mathrm{r}}{\leftarrow} \mathbb{G}; \beta, \gamma \overset{\mathrm{r}}{\leftarrow} \mathbb{Z}_p : \mathcal{A}(u, v, w, u^\beta, v^\gamma, \chi) = 1]\end{array}\right| \leq \mathsf{negl}(\lambda)$$

$$crs = (q, \mathbf{g}, \mathbf{h}, \mathbf{G}, \mathsf{hash}; \mathsf{n}, \mathsf{g}; p, g, u, v, w, X, \mathbb{G}, \mathbb{G}_t, \breve{\mathsf{e}})$$

$\boxed{P}$
$\texttt{input} = (m, \sigma)$ s.t.
$\breve{\mathsf{e}}(g^m X, \sigma) = \breve{\mathsf{e}}(g, g)$

$\boxed{T}$
$\texttt{input} = \sigma'$

$\eta', \theta' \xleftarrow{\mathrm{r}} \mathbb{Z}_p;$
$U' \leftarrow u^{\eta'};\ V' \leftarrow v^{\theta'};\ W' \leftarrow w^{\eta'+\theta'}\sigma';$

$\xleftarrow{U', V', W'}$

$\beta, \gamma, \delta, \zeta_\beta, \zeta_\gamma, \zeta_\delta \xleftarrow{\mathrm{r}} \mathbb{Z}_p;$
$s_1 \leftarrow u^\beta w^\delta;\ s_2 \leftarrow v^\gamma w^\delta;$
$\kappa \leftarrow (U')^\beta (V')^\gamma (\frac{W'}{\sigma})^\delta;$ $\xrightarrow{s_1, s_2, \kappa}$

$\kappa' \leftarrow (s_1)^{\eta'}(s_2)^{\theta'};$
$\kappa =^? \kappa';$

$\zeta_m \xleftarrow{\mathrm{r}} \pm[0, 2^{\ell_0+\ell_d+\ell_p}];\ \chi, \zeta_\chi \xleftarrow{\mathrm{r}} \mathbb{Z}_{\mathsf{n}}^*;$
$E \leftarrow \mathsf{g}^m \chi^{\mathsf{n}} \bmod \mathsf{n}^2;$
$B_1 \leftarrow \mathsf{g}^{\zeta_m}(\zeta_\chi)^{\mathsf{n}} \bmod \mathsf{n}^2;$
$\eta, \theta, \zeta_\eta, \zeta_\theta, \zeta_{m\eta}, \zeta_{m\theta}, \zeta_{\delta\eta}, \zeta_{\delta\theta} \xleftarrow{\mathrm{r}} \mathbb{Z}_p;$
$U \leftarrow u^\eta;\ V \leftarrow v^\theta;\ W \leftarrow w^{\eta+\theta}\sigma;$
$B_2 \leftarrow u^{\zeta_\eta};\ B_3 \leftarrow v^{\zeta_\theta};$
$B_4 \leftarrow (U')^{\zeta_\beta}(V')^{\zeta_\gamma}(W'W^{-1})^{\zeta_\delta} w^{\zeta_{\delta\eta}+\zeta_{\delta\theta}};$
$B_5 \leftarrow \breve{\mathsf{e}}(g, W)^{\zeta_m} \breve{\mathsf{e}}(g, w)^{-\zeta_{m\eta}-\zeta_{m\theta}} \breve{\mathsf{e}}(X, w)^{-\zeta_\eta-\zeta_\theta};$
$\omega \leftarrow \mathsf{hash}(B_1, B_2, B_3, B_4, B_5);$
$\mu \xleftarrow{\mathrm{r}} \mathbb{Z}_q;\ C \leftarrow \mathbf{g}^\omega \mathbf{h}^\mu;$ $\xrightarrow{E, U, V, W, C}$

$d \xleftarrow{\mathrm{r}} \{0, 1\}^{\ell_d};$

$\xleftarrow{d}$

$\xi_\chi \leftarrow \zeta_\chi \cdot \chi^d \bmod \mathsf{n};\quad \xi_\eta \leftarrow \zeta_\eta + d \cdot \eta;$
$\xi_\theta \leftarrow \zeta_\theta + d \cdot \theta;\quad \xi_\delta \leftarrow \zeta_\delta + d \cdot \delta;$
$\xi_\beta \leftarrow \zeta_\beta + d \cdot \beta;\quad \xi_\gamma \leftarrow \zeta_\gamma + d \cdot \gamma;$
$\xi_{\delta\eta} \leftarrow \zeta_{\delta\eta} + d \cdot \delta\eta;$
$\xi_{\delta\theta} \leftarrow \zeta_{\delta\theta} + d \cdot \delta\theta;$
$\xi_m \leftarrow \zeta_m + d \cdot m$ (in $\mathbb{Z}$);
$\xi_{m\eta} \leftarrow \zeta_{m\eta} + d \cdot m\eta;$
$\xi_{m\theta} \leftarrow \zeta_{m\theta} + d \cdot m\theta;$
$\boldsymbol{\xi} \leftarrow (\xi_\chi, \xi_\eta, \xi_\theta, \xi_\delta, \xi_\beta, \xi_\gamma, \xi_{\delta\eta}, \xi_{\delta\theta}, \xi_m, \xi_{m\eta}, \xi_{m\theta});$
$\boldsymbol{B} \leftarrow (B_1, B_2, B_3, B_4, B_5);$ $\xrightarrow{\boldsymbol{\xi}, \boldsymbol{B}, \mu}$

$\omega \leftarrow \mathsf{hash}(B_1, B_2, B_3, B_4, B_5); \mathbf{g}^\omega \mathbf{h}^\mu =^? C;$
$\xi_m \in^? \pm[0, 2^{\ell_0+\ell_d+\ell_p+1}]; E \in^? \mathbb{Z}_{\mathsf{n}^2}^*;$
$\mathsf{g}^{\xi_m}(\xi_\chi)^{\mathsf{n}} =^? B_1 E^d \bmod \mathsf{n}^2;$
$u^{\xi_\eta} =^? B_2 U^d;\ v^{\xi_\theta} =^? B_3 V^d;$
$(U')^{\xi_\beta}(V')^{\xi_\gamma}(W'W^{-1})^{\xi_\delta} w^{\xi_{\delta\eta}+\xi_{\delta\theta}} =^? B_4 \kappa^d;$
$\breve{\mathsf{e}}(g, W)^{\xi_m} \breve{\mathsf{e}}(g, w)^{-\xi_{m\eta}-\xi_{m\theta}} \breve{\mathsf{e}}(X, w)^{-\xi_\eta-\xi_\theta}$
$=^? B_5 \breve{\mathsf{e}}(g, g)^d \breve{\mathsf{e}}(X, W)^{-d};$

Fig. 3. Concrete construction of zero-knowledge with witness elimination

Hard smooth projective hashing in the context of linear ElGamal encryption. We now define a smooth projective hash function LEH for this encryption scheme. Here $C_{pk} \overset{\mathsf{def}}{=} (\mathbb{G})^3$ and $M \overset{\mathsf{def}}{=} \mathbb{G}$, so $X = (\mathbb{G})^4$; and $W = (\mathbb{Z}_p)^2$

and $G = \mathbb{G}$; the key space is defined by $K = (\mathbb{Z}_p)^3$, i.e., a key is a triple $k = (\beta, \gamma, \delta)$, where $\beta, \gamma, \delta \xleftarrow{r} \mathbb{Z}_p$; the key projection function α on input the key $k = (\beta, \gamma, \delta)$ and ciphertext $c = (U, V, W)$ is defined by $s = (s_1, s_2) = (u^\beta w^\delta, v^\gamma w^\delta)$; function $H_k : (\mathbb{G})^4 \rightarrow \mathbb{G}$ is defined as $H_k((U, V, W), m) = U^\beta V^\gamma (W/m)^\delta$; function $J_s : (\mathbb{G})^4 \times (\mathbb{Z}_p)^2 \rightarrow \mathbb{G}$ is defined as $J_s((U, V, W), m, (\eta, \theta)) = (s_1)^\eta (s_2)^\theta$.

Lemma 2. *LEH is a hard smooth projective hash function.*

The zero-knowledge with witness elimination protocol. In our protocol, party P keeps a witness including two parts m and σ such that the relation $\breve{e}(g^m X, \sigma) = \breve{e}(g, g)$ which can be viewed as a non-adaptive BB message-signature pair with public parameter $(p, g, X, \mathbb{G}, \mathbb{G}_t, \breve{e})$ and signing secret x, where $X = g^x$. The relation Q for which the witness elimination is performed is an equality test on the signature part of the message-signature pair. Note that in the non-adaptive BB-signature there is a correspondence between messages and signatures and thus Q amounts to simply an equality relation.

The inequality test subprotocol is based on a hard smooth projective hashing in the context of linear ElGamal encryption introduced in the previous subsection. Then we use two encryption schemes to encrypt P's witness, i.e., a Paillier [29] encryption to encrypt m into E, and a linear ElGamal encryption to encrypt σ into $\langle U, V, W \rangle$. We construct a three-move Sigma-protocol to prove: (1) m and σ is a valid non-adaptive BB message-signature pair; (2) the inequality subprotocol is consistent with the statement in (1); and (3) E and $\langle U, V, W \rangle$ are cipertexts for m based on Paillier encryption and for σ based on linear ElGamal encryption respectively. Further to defend against a dishonest verifier, we use a technique of [14] to wrap up the above Sigma-protocol by using a commitment scheme. The protocol is presented in full details in figure 3.

Acknowledgement

We thank Juan Garay for helpful discussions, and the current title was suggested by him. We also thank the anonymous reviewers for valuable comments.

References

1. Ateniese, G., Song, D.X., Tsudik, G.: Quasi-efficient revocation in group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
2. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC, pp. 103–112. ACM Press, New York (1988)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

5. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 168–177. ACM Press, New York (2004)
6. Camenisch, J.: Group signature schemes and payment systems based on the discrete logarithm problem. ETH Series in Information Security an Cryptography, 2, PhD thesis (1998), `http://www.zurich.ibm.com/~jca/papers/diss.ps`
7. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society, Los Alamitos (2001)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Cryptology ePrint Archive, Report 2000/067 (December 2005), `http://eprint.iacr.org/2000/067/`
9. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC, pp. 235–244 (2000), `http://eprint.iacr.org/1999/022`
10. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005), `http://eprint.iacr.org/2005/196`
11. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503. ACM Press, New York (2002), `http://eprint.iacr.org/2002/140`
12. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
13. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002), `http://www.shoup.net/papers/uhp.pdf`
14. Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000)
15. Damgård, I., Nielsen, J.B., Wichs, D.: Isolated proofs of knowledge and isolated zero knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 509–526. Springer, Heidelberg (2008)
16. De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On monotone formula closure of szk. In: FOCS, pp. 454–465. IEEE Computer Society Press, Los Alamitos (1994)
17. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. 30(2), 391–437 (2000); preliminary version appears at STOC 1991
18. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM 51(6), 851–898 (2004); preliminary version appears at STOC 1998
19. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC, pp. 416–426. ACM Press, New York (1990)
20. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
21. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. ACM Trans. Inf. Syst. Secur. 9(2), 181–234 (2006); preliminary version appears at Eurocrypt 2003

22. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. ICALP 1990 25(1), 169–192 (1990); preliminary version appears at ICALP 1990
23. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)
24. Halevi, S., Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. In: Cryptology ePrint Archive: Report 2007/118, 2007 (2005); preliminary version appears at Eurocrypt 2005
25. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
26. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
27. Nakanishi, T., Funabiki, N.: Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 533–548. Springer, Heidelberg (2005)
28. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. 35(5), 1254–1281 (2006); preliminary version appears at STOC 1999
29. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
30. Schnorr, C.-P.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)
31. Song, D.X.: Practical forward secure group signature schemes. In: ACM Conference on Computer and Communications Security, pp. 225–234 (2001)

Distributed Public-Key Cryptography from Weak Secrets

Michel Abdalla[1], Xavier Boyen[2], Céline Chevalier[1], and David Pointcheval[1]

[1] Ecole Normale Supérieure, CNRS-INRIA, Paris, France
[2] Stanford University, Stanford, California

Abstract. We introduce the notion of distributed password-based public-key cryptography, where a virtual high-entropy private key is implicitly defined as a concatenation of low-entropy passwords held in separate locations. The users can jointly perform private-key operations by exchanging messages over an arbitrary channel, based on their respective passwords, without ever sharing their passwords or reconstituting the key.

Focusing on the case of ElGamal encryption as an example, we start by formally defining ideal functionalities for distributed public-key generation and virtual private-key computation in the UC model. We then construct efficient protocols that securely realize them in either the RO model (for efficiency) or the CRS model (for elegance).

We conclude by showing that our distributed protocols generalize to a broad class of "discrete-log"-based public-key cryptosystems, which notably includes identity-based encryption. This opens the door to a powerful extension of IBE with a virtual PKG made of a group of people, each one memorizing a small portion of the master key.

1 Introduction

Traditional wisdom says that it is impossible to do public-key cryptography from short passwords. This is because any low-entropy private key will quickly succumb to an off-line dictionary attack, made possible by the very publication of the public key, which can thus be used as a non-interactive test function. Since off-line attacks are very effective against weak secrets, it is imperative that the private keys in public-key systems be highly random and complex, but that makes them hopelessly impossible to be remembered by humans.

But, what if, instead of being held as an indivisible entity, the private key were chopped into many little pieces, each one of them independently memorized by a different person in a group of friends or colleagues? The components of the key would be safe in the respective memories of the individual group members, at least as long as it is not used. The only complication is the need to reassemble the full private key from the various components, so that private-key operations can be performed. Naturally, the secret holders should not actually reassemble the key, but instead perform a distributed computation of whichever private-key operation they need, without ever having to meet or even reconstitute the key.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 139–159, 2009.

Unusual Requirements. Even if one can perform private-key computations without reassembling the key, there are other, more subtle vulnerabilities.

For starters, we cannot simply assume that the (virtual) private key is simply made of some number of random components (one per user) generated independently and uniformly at random. On the contrary, we must assume that the various components are arbitrary and possibly correlated, and some of them potentially very weak and easily guessable. This is because of our requirement of human-memorability: for the components to be truly memorable, it is imperative that their respective owners choose them in whichever way they please.

A consequence of the above is that it also opens the possibility of *password reuse* by the various users: although this is a bad security practice that should be discouraged, it is also one that is very common and that we should acknowledge and handle the best way we can, rather than pretend that it will not happen.

Additionally, since the various secret holders do not necessarily trust each other, it is necessary that they be able to choose their individual secrets in complete privacy. In fact, any solution to our question must deal with user corruptions and collusions, and remain as secure as the “sum total” of the key components of the remaining honest users.

Finally, we must have a notion of “group leader”, which is the person who will actually “own” the distributed virtual private key. By “own”, we mean that only the group leader will be able to use that key, i.e., obtain the results of any private computation based on it, with the help of the other group members. We stress that neither the leader nor anyone else should actually learn the key itself.

An important difference between our requirements and essentially all existing distributed protocols that deal with weak secrets (such as Group Password-based Key Agreement), is that here the secrets are chosen arbitrarily and privately by each user. We neither assume that all the secrets are the same (as in Group PAKE), or that they are all independent (as in Threshold Cryptography). The whole system should thus: (1) not fall apart if some of the passwords become exposed, as long as the combined entropy of the uncompromised passwords remains high; (2) preserve the privacy of all uncompromised passwords at all stages of the process (during the initial computation of the public key and any subsequent utilization of the virtual private key).

The notion of group leader is something necessary for our application. Most password-based protocols seek to achieve a *symmetric* outcome. Here, by contrast, the impetus to create a public/private key pair must originate in a particular user, who will become the leader, and who seeks the help of other, semi-trusted individuals to help him or her remember the key. (The leader can return the favor later or share the result of any private computation, outside of the core protocol.) Remark also that whereas it is easy for the leader to share the result of a private computation with the other members, it would be almost impossible to restrict such result to the leader if the computation gave the result to all.

General Approach. The aim of this paper is thus primarily to show how to do asymmetric cryptography from a distributed set of human-memorable secrets. Since public-key cryptography from *single* passwords is irremediably insecure, the best we can hope for is to base it on moderately-sized *distributed collections* of them: Given a regular system (such as signature, encryption, or IBE), we devise a pair of protocols that take independent user passwords as inputs, and, in a distributed manner: 1) generate a publishable public key that corresponds to the set of passwords; 2) do private computations on the virtual private key.

To create a key pair, a group of players led by a designated "group leader" engages in the distributed key generation protocol. The protocol runs over unauthenticated channels, and if all goes well, results in an explicit public key for anyone to see and use. The private key is not explicitly computed and remains implicitly defined by the set of passwords. To use the private key, the same group of players engages in another protocol, using the same passwords as in the key generation protocol. The protocol again runs over unauthenticated channels. If all goes well, the leader, and only the leader, obtains the results of the computation. Again, the private key is not explicitly computed, and the passwords remain private to their respective owners.

Unlike *regular public-key cryptosystems*, the private key is never stored or used all at once; it remains virtual and delocalized, and the private-key operation is done using an interactive protocol. But unlike *threshold cryptography*, where the shares are uniformly randomized and typically as long as the shared secret itself, here the passwords are arbitrary and user-selected. Unlike *password-based encryption*, off-line attacks are thwarted by virtue of the high joint entropy from many distinct user passwords, which must be guessed all at once. On-line attacks against single passwords cannot be prevented, but are very slow as they require an on-line commitment for each guess. Unlike *password-authenticated key exchange protocols*, here the user passwords are not the same or even related to each other: the passwords are truly personal.

Our Results. First, we formalize this class of protocols and their security requirements; for convenience we do so in the UC model [12], which lends itself nicely to the analysis of password-based protocols. Second, we propose a reasonably efficient construction for the ElGamal cryptosystem as a working example [18], which we prove secure both in the RO and CRS models. Third, we conclude by showing that our construction generalizes easily to a broad class of "discrete-log"-type public-key schemes, and, quite notably, the whole set of schemes derived from the BF and BB identity-based cryptosystems [9,7].

Even though for simplicity we focus on public-key systems with a special form (those that operate by raising elements of an algebraic group to the power of the private key and/or ephemeral exponents), this structure is general enough to capture many examples of exponentiation-based cryptosystems, and even IBE systems that require a pairing, as we just mentioned.

Remarkably, and of independent interest, this gives us an interesting twist on the notion of IBE, where the "central" key generation authority is replaced by a distributed set of users, each one of them holding a small piece of the master secret in the form of a self-selected easily memorable short password.

Related Work. Although there is no prior work on distributed cryptography from weak secrets *proper*, this notion is of course related to a fairly large body of literature that includes Password-Authenticated Key Exchange (PAKE) and Multi-Party Computation (MPC).

Multi-Party Computation. The first and most famous MPC protocol is due to Yao [30]. Depending on the setup, such protocols allow two participants with secret inputs to compute a public function of their joint inputs, without leaking anything other than the output of the function [24,23,6,15]. MPC protocols typically assume all communications between the players to be authentic: that is, an external mechanism precludes modifications or fake message insertions. The flip side is that such protocols tend to become insecure when the number of dishonest players reaches a certain threshold that allows them to take over the computation and from there recover the other players' inputs [28,2,25].

Several works have dealt with the case of MPC over unauthenticated channels [13,19,1], by prefacing the multi-party computation proper with some flavor of authentication based on non-malleable commitments or signatures [17]. The work of Barak *et al.* [1] in particular gives general conditions of what can and cannot be achieved in unauthenticated multi-party computations: they show that an adversary is always able to partition the set of players into disjoint "islands" that end up performing independent computations, but nothing else besides dropping messages and/or relaying them faithfully. They show how to transform any (realization of an) UC functionality into a multi-party version of the same that merely lets the adversary split the players into disjoint islands. They also show how to build password-based group key agreement (GPAKE) from this notion, first by creating a random session key for the group by running an MPC protocol without authentication, and then by verifying that all players have the same key using a "string equality" functionality. (By comparison, here, we force the users to commit to their passwords first, and then perform the actual computation based on those commitments.)

Although it is clear that, like so many other things in cryptography, our work can be viewed as a special case of unauthenticated MPC, our contribution lies not in this obvious conceptual step, but in the specification of suitable functionalities for the non-trivial problem of password-based threshold cryptography (and their efficient implementation). In particular, much grief arises from our requirement that each user has its *own* password (which may even be reused in other contexts), instead of a single common password for the whole group as in the applications considered in [1] and elsewhere.

On-line Passwords. The first insight that weak passwords could be used on-line (in a key exchange protocol) with relative impunity was made in [5]. It captured the idea that the success of an adversary in breaking the protocol should be proportional to the number of times this adversary interacts with the server, and only negligibly in its off-line computing capabilities.

In the password-only scenario (without public-key infrastructure), the first protocols with a proof of security appeared contemporaneously in [11] and [3], both in the random-oracle model. A (somewhat inefficient) protocol without any setup assumption was first proposed in [22]. A fairly efficient one in the common random string model was first given in [26] and generalized in [21].

To cope with concurrent sessions, the work of [14] was the first to propose an ideal functionality for PAKE in the UC model, as well as a protocol that securely realizes it. Unlike previous models, one of the major advantages of the UC one is that it makes no assumption on the distribution of the passwords; it also considers, for instance, some realistic scenarios such as participants running the protocol with different but possibly related passwords.

2 Security Model

The UC Framework. Throughout this paper, we assume basic familiarity with the universal composability (UC) framework [12].

Split Functionalities. Without any strong authentication mechanisms, the adversary $\mathcal{A}$ can always partition the players into disjoint subgroups and execute independent sessions of the protocol with each one, playing the role of the other players. Such an attack is unavoidable since players cannot distinguish the case in which they interact with each other from the case where they interact with $\mathcal{A}$. The authors of [1] addressed this issue by proposing a new model based on *split functionalities* which guarantees that this attack is the only one available to $\mathcal{A}$.

The split functionality is a generic construction based upon an ideal functionality: Its description can be found in the full version. In the initialization stage, the adversary $\mathcal{A}$ adaptively chooses disjoint subsets of the honest parties (with a unique session identifier that is fixed for the duration of the protocol). During the computation, each subset H activates a separate instance of the functionality $\mathcal{F}$. All these functionality instances are independent: The executions of the protocol for each subset H can only be related in the way $\mathcal{A}$ chooses the inputs of the players it controls. The parties $P_i \in H$ provide their own inputs and receive their own outputs, whereas $\mathcal{A}$ plays the role of all the parties $P_j \notin H$.

In the sequel, as we describe our two general functionalities $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$, one has to keep in mind that an attacker controlling the communication channels can always choose to view them as the split functionalities $s\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $s\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ implicitly consisting of multiple instances of $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ for non-overlapping subsets of the original players. Furthermore, one cannot prevent $\mathcal{A}$ from keeping some flows, which will never arrive. This is modelled in our functionalities (Figures 1 and 2) by a bit b, which specifies whether the flow is really sent or not.

The Ideal Functionalities. In the sequel we denote by n the number of users involved in a given execution of the protocol. One of the users plays a particular role and is denoted as the *group leader*, the others are simply denoted as players. Groups can be formed arbitrarily. Each group is *defined* by its leader (who "owns" the group by being the one to receive the result of any private computation) and an arbitrary number of other players in a specific order (who “assist” and “authorize” the leader in his or her use of the group’s virtual key).

We stress that the composition and ordering of a group is what defines it and cannot be changed: this ensures that any third-party who uses the group’s public key knows exactly how the corresponding private key will be accessed. If another player wants to be the leader, he or she will have to form a new group. (Even though such new group may contain the same set of members with possibly unchanged passwords, the two groups will be distinct and have different incompatible key pairs because of the different ordering).

As in [14], the functionality is not in charge of providing the passwords to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where a honest user executes the protocol with an incorrect password: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where related passwords are used in different protocols.

Since the functionalities are intended to capture distributed password protocols for (the key generation and private-key operation of) an arbitrary public-key primitive, we will represent all the primitive’s algorithms as black box parameters in our definitions. In general, we shall require: a function SecretKeyGen to combine a vector of passwords into a single secret key; a function PublicKeyGen to compute from a password vector a matching public key; a predicate PublicKeyVer to verify such public key against any password vector: this is important for the correctness of the ideal functionalities, but it also simplifies the use of the joint-state UC Theorem since it abstracts away the passwords that then do not need to be considered as part of the joint data; a function PrivateComp to perform the operation of interest using the private key: this could be the decryption function Dec of a public-key encryption scheme, the signing function Sign in a signature scheme, or the identity-based key extraction function Extract in an IBE system.

Both functionalities start with an initialization step, which basically waits for all the users to notify their interest in computing a public key or performing a private computation, as the case may be. Such notification is provided via newSession queries (containing the session identifier sid of the instance of the protocol, the user’s identity P_i, the identity of the group Pid, the user’s password pw_i, and when computing the private function, a public key pk and input in) sent by the players or by the simulator $\mathcal{S}$ in case of corruptions during the first flow (corresponding to the split functionality). Once all the users (sharing the same sid and Pid) have sent their notification message, the functionality informs the adversary that it is ready to proceed.

In principle, after the initialization stage is over, the eligible users are ready to receive the result. However the functionality waits for $\mathcal{S}$ to send a compute message before proceeding. This allows $\mathcal{S}$ to decide the exact moment when the key should be sent to the users and, in particular, it allows $\mathcal{S}$ to choose the exact moment when corruptions should occur (for instance $\mathcal{S}$ may decide to corrupt some party P_i before the key is sent but after P_i decided to participate

$\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ is parametrized by a security parameter k and an efficiently computable function $\mathsf{PublicKeyGen} : (\mathsf{pw}_1, \mathsf{pw}_2, \ldots, \mathsf{pw}_n) \mapsto \mathsf{pk}$ that derives a public key pk from a set of passwords. Denote by *role* either *player* or *leader*. The functionality interacts with an adversary $\mathcal{S}$ and a set of parties $P_1, \ldots, P_n$ via the following queries:

Initialization. Upon receiving a query $(\mathsf{newSession}, sid, Pid, P_i, \mathsf{pw}_i, role)$ from user P_i for the first time, where Pid is a set of at least two distinct identities containing P_i, ignore it if $role = leader$ and if there is already a record of the form $(sid, Pid, *, *, leader)$. Record $(sid, Pid, P_i, \mathsf{pw}_i, role)$ and send $(sid, Pid, P_i, role)$ to $\mathcal{S}$. Ignore any subsequent query $(\mathsf{newSession}, sid, Pid', *, *, *)$ where $Pid' \neq Pid$.
If there are already $|Pid|-1$ recorded tuples $(sid, Pid, P_j, \mathsf{pw}_j)$ for $P_j \in Pid \backslash \{P_i\}$, and exactly one of them such that $role = leader$, then while recording the $|Pid|$-th tuple, also record $(sid, Pid, \mathsf{ready})$ and send this to $\mathcal{S}$. Otherwise, record $(sid, Pid, \mathsf{error})$ and send $(sid, Pid, \mathsf{error})$ to $\mathcal{S}$.

Key Computation. Upon receiving a message $(\mathsf{compute}, sid, Pid)$ from the adversary $\mathcal{S}$ where there is a recorded tuple $(sid, Pid, \mathsf{ready})$, then compute $\mathsf{pk} = \mathsf{PublicKeyGen}(\mathsf{pw}_1, \ldots, \mathsf{pw}_n)$ and record (sid, Pid, pk).

Leader Key Delivery. Upon receiving a message $(\mathsf{leaderDeliver}, sid, Pid, \mathsf{b})$ from the adversary $\mathcal{S}$ for the first time, where there is a recorded tuple (sid, Pid, pk) and a record $(sid, Pid, P_i, pw_i, leader)$, send (sid, Pid, pk) to P_i and to $\mathcal{S}$ if $\mathsf{b} = 1$, or $(sid, Pid, \mathsf{error})$ otherwise. Record $(sid, Pid, \mathsf{sent})$ and send this to $\mathcal{S}$.

Player Key Delivery. Upon receiving $(\mathsf{playerDeliver}, sid, Pid, \mathsf{b}, P_i)$ from the adversary $\mathcal{S}$ where there are recorded tuples (sid, Pid, pk), $(sid, Pid, P_i, pw_i, player)$ and $(sid, Pid, \mathsf{sent})$, send (sid, Pid, pk) to P_i if $\mathsf{b} = 1$, or $(sid, Pid, \mathsf{error})$ otherwise.

User Corruption. If $\mathcal{S}$ corrupts $P_i \in Pid$ where there is a recorded tuple $(sid, Pid, P_i, \mathsf{pw}_i)$, then reveal pw_i to $\mathcal{S}$. If there also is a recorded tuple (sid, Pid, pk) and if (sid, Pid, pk) has not yet been sent to P_i, send (sid, Pid, pk) to $\mathcal{S}$.

Fig. 1. The Distributed Key Generation Functionality $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$

to a given session of the protocol; see [27]). Also, although in the key generation functionality all users are normally eligible to receive the public key, in the private computation functionality it is important that only the group leader receives the output (though he may choose to reveal it afterwards to others, outside of the protocol, depending on the application).

The Distributed Key Generation Functionality (Figure 1). The aim of this functionality is to provide a public key to the users, computed according to their passwords with respect to the previously mentioned function $\mathsf{PublicKeyGen}$ given in parameter, and it ensures that the group leader never receives an incorrect key in the end, whatever does the adversary. The protocol starts with an initialization phase as already described, followed by a key computation phase triggered by an explicit key computation query (so that $\mathcal{S}$ can control its timing.)

After the key is computed, the adversary can choose whether the group leader indeed receives this key. If delivery is denied, then nobody gets the key, and it is as if it was never computed. If delivery is allowed, then the group leader and $\mathcal{S}$ both receive the public key. This behavior captures the fact that the generated public key is intended to be available to all, starting with the opponent. (More to the point, this requirement will also weed out some bogus protocols that could only be secure if the public key remained unavailable to $\mathcal{S}$.) Once they have received the public key, the other players may be allowed to receive it too, according to a schedule chosen by $\mathcal{S}$, and modeled by means of key delivery queries from $\mathcal{S}$. Once $\mathcal{S}$ asks to deliver the key to a player, the key is sent immediately.

Note that given the public key, if the adversary knows sufficiently many passwords that the combined entropy of the remaining passwords is low enough, he will be able to recover these remaining passwords by brute force attack. This is unavoidable and explains the absence of any testPwd query in this functionality. (This has nothing to do with the fact that our system is distributed: off-line attacks are always possible in principle in public-key systems, and become feasible as soon as a sufficient portion of the private key becomes known.)

The Distributed Private Computation Functionality (Figure 2). The aim here is to perform a private computation for the sole benefit of the group leader. The leader is responsible for the correctness of the computation; in addition, it is the only user to receive the end result.

This functionality will thus compute a function of some supplied input *in*, depending on a set of passwords that must define a secret key corresponding to a given public key. More precisely, the functionality will be able to check the compatibility of the passwords with the public key thanks to the verification function PublicKeyVer, and if it is correct it will then compute the secret key sk with the help of the function SecretKeyGen, and from there evaluate PrivateComp(sk, *in*) and give the result to the leader. Note that SecretKeyGen and PublicKeyVer are naturally related to the function PublicKeyGen called by the former functionality. In all generality, unless SecretKeyGen and PublicKeyGen are both assumed to be deterministic, we need the predicate PublicKeyVer in order to verify that a public key is "correct" without necessarily being "equal" (to some canonical public key). Also note that the function SecretKeyGen is not assumed to be injective, lest it unduly restrict the number of users and the total size of their passwords.

Phases and Queries. During the initialization phase, each user is given as input a password pw_i as outlined earlier, but also an input *in*, and a public key pk. We stress that the security is guaranteed even if the users do not share the same values for *in* and pk, because then the functionality fails directly at the end of the initialization phase. At the end of this step, the adversary is also given knowledge of the common *in* and pk (as these are supposedly public).

After this initialization step is over, but before the actual computation, the adversary $\mathcal{S}$ is given the opportunity to make one or more *simultaneous* password guesses, by issuing a single Password Test query, to model a "man-in-the-middle" impersonation attack against a subset of users. The query must indicate the subset of user(s) targeted in the attack, and what password(s) $\mathcal{S}$ wishes to test for those user(s). If all passwords are compatible with pk, the affected users are marked as compromised, otherwise they are all marked as interrupted. Unaffected users remain marked as fresh. Observe that it is in the opponent's best interest

$\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ is parametrized by a security parameter k and three functions. $\mathsf{PublicKeyVer}$ is a boolean function $\mathsf{PublicKeyVer} : (\mathsf{pw}_1, \mathsf{pw}_2, \ldots, \mathsf{pw}_n, \mathsf{pk}) \mapsto b$, where $b = 1$ if the passwords and the public key are compatible, $b = 0$ otherwise. $\mathsf{SecretKeyGen}$ is a function $\mathsf{SecretKeyGen} : (\mathsf{pw}_1, \mathsf{pw}_2, \ldots, \mathsf{pw}_n) \mapsto \mathsf{sk}$, where sk is the secret key obtained from the passwords. Finally, $\mathsf{PrivateComp}$ is a private-key function $\mathsf{PrivateComp} : (\mathsf{sk}, c) \mapsto m$, where sk is the secret key, c is the function input (e.g., a ciphertext) and m the private result of the computation (e.g., the decrypted message). Denote by *role* either *player* or *leader*. The functionality interacts with an adversary $\mathcal{S}$ and a set of parties P_1,…,P_n via the following queries:

Initialization. Upon receiving a query $(\mathsf{newSession}, sid, Pid, P_i, \mathsf{pk}, c, \mathsf{pw}_i, role)$ from user P_i for the first time, where Pid is a set of at least two distinct identities containing P_i, ignore it if $role = leader$ and if there is already a record of the form $(sid, Pid, *, *, *, *, leader)$. Record $(sid, Pid, P_i, \mathsf{pk}, c, \mathsf{pw}_i, role)$, mark it fresh, and send $(sid, Pid, P_i, \mathsf{pk}, c, role)$ to $\mathcal{S}$. Ignore any subsequent query $(\mathsf{newSession}, sid, Pid', *, *, *, *, *)$ where $Pid' \neq Pid$.

If there are already $|Pid| - 1$ recorded tuples $(sid, Pid, P_i, \mathsf{pk}, c, \mathsf{pw}_i, role)$, and exactly one of them such that $role = leader$, then after recording the $|Pid|$-th tuple, verify that the values of c and pk are the same for all the users. If the tuples do not fulfill all of these conditions, report $(sid, Pid, \mathsf{error})$ to $\mathcal{S}$ and stop. Otherwise, record $(sid, Pid, \mathsf{pk}, c, \mathsf{ready})$ and send it to $\mathcal{S}$. The group leader is P_j.

Password Test. Upon receiving a first query $(\mathsf{testPwd}, sid, Pid, \{P_{i_1}, \ldots, P_{i_l}\}, \{\mathsf{pw}_{i_1}, \ldots, \mathsf{pw}_{i_l}\})$ from $\mathcal{S}$, if there exist l records $(sid, Pid, P_{i_k}, \mathsf{pk}, c, *, *)$, necessarily still marked fresh, and a record $(sid, Pid, \mathsf{pk}, c, \mathsf{ready})$, then denote by $\mathsf{pw}_{j_{l+1}}, \ldots, \mathsf{pw}_{j_n}$ the passwords of the other users of the group. If $\mathsf{PublicKeyVer}(\mathsf{pw}_1, \ldots, \mathsf{pw}_n, \mathsf{pk}) = 1$, edit the records of $P_{i_1}, \ldots, P_{i_l}$ to be marked $\mathsf{compromised}$ and reply to $\mathcal{S}$ with "correct guess". Otherwise, mark the records of the users $P_{i_1}, \ldots, P_{i_l}$ as $\mathsf{interrupted}$ and reply to $\mathcal{S}$ with "wrong guess". Ignore all subsequent queries of the form $(\mathsf{testPwd}, sid, Pid, *, *)$.

Private Computation. Upon receiving a message $(\mathsf{compute}, sid, Pid)$ from $\mathcal{S}$ where there is a recorded tuple $(sid, Pid, \mathsf{pk}, c, \mathsf{ready})$, then, if all records are fresh or $\mathsf{compromised}$ and $\mathsf{PublicKeyVer}(\mathsf{pw}_1, \ldots, \mathsf{pw}_n, \mathsf{pk}) = 1$, then compute $\mathsf{sk} = \mathsf{SecretKeyGen}(\mathsf{pw}_1, \ldots, \mathsf{pw}_n)$ and $m = \mathsf{PrivateComp}(\mathsf{sk}, c)$, and store (sid, Pid, m); Next, for all $P_i \in Pid$ mark the record $(sid, Pid, P_i, \mathsf{pk}, c, \mathsf{pw}_i, role)$ as $\mathsf{complete}$. In any other case, store $(sid, Pid, \mathsf{error})$. When the computation result is set, report the outcome (either error or $\mathsf{complete}$) to $\mathcal{S}$.

Leader Computation Delivery. Upon receiving $(\mathsf{leaderDeliver}, sid, Pid, \mathsf{b})$ from $\mathcal{S}$, where there is a recorded tuple (sid, Pid, m) such that $m \in \{\textit{well-formed messages}\} \cup \{\mathsf{error}\}$, and there exists a record $(sid, Pid, P_i, \mathsf{pk}, c, \mathsf{pw}_i, leader)$, send (sid, Pid, m) to P_i if b is equal to 1, or send $(sid, Pid, \mathsf{error})$ if b is equal to 0. If the group leader P_i is corrupted or $\mathsf{compromised}$, then send (sid, Pid, m) to $\mathcal{S}$ as well (note that $\mathcal{S}$ gets m automatically if P_j is corrupted).

User Corruption. If $\mathcal{S}$ corrupts $P_i \in Pid$ where there is a recorded tuple $(sid, Pid, P_i, \mathsf{pk}, c, \mathsf{pw}_i, role)$, then reveal pw_i to $\mathcal{S}$. If $role = leader$, if there also is a recorded tuple (sid, Pid, m), and if (sid, Pid, m) has not yet been sent to P_i, then also send (sid, Pid, m) to $\mathcal{S}$.

Fig. 2. The Distributed Private Computation Functionality $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$

to target only a single user in the Password Test query to optimize compromising probability.

Once the functionality receives a message of the form $(\mathsf{compute}, sid, Pid)$ from $\mathcal{S}$, it proceeds to the computation phase. This is done as follows. If (1) all records are fresh or $\mathsf{compromised}$, and (2) the passwords are compatible with the common public key pk, then the functionality computes the private key sk and then the output out. In all other cases, no message is computed.

In any case, after the key generation, the functionality informs the adversary of the result, meaning that $\mathcal{S}$ is told whether a message was actually computed or not. In particular, this means that the adversary also learns whether the users' passwords are compatible with pk or not. At first glance this may seem like a critical information to provide to the adversary. We argue, however, that this is not the case in our setting. Firstly, learning the status of the protocol (that is, whether it succeeded) without having any knowledge of the passwords that went into it is completely pointless, and the only knowledge that the adversary may have about those passwords are the ones it used in the $\mathsf{testPwd}$ impersonation query. Hence, as one should expect, from the status of the protocol the only useful thing that the adversary can learn is whether the password guesses it made were *all* good or not (as a single yes/no answer), but nothing else. Secondly, even if the adversary could somehow derive more utility from the protocol status, modeling that status as secret is not sensible because in most real-world scenarios it will be easy to infer from the users' behavior.

At the end, and similarly to the first functionality, the final result can either be released to the group leader, or withheld from it. However, this time, since the final result is a private output, there is no provision to distribute it to the other players. Also, $\mathcal{S}$ only gets the message if the leader either has been previously corrupted or if it is in the compromised state (either the leader has fallen under $\mathcal{S}$'s control, or $\mathcal{S}$ has successfully taken its place in the protocol).

Discussion. We emphasize that in this model only the leader and no other player receives the final result. Although this has the advantage of making the construction simpler, it is also the most useful and the only sensible choice. For starters, this makes our protocol much more resilient to password breaks in online impersonation attacks. To see why, suppose that the final output were indeed sent to all users. Then cracking the password of a single user would be all it took to break the system: adding more users would actually decrease the overall online security, because with a larger group comes a greater chance that some user will choose a weak password. By contrast, in the actual model, breaking the password of an ordinary user has no dire consequence: the protocol security will simply continue to rest on the passwords that remain. Since compromising ordinary users brings no other direct reward than to expose their passwords, it is just as if broken passwords were removed from the key in future protocol executions, or never contributed to it in the first place.

Of course, cracking the password of the *leader* will compromise the group and grant access to private computations (with the help of the other players, still), but that is only natural since the leader "owns" the group. There is an important distinction between exposure of an ordinary player's password and the leader's password: the leader represents the group with respect to third parties, i.e., when third parties use the group's public key their intention is to communicate with the leader. By contrast, ordinary players are not meant to be trusted and their inclusion to the group is a choice by the leader to help him or her increase the security of the private key — or leave it unchanged if that player turns out to be compromised — but never decrease it.

Revocation. In case of compromise of the leader password, it is possible for the leader to "revoke" the group by instructing the other players to stop participating in that group (e.g., by using the group's resources one last time to sign a revocation certificate using the group's private key). This will prevent any further use of the group's resources, unless of course the adversary manages to crack *all* of the players' passwords *jointly*. Such revocation mechanism falls outside of the protocol, so we do not model it in the functionalities.

User Corruptions. Our definition of the $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ functionality deals with user corruptions in a way that is quite different to that of other password-based group protocols. E.g., in the group key exchange functionality of [27], if the adversary has obtained the passwords of some participants (via password guesses or user corruptions), it may freely set the resulting session key to any value. Here, our functionalities are much more demanding in two important ways: first, $\mathcal{S}$ is much constrained in the way it can make and test online password guesses; second, $\mathcal{S}$ can never alter the computation in any way once it has started.

Password Tests. The first difference is that the testPwd query can only be asked once, early in the protocol, and it does not actually test the password of the users, but rather the compatibility between (1) the guessed passwords of any specified subset of users, (2) the real passwords of the rest of the group (known by the functionality thanks to the newSession queries), and (3) the public key (which at this stage is already guaranteed to be the same in all the users' views). This unusual shape for the testPwd query provides a very high level of security, because (A) at most a single set of password guesses can be tested against any player in any protocol instance, and (B) if $\mathcal{S}$ chooses to test a set of more than one password at once, then to cause a positive response all the guesses must be correct simultaneously (and since this becomes exponentially unlikely, the astute adversary should be content to test sets of one password at a time). After the private computation, all the records, initially fresh, compromised, or interrupted, become either complete or error. No more testPwd query is accepted at this stage, because once the users have completed their task it is too late for $\mathcal{S}$ to impersonate them (though corruption queries can still be made to read their state). Note that one testPwd query is allowed for each instance of $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$, several of which may be invoked by the split functionality $s\mathcal{F}_{\mathsf{pwDistPrivateComp}}$.

Robustness. The second difference with the model in [27] is that we do not grant the adversary the right to alter the computation result when corrupting some users or learning some passwords. This in particular means that either the group leader receives something coherent, or he receives an error; he cannot receive something wrong, which makes the protocol robust. Robustness is actually automatic if we make the assumption that the computation function $\mathsf{PrivateComp}$ is deterministic; for simplicity, this is the setting of the generic protocol described in detail in this paper. At the end, however, we shall mention some applications that require randomness in the computation. Without going into details, we can keep the protocol robust by having all the parties commit to their random coins in the first round, in the same way as they will also commit to their passwords (see below): this allows us to treat such coins as any regular private input in the model, and hence forbid the adversary from modifying them once the computation has started.

We remark that, although the adversary cannot spoof the computation, the environment does become aware of the completion of the protocol, and hence could distinguish between the ideal and the real worlds if the adversary won more often in one than the other. Such environmental awareness of the final state is of course to be expected in reality, and so it is natural that our model should capture it. (Our implementation will thus have to ensure that the success conditions are the same in both worlds.)

Implicit Corruptions. Because we have a set of initially unauthenticated players communicating over adversarially controlled channels, it is always possible for the adversary to partition the actual players into isolated islands [1], and act on behalf of the complement of players with respect to each island. We call this an implicit corruption, meaning that the adversary usurps the identity of a regular player (or players) from the very start, before the key generation is even initiated. The adversary then sends the $\mathsf{newSession}$ query on behalf of such implicitly corrupted players, who never really *became* corrupted but always *were* the adversary. As mentioned previously, this situation is modeled in the ideal world by the respective split functionalities $s\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $s\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ spawning one or more instances of the normal functionalities $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ over disjoint sets of (actual) players.

3 Protocol Description

The following protocol deals with a particular case of unauthenticated distributed private computation [1], as captured by our functionalities. Informally, assuming s to be a secret key, the aim of the protocol is to compute a value c^s given an element c of the group. This computation can be used to perform distributed BLS signatures [10], ElGamal decryptions [18], linear decryptions [8], and BF or BB1 identity-based key extraction [9,7].

Here we focus on ElGamal decryptions, relying on the DDH assumption. We emphasize that the protocol as given relies exclusively on DDH, not requiring any additional assumption; and that it can be easily modified to rely on the Decision Linear assumption for compatibility with bilinear groups [8].

Building Blocks. Let $\mathbb{G}$ be a group of prime order p, and g a generator of this group. We furthermore assume to be given an element h in $\mathbb{G}$ as a CRS. We use the following building blocks:

Password Selection. Each user P_i owns a privately selected password pw_i, to act as the i-th share of the secret key sk (see below). For convenience, we write $\mathsf{pw}_i = \mathsf{pw}_{i,1} \dots \mathsf{pw}_{i,\ell} \in \{0, \dots, 2^{L\ell} - 1\}$, i.e., we further divide each password pw_i into ℓ blocks $\mathsf{pw}_{i,j} \in \{0, \dots, 2^L - 1\}$ of L bits each, where $p < 2^{\ell L}$. The segmentation into blocks is a technicality to get efficient extractable commitments for long passwords: in the concrete scheme, for example, we shall use single-bit blocks in order to achieve the most efficient extraction (i.e, $L = 1$ and $\ell = 160$ for a 160-bit prime p). Notice that although we allow full-size passwords of up to $L\ell$ bits (the size of p), users are of course permitted to choose shorter passwords.

Password Combination. The private key sk is defined as the (virtual) combination of all the passwords pw_i. It does not matter how precisely such combination is done, as long as it is reproducible and preserves the joint entropy of the set of passwords (up to $\log_2 p$ bits, since that is the length of sk). For example, if there are n users, all with short passwords $\mathsf{pw}_i^* \in \{0, \dots, \Delta - 1\}$ with $\Delta^n < p$, defining $\mathsf{pw}_i = \Delta^i \mathsf{pw}_i^*$ and taking $\mathsf{sk} = \sum_i \mathsf{pw}_i$ will ensure that there are no "aliasing effects", or mutual cancellation of two or more passwords.

In general, it is preferable that each user independently transforms his or her true password pw_i^* into an effective password pw_i by applying a suitable extractor $\mathsf{pw}_i = \mathsf{H}(i, \mathsf{pw}_i^*, Z_i)$ where Z_i is any relevant public information such as a description of the group and its purpose. We can then safely take $\mathsf{sk} = \sum_i \mathsf{pw}_i$ and be assured that the entropy of sk will closely match the joint entropy of the vector $(\mathsf{pw}_1^*, \dots, \mathsf{pw}_n^*)$ taken together. Such password pre-processing using hashing is very standard but falls outside of the functionalities proper.

Public and Private Keys. We use the (effective) passwords pw_i to define a key pair $(\mathsf{sk}, \mathsf{pk} = g^{\mathsf{sk}})$ for a password-based ElGamal key encapsulation mechanism (KEM). Based on the above, we define $\mathsf{sk} = \mathsf{SecretKeyGen}(\mathsf{pw}_1, \dots, \mathsf{pw}_n) \overset{\mathsf{def}}{=} \sum_{i=1}^n \mathsf{pw}_i$ and $\mathsf{pk} = \mathsf{PublicKeyGen}(\mathsf{pw}_1, \dots, \mathsf{pw}_n) \overset{\mathsf{def}}{=} g^{\sum \mathsf{pw}_i}$. The public-key verification function is then $\mathsf{PublicKeyVer}(\mathsf{pw}_1, \dots, \mathsf{pw}_n, \mathsf{pk}) \overset{\mathsf{def}}{=} \left(\mathsf{pk} \overset{?}{=} g^{\sum \mathsf{pw}_i}\right)$.

The ElGamal KEM public-key operation is the encapsulation $\mathsf{Enc} : (\mathsf{pk}, r) \mapsto (c = g^r, m = \mathsf{pk}^r)$, which outputs a random session key m and a ciphertext c. The private-key operation is the decapsulation $\mathsf{Dec} : (\mathsf{sk}, c) \mapsto m = c^{\mathsf{sk}}$, which here is deterministic. Observe that whereas Dec instantiates $\mathsf{PrivateComp}$ in the functionalities, Enc is intended for public third-party usage and never appears in the private protocols.

Entropy Preservation. In order for the low password entropies to combine nicely in the secret key $\mathsf{sk} = \sum_i \mathsf{pw}_i$, the effective pw_i must be properly "decoupled" to avoid mutual cancellations, as just discussed.

We note that, even with the kind of shuffling previously considered, it is quite possible that the actual entropy of sk will be smaller than its maximum value of $\log_2 p$ bits, e.g., if there are not enough non-corrupted users or if their passwords are too small. Nevertheless, there is no known effective attack against discrete logarithm and related problems that can take advantage of any reduced entropy of sk, barring an exhaustive search over the space of possible values. Specifically, regardless of how the passwords are actually combined, one could easily prove that no generic attack [29] can solve the discrete logarithm or the DDH problem in less than $\sqrt{2^h}$ operations, where h is the min-entropy of the private key sk conditionally on all known passwords.

Computational Assumption. Our concrete protocols rely on the Decisional Diffie-Hellman (DDH) assumption, stated here for completeness: Let $\mathbb{G} = \langle g \rangle$ be a multiplicative abelian cyclic group of prime order p. For random $x, y, z \in \mathbb{Z}_p^*$, it is computationally intractable to distinguish (g, g^x, g^y, g^{xy}) from (g, g^x, g^y, g^z).

Extractable Homomorphic Commitments. The first step of our distributed decryption protocol is for each user to commit to his password (the details are given in the following section). The commitment needs to be extractable, homomorphic, and compatible with the shape of the public key. Generally speaking, one needs a commitment $\mathsf{Commit}(\mathsf{pw}, r)$ that is additively homomorphic on pw and with certain properties on r. In order to simplify the following description of the protocols, we chose to use ElGamal's scheme [18], which is additive on the random value r, and given by: $\mathsf{Commit}_v(\mathsf{pw}, r) = (v^{\mathsf{pw}} h^r, g^r)$. The semantic security relies on the above DDH assumption. Extractability is possible granted the decryption key x, such that $h = g^x$ in the common reference string.

Simulation-Sound Non-Interactive Zero-Knowledge Proofs. Informally speaking, a zero-knowledge proof system is said to be simulation-sound if it has the property that an adversary cannot give a convincing proof for a false statement, even if it has oracle access to the zero-knowledge simulator. We also require non-malleability, which is to say that a proof of some theorem cannot be turned into a proof of another theorem. De Santis *et al.* proved in [16] the existence of such a scheme, with the additional property of being non-interactive, if we assume the existence of one-way trapdoor permutations. Note that their scheme allows for multiple simulations with a unique common random string (CRS), which is crucial for the multi-session case. If we instantiate all the SSNIZK proofs with those, then our protocols are UC-secure in the CRS model.

However, for sake of efficiency, we can instead instantiate them using Schnorr-like proofs of equality of discrete logarithms [20], which rely on the random-oracle model [4], but are significantly more practical. These SSNIZK are well-known (see details in the full version and their proofs in [20]), but along these lines, we use the notation $\mathsf{SSNIZK}(\mathcal{L}(w))$ for a proof that w lies in the language $\mathcal{L}$. More precisely, $\mathsf{CDH}(g, G, h, H)$ will state that (g, G, h, H) lies in the CDH language: there exists a common exponent x such that $G = g^x$ and $H = h^x$.

$$(1a)\ r_{i,j} \overset{R}{\leftarrow} \mathbb{Z}_q^* \qquad C_{i,j} = \mathsf{Commit}_g(\mathsf{pw}_{i,j}, r_{i,j}) = (g^{\mathsf{pw}_{i,j}} h^{r_{i,j}}, g^{r_{i,j}})$$

$$\Pi^0_{i,j} = \mathsf{SSNIZK}(\mathsf{CDH}(g, C^{(2)}_{i,j}, h, C^{(1)}_{i,j}) \vee \mathsf{CDH}(g, C^{(2)}_{i,j}, h, C^{(1)}_{i,j}/g))$$

$$C_i = \{C_{i,j}\}_j, \{\Pi^0_{i,j}\}_j \qquad \xrightarrow{C_i}$$

$$(1b)\ H = \mathcal{H}(C_1, \ldots, C_n) \qquad s_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$$

$$C'_i = \mathsf{Commit}^H_g(\mathsf{pw}_i, s_i) = (g^{\mathsf{pw}_i} h^{s_i}, g^{s_i}, H)$$

$$\Pi^1_i = \mathsf{SSNIZK}^H(\mathsf{CDH}(g, C'^{(2)}_i / \textstyle\prod_j C^{(2)}_{i,j}, h, C'^{(1)}_i / \prod_j C^{(1)}_{i,j})) \qquad \xrightarrow{C'_i, \Pi^1_i}$$

$$(1c)\ \gamma^{(1)}_0 = \textstyle\prod_i C'^{(1)}_i = g^{\sum_i \mathsf{pw}_i} h^{\sum_i s_i} \quad \gamma^{(2)}_0 = h$$

$$\text{given, for } j = 1, \ldots, i-1 \qquad (\gamma^{(1)}_j, \gamma^{(2)}_j, \Pi^2_j)$$

$$\text{check } \Pi^2_j \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(\gamma^{(1)}_{j-1}, \gamma_j^{(1),\gamma^{(2)}_{j-1}}, \gamma^{(2)}_j)$$

$$\alpha_i \overset{R}{\leftarrow} \mathbb{Z}_q^* \quad \gamma^{(1)}_i = (\gamma^{(1)}_{i-1})^{\alpha_i} \quad \gamma^{(2)}_i = (\gamma^{(2)}_{i-1})^{\alpha_i}$$

$$\Pi^2_i = \mathsf{SSNIZK}(\mathsf{CDH}(\gamma^{(1)}_{i-1}, \gamma^{(1)}_i, \gamma^{(2)}_{i-1}, \gamma^{(2)}_i)) \qquad \xrightarrow{\gamma^{(1)}_i, \gamma^{(2)}_i, \Pi^2_i}$$

$$(1d)\ \text{given } \gamma^{(1)}_n = g^{\alpha \sum_i \mathsf{pw}_i} h^{\alpha \sum_i s_i} \quad \gamma^{(2)}_n = h^\alpha$$

$$\text{check } \Pi^2_n \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(\gamma^{(1)}_{n-1}, \gamma^{(1)}_n, \gamma^{(2)}_{n-1}, \gamma^{(2)}_n)$$

$$h_i = (\gamma^{(2)}_n)^{s_i} \qquad \Pi^3_i = \mathsf{SSNIZK}(\mathsf{CDH}(g, C'^{(2)}_i, \gamma^{(2)}_n, h_i)) \qquad \xrightarrow{h_i, \Pi^3_i}$$

$$(1e)\ \text{given, for } j = 1, \ldots, n \qquad (h_j, \Pi^3_j)$$

$$\text{check } \Pi^3_j \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(g, C'^{(2)}_j, \gamma^{(2)}_n, h_j))$$

$$\zeta_{n+1} = \gamma^{(1)}_n / \textstyle\prod_j h_j = g^{\alpha \sum_j \mathsf{pw}_j}$$

$$\text{given, for } j = n, \ldots, i+1 \qquad (\zeta_j, \Pi^4_j)$$

$$\text{check } \Pi^4_j \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(\gamma^{(1)}_{j-1}, \gamma^{(1)}_j, \zeta_j, \zeta_{j+1}))$$

$$\zeta_i = (\zeta_{i+1})^{1/\alpha_i} \quad \Pi^4_i = \mathsf{SSNIZK}(\mathsf{CDH}(\gamma^{(1)}_{i-1}, \gamma^{(1)}_i, \zeta_i, \zeta_{i+1})) \qquad \xrightarrow{\zeta_i, \Pi^4_i}$$

$$(1f)\ \text{given, for } j = i-1, \ldots, 1 \qquad (\zeta_j, \Pi^4_j)$$

$$\text{check } \Pi^4_j \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(\gamma^{(1)}_{j-1}, \gamma^{(1)}_j, \zeta_j, \zeta_{j+1}))$$

$$\mathsf{pk} = \zeta_1$$

Fig. 3. Individual steps of the distributed key generation protocol

Intuition. We first describe the distributed decryption algorithm. All the users are provided with a password pw_i, a public key pk, and a ciphertext c. One of them is the leader of the group, denoted by P_1, and the others are $P_2, \ldots, P_n$. For this given ciphertext $c \in \mathbb{G}$, the leader wants to obtain $m = c^{\mathsf{sk}}$. But before computing this value, everybody wants to be sure that all the users are honest, or at least that the combination of the passwords is compatible with the public key.

The protocol starts by verifying that they will be able to decrypt the ciphertext, and thus that they indeed know a representation of the decryption key into shares. Each user sends a commitment C_i of his password. As we see in the proof (see full version), this commitment needs to be extractable so that the simulator is able to recover the passwords used by the adversary: this is a requirement of the UC model, as in [14]. Indeed, the simulator needs to be able to simulate everything without knowing any passwords, he thus recovers the passwords by extracting them from the commitments C_i made by the adversary in this first round, enabling him to adjust his own values before the subsequent commitments, so that all the passwords are compatible with the public key (if they should be in the situation at hand). If we think in terms of ElGamal encryption, the extraction is proportional in the square root of the size of the alphabet, which would be practical for 20-bit passwords but not 160-bit ones (and even if passwords are usually small, we do not want to restrict the size of the passwords). This is the reason why we segmented all the passwords into small blocks: to commit to them block by block. In our concrete description, blocks are of size 1, which will help to make the proof of validity: ElGamal encryption of one bit.

Once this first step is done, the users commit again to their passwords. The new commitments C'_i will be the ones used in the rest of the protocol. They need not be segmented (since we will not extract anything from them), but we ask the users to prove that they are compatible with the former ones. Note that they use the three values $H = \mathcal{H}(C_1, \ldots, C_n)$ (where $\mathcal{H}$ is a collision-resistant hash function), pk, and c, as "labels" of these commitments (see below), to avoid malleability and replay from the previous sessions, granted the SSNIZK proofs that include and thus check these labels.

Next, the users make yet another commitment A_i to their passwords, but this time they do an ElGamal encryption of pw_i in base c instead of in base g (in the above C'_i commitment). That is, each user computes $A_i = (c^{\mathsf{pw}_i} h^{t_i}, g^{t_i})$. The commitment C'_i will be used to check the possibility of the decryption (that it is consistent with $\mathsf{pk} = g^{\mathsf{sk}}$), whereas A_i will be used to actually compute the decryption c^{sk}, hence the two different bases g and c in C'_i and A_i, respectively.

All the users send these last two commitments to everybody, along with a SSNIZK proof that the same password was used each time. These proofs are "labeled" by H, pk, and c, and the verification by the other users will succeed only if their "labels" are identical. This enables all the players to check that everybody shares the same public key pk and the same ciphertext c. It thus avoids situations in which a group leader with an incorrect key obtains a correct decryption message, contrary to the ideal functionality. The protocol will thus fail if H, pk, or c is not the same to everyone, which is the result required by the ideal functionality. Note that the protocol will also fail if the adversary drops or modifies a flow received by a user, even if everything was correct (compatible passwords, same public key, same ciphertext). This situation is modeled in the functionality by the bit b of the key/decryption delivery queries, for when everything goes well but the group leader does not obtain the result.

After these rounds of commitments, a verification step allows for the group leader, but also all the players, to check whether the public key and the passwords are compatible. Note that at this point, everything has become publicly verifiable so that the group leader will not be able to cheat and make the other players believe that everything is correct when it is not. Verification starts from the commitments $C'_i = (C'^{(1)}_i, C'^{(2)}_i)$, and involves two "blinding rings" to raise the two values $\prod_i C'^{(1)}_i$ and $\prod_i C'^{(2)}_i$ to some distributed random exponent $\alpha = \sum_i \alpha_i$. The ratio of the blinded values is taken to cancel the $h^{\sum_i s_i}$, leaving $g^{\alpha \mathsf{sk}}$. A final "unblinding ring" is applied to remove the exponent α and expose g^{sk}. This ends with a decision by the group leader on whether to abort the protocol (when the passwords are incompatible with the public key) or go on to the computation step. We stress that every user is able to check the validity of the group leader's decision: A dishonest execution cannot continue without an honest user becoming aware of it (and aborting it). Note however that an honest execution can also be stopped by a user if the adversary modifies a flow destined to it, as reflected by the bit b in the ideal functionality.

If the group leader decides to go on, the players assist in the computation of c^{sk}, again with the help of two blinding and one unblinding rings, starting from the commitments A_i. Note that if at some point a user fails to send its value to everyone (for instance due to a denial of service attack) or if the adversary modifies a flow (in a man-in-the-middle attack), the protocol will fail. In the ideal world this means that the simulator makes a decryption delivery with a bit b set to zero. Because of the SSNIZK proofs, in these decryption rounds exactly the same sequence of passwords as in the first rounds has to be used by the players. This necessarily implies compatibility with the public key, but may be a stronger condition.

As a side note, observe that all the blinding rings in the verification and the computation steps could be made concurrent instead of sequential, in order to simplify the protocol. Notice however that the final unblinding ring of c^{sk} in the computation step should only be carried out after the public key and the committed passwords are known to be compatible, and the passwords to be the same in both sequences of commitments, *i.e.* after the verification step succeeded.

We show in the full version that we can *efficiently* simulate these computations without the knowledge of the pw_i's, so that they do not reveal anything more about the pw_i's than pk already does. More precisely, we show that such computations are indistinguishable to $\mathcal{A}$ under the DDH assumption.

The key generation protocol (computation of $\mathsf{pk} = g^{\mathsf{sk}}$) is a special case of the decryption protocol outlined above (computation of g^{sk}, test that $g^{\mathsf{sk}} = \mathsf{pk}$, computation of $m = c^{\mathsf{sk}}$), only simpler. Indeed, we only need one set of commitments for the last rounds of blinding/unblinding, as we omit all the prior verifications (since there is nothing to verify when the key is first set up).

We refer to the full version for the precise details of the protocols (see Figures 3 and 4), in particular the exact definition of the languages for the SSNIZK proofs, and the proofs of the following security theorems. Our protocol is proven secure against static adversaries only, that are allowed to corrupt players prior to the beginning of the protocol execution.

Theorem 1. *Let $\widehat{\mathcal{F}}_{\mathsf{pwDistPublicKeyGen}}$ be the concurrent multi-session extension of $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$. The distributed key generation protocol in Figure 3 securely realizes $\widehat{\mathcal{F}}_{\mathsf{pwDistPublicKeyGen}}$ for ElGamal key generation, in the CRS model, in the presence of static adversaries, provided that DDH is infeasible in $\mathbb{G}$, $\mathcal{H}$ is collision-resistant, and SSNIZK proofs for the CDH language exist.*

Theorem 2. *Let $\widehat{\mathcal{F}}_{\mathsf{pwDistPrivateComp}}$ be the concurrent multi-session extension of $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$. The distributed decryption protocol in Figure 4 securely realizes $\widehat{\mathcal{F}}_{\mathsf{pwDistPrivateComp}}$ for ElGamal decryption, in the CRS model, in the presence of static adversaries, provided that DDH is infeasible in $\mathbb{G}$, $\mathcal{H}$ is collision-resistant, and SSNIZK proofs for the CDH language exist.*

$$(2a) = (1a) \qquad \xrightarrow{\{C_{i,j}, \Pi^0_{i,j}\}_j}$$

$$(2b) = (1b) \quad \text{except}$$

$$C'_i = \mathsf{Commit}_g^{H,\mathsf{pk},c}(\mathsf{pw}_i, s_i) = (g^{\mathsf{pw}_i} h^{s_i}, g^{s_i}, H, \mathsf{pk}, c)$$

$$A_i = \mathsf{Commit}_c(\mathsf{pw}_i, t_i) = (c^{\mathsf{pw}_i} h^{t_i}, g^{t_i})$$

$$\Pi^1_i = \mathsf{SSNIZK}^{H,\mathsf{pk},c}(\mathsf{CDH}(g, {C'_i}^{(2)} / \textstyle\prod_j C^{(2)}_{i,j}, h, {C'_i}^{(1)} / \textstyle\prod_j C^{(1)}_{i,j}))$$

$$\bar{\Pi}^1_i = \mathsf{SSNIZK}(C'_i \overset{g,c}{\approx} A_i) \qquad \xrightarrow{C'_i, A_i, \Pi^1_i, \bar{\Pi}^1_i}$$

$$(2c) = (1c) \qquad \xrightarrow{\gamma^{(1)}_i, \gamma^{(2)}_i, \Pi^2_i}$$

$$(2d) = (1d) \qquad \xrightarrow{h_i, \Pi^3_i}$$

$$(2e) = (1e) \qquad \xrightarrow{\zeta_i, \Pi^4_i}$$

$$(2f) = (1f) \qquad \mathsf{pk} \overset{?}{=} \zeta_1$$

$$(3a)\ \delta^{(1)}_0 = \textstyle\prod_i A^{(1)}_i = c^{\sum_i \mathsf{pw}_i} h^{\sum_i t_i} \qquad \delta^{(2)}_0 = h$$

$$\text{given, for } j = 1, \ldots, i-1 \qquad (\delta^{(1)}_j, \delta^{(2)}_j, \Pi^5_j)$$

$$\text{check } \Pi^5_j \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(\delta^{(1)}_{j-1}, \delta^{(1)}_j, \delta^{(2)}_{j-1}, \delta^{(2)}_j)$$

$$\beta_i \overset{R}{\leftarrow} \mathbb{Z}^*_q \qquad \delta^{(1)}_i = (\delta^{(1)}_{i-1})^{\beta_i} \qquad \delta^{(2)}_i = (\delta^{(2)}_{i-1})^{\beta_i}$$

$$\Pi^5_i = \mathsf{SSNIZK}(\mathsf{CDH}(\delta^{(1)}_{i-1}, \delta^{(1)}_i, \delta^{(2)}_{i-1}, \delta^{(2)}_i)) \qquad \xrightarrow{\delta^{(1)}_i, \delta^{(2)}_i, \Pi^5_i}$$

$$(3b)\ \text{given } \delta^{(1)}_n = c^{\beta \sum_i \mathsf{pw}_i} h^{\beta \sum_i t_i} \qquad \beta^{(2)}_n = h^\beta$$

$$h'_i = (\delta^{(2)}_n)^{t_i} \qquad \Pi^6_i = \mathsf{SSNIZK}(\mathsf{CDH}(g, {A_i}^{(2)}, \delta^{(2)}_n, h'_i)) \qquad \xrightarrow{h'_i, \Pi^6_i}$$

$$(3c)\ \text{given, for } j = 1, \ldots, n \qquad (h'_j, \Pi^6_j)$$

$$\text{check } \Pi^6_j \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(g, {A_j}^{(2)}, \delta^{(2)}_n, h'_j))$$

$$\zeta'_{n+1} = \delta^{(1)}_n / \textstyle\prod_j h'_j = c^{\beta \sum_j \mathsf{pw}_j}$$

$$\text{If } i \neq 1, \text{ given, for } j = n, \ldots, i+1 \qquad (\zeta'_j, \Pi^7_j)$$

$$\text{check } \Pi^7_j \overset{?}{=} \mathsf{SSNIZK}(\mathsf{CDH}(\delta^{(1)}_{j-1}, \delta^{(1)}_j, \zeta'_j, \zeta'_{j+1}))$$

$$\zeta'_i = (\zeta'_{i+1})^{1/\beta_i} \qquad \Pi^7_i = \mathsf{SSNIZK}(\mathsf{CDH}(\delta^{(1)}_{i-1}, \delta^{(1)}_i, \zeta'_i, \zeta'_{i+1})) \qquad \xrightarrow{\zeta'_i, \Pi^7_i}$$

$$(3d)\ P_1 \text{ gets } \zeta'_1 = (\zeta'_2)^{1/\beta_1} = c^{\sum \mathsf{pw}_i} = c^{\mathsf{sk}}$$

Fig. 4. Individual steps of the distributed decryption protocol

4 Discussion and Conclusion

In this work, we have brought together ideas from secret sharing, threshold cryptography, password-based protocols, and multi-party computation, to devise a practical approach to (distributed) password-based public-key cryptography. For a given cryptosystem, the objective was to define, from a set of user-selected weak passwords held in different locations, a virtual private key that is as strong and resistant to attacks as any regular key, and that can be used in a distributed manner without ever requiring its actual reconstitution.

We proposed general definitions of such functionalities in the UC model, carefully justifying all our design choices along the way. In particular, we saw that it is mandatory to require the presence of a "group leader" who directs the private computation process and solely obtains its end result. We then constructed explicit protocols for the simple but instructive case of ElGamal encryption.

Specifically, relying on the DDH assumption, we constructed and proved the security of two ElGamal key generation and decryption protocols, whose private key is virtual and implied by a distributed collection of arbitrary passwords.

To conclude, we now argue that the approach outlined in this paper is in fact quite general and has broad applications. It can of course be viewed as a restriction of the Unauthenticated MPC framework of [1]; but this would be missing the point, since as often in the UC model, much (or most) of the work has been done once the functionality definitions have been laid down. The functionalities that we have carefully crafted here should apply essentially without change to most kinds of public-key primitives.

The protocols also generalize easily beyond ElGamal decryption. The same method that let us compute c^{sk} from a distributed $\mathsf{sk} = \langle \mathsf{pw}_1, \ldots, \mathsf{pw}_n \rangle$, can also compute pairs of vectors $(\boldsymbol{c}_i^{\mathsf{sk}}, \boldsymbol{c}_j^{r})$ for a random ephemeral r contributed by all the players — or, precisely, for $r = \sum_i r_i$ where each r_i is initially committed to by each player, in a similar way as they initially commit to their passwords. By the hiding and binding properties of the commitments this guarantees that r is uniform and unpredictable if at least one player draws r_i at random.

Remarkably, this is enough to let us do "password-based distributed IBE", where the private-key generator is decentralized over a set of users, each of them holding only a short private password of their own choosing. PrivateComp is now a key extraction function that maps user identities id to user decryption keys d_{id}. To get: **"Password-based" Boneh-Franklin (BF) IBE** [9], we need to compute $d_{id} = H(id)^{\mathsf{sk}}$ where $H(id)$ is a public hash of a user's identity. This is analogous to c^{sk}, and thus our protocol works virtually unchanged. To get: **"Password-based" Boneh-Boyen (BB$_1$) IBE** [7], here d_{id} is randomized and of the form $(g_0^{\mathsf{sk}}(g_1^{id} g_2)^r, g_3^r)$. This fits the general form of what we can compute by adding ephemerals to our protocol as just discussed.

Note that in some bilinear groups the DDH problem is easy: in those groups, we must replace DDH-based commitments with ones based on a weaker assumption, such as D-Linear [8]; such changes are straightforward.

Acknowledgments

This work was supported in part by the French ANR-07-SESU-008-01 PAMPA Project. The second author thanks ECRYPT and the hospitality of ENS.

References

1. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
2. Beaver, D., Goldwasser, S.: Multiparty computation with faulty majority. In: 30th FOCS, pp. 468–473. IEEE Computer Society Press, Los Alamitos (1989)
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)

4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
5. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
6. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for noncryptographic fault-tolerant distributed computations. In: 20th ACM STOC, pp. 1–10. ACM Press, New York (1988)
7. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
8. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
9. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
11. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
12. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)
13. Canetti, R., Halevi, S., Herzberg, A.: Maintaining authenticated communication in the presence of break-ins. Journal of Cryptology 13(1), 61–105 (2000)
14. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
15. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: 20th ACM STOC, pp. 11–19. ACM Press, New York (1988)
16. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
17. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
18. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
19. Fitzi, M., Gottesman, D., Hirt, M., Holenstein, T., Smith, A.: Detectable byzantine agreement secure against faulty majorities. In: 21st ACM PODC, pp. 118–126. ACM Press, New York (2002)
20. Fouque, P.-A., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 351–368. Springer, Heidelberg (2001)
21. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)

22. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
23. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In: 19th ACM STOC, pp. 218–229. ACM Press, New York (1987)
24. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (1987)
25. Holtby, D., Kapron, B.M., King, V.: Lower bound for scalable Byzantine agreement. In: 25th ACM PODC, pp. 285–291. ACM Press, New York (2006)
26. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
27. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: ACM CCS 2005, pp. 180–189. ACM Press, New York (2005)
28. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: 21st ACM STOC, pp. 73–85. ACM Press, New York (1989)
29. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
30. Yao, A.C.: Protocols for secure computations. In: 23rd FOCS, pp. 160–164. IEEE Computer Society Press, Los Alamitos (1982)

Asynchronous Multiparty Computation: Theory and Implementation

Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen*

Dept. of Computer Science, University of Aarhus, Denmark
{ivan,mg,mk,buus}@cs.au.dk

Abstract. We propose an asynchronous protocol for general multiparty computation. The protocol has perfect security and communication complexity $\mathcal{O}(n^2|C|k)$, where n is the number of parties, $|C|$ is the size of the arithmetic circuit being computed, and k is the size of elements in the underlying field. The protocol guarantees termination if the adversary allows a preprocessing phase to terminate, in which no information is released. The communication complexity of this protocol is the same as that of a passively secure solution up to a constant factor. It is secure against an adaptive and active adversary corrupting less than $n/3$ players. We also present a software framework for implementation of asynchronous protocols called VIFF (Virtual Ideal Functionality Framework), which allows automatic parallelization of primitive operations such as secure multiplications, without having to resort to complicated multithreading. Benchmarking of a VIFF implementation of our protocol confirms that it is applicable to practical non-trivial secure computations.

1 Introduction

A general multiparty computation protocol is an extremely powerful tool that allows n parties to compute any agreed function $f(x_1, \ldots, x_n)$, where each input x_i is privately held by the i'th player P_i, and where only the intended result becomes known to the players. The function is often represented by an arithmetic circuit C over some suitable finite field $\mathbb{F}$. It is required that privacy of the inputs and correctness of the result is ensured even in the presence of an adversary who may corrupt some number t of the players.

From the basic feasibility results of the late 80-ties [4,7], it follows that any efficiently computable function may be computed securely in the model where players have secure point-to-point channels, if and only if $t < n/3$. In case the adversary is passive, i.e., corrupted players follow the protocol, the bound is $t < n/2$. Under computational assumptions, $t < n/2$ corruptions can be tolerated even if the adversary is active, i.e., corrupt players behave arbitrarily [10].

The solution from [4] with passive security can lead to quite practical solutions, when combined with later techniques for optimizing the efficiency of particular primitives, such as integer comparison – even to the point where large-scale practical applications can be handled [5].

* Supported by Ministry of Science, Technology and Innovation.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 160–179, 2009.

On the other hand, this type of solution is not satisfactory in all cases. It is of course desirable to provide security against active cheating. However, this usually incurs a large cost in terms of efficiency. Techniques have been proposed to reduce this cost [11], but they – like most previous protocols – are designed for synchronous communication. Common ways to communicate, such as the Internet, are arguably better modeled as asynchronous networks, where there is no guarantee that a message is delivered before a certain time. Note that the way we model the network can have dramatic consequences for the practical efficiency of a protocol. If we run a synchronous protocol on top of any real network, we are forced to make every round last enough time so that we can be sure that all messages from honest players have been delivered. Otherwise, we may conclude that an honest player is corrupt because he did not send the message he was supposed to, and take action accordingly. Now, of course, the protocol is no longer secure. It follows that, for instance, on a network that usually delivers messages fast, but occasionally takes much longer time, a synchronous protocol may be much slower in practice than an asynchronous one, where every player may continue as soon as he has enough information to do so.

In the project reported on in this paper, our goal was therefore to develop and implement a practical general MPC protocol, with active security on an asynchronous network. Compared to the usual model for asynchronous MPC, we make two modifications, both of which we believe are well motivated:

- We allow our protocol to have one synchronization point. More precisely, the assumption is that we can set a certain time-out, and all messages sent by honest players before the deadline will also be delivered before the deadline.
- We do not guarantee that the protocol always terminates and gives output to all honest players. Instead we require the following: The preprocessing phase of the protocol, up to the synchronization point, never releases any new information to the adversary. The adversary may cause the preprocessing to fail, but if it terminates successfully, the entire protocol is guaranteed to terminate with output to all honest parties.

A discussion of this model: Without the first assumption, i.e., if the protocol is fully asynchronous, one cannot guarantee that all honest players will be able to contribute input since the protocol cannot distinguish between t corrupt players that have not sent anything, and t honest players whose messages have been delayed. We believe that in most practical applications, this is not acceptable, and this is why we introduce the synchronization point, it is a minimal assumption allowing all honest players to provide input. We stress that the protocol is *asynchronous both before and after the synchronization point.* In other words, a protocol in this model is free to harvest the efficiency gain that follows from players being able to proceed as soon as possible. The only constraint we put is that honest players must reach the deadline on time, so we can have agreement on whether the preprocessing succeeded.

On the second point, although we do give the adversary extra power to stop the protocol, this is arguably of no use in practice: If the corrupted players can only stop the protocol at a point where they have learned nothing new, they have very little incentive to do so.

In this model, assuming secure point-to-point channels and that Byzantine agreement is available, we present a protocol that is perfectly secure against an adaptive and active adversary corrupting less than $n/3$ of the players. The communication and computational complexities (total communication and work done) are $\mathcal{O}(n^2|C|k)$ where $|C|$ is the size of the arithmetic circuit being computed and k is the bit length of elements in the field used. It is noteworthy that a straightforward implementation with only passive security would have the same asymptotic complexity, all other things being equal.

As for any protocol in the point-to-point model, the exact security properties of an actual implementation of our protocol depend on how the point-to-point channels and – in our case – the Byzantine agreement are implemented. The choice of implementation does not, however, affect the complexities since the Byzantine agreement is only used once. In a typical implementation where one goes for efficiency – such as the one we present below – one would use standard encryption tools to implement the channels and do the Byzantine agreement based on public-key signatures. This gives a protocol with computational security against a static adversary (also, such an implementation is not known to be insecure against an adaptive adversary).

In recent concurrent work, Hirt *et al.* [12] construct an asynchronous protocol of similar asymptotic complexity as ours. This protocol is fully asynchronous, so it does not guarantee that all honest parties can provide inputs, and it is computationally secure against a static adversary. In another recent work Beerliová-Trubíniová *et al.* [3] present a protocol with a single synchronization point like we have. This protocol guarantees termination, has a better security threshold ($n/2$), but is only computationally secure against a static adversary, and has larger asymptotic complexity than our protocol.

Regarding efficiency in practice, we stress that although our implementation is only computationally secure, it is an advantage, also from a practical point of view, that the basic protocol is information theoretic, because the tools used (secret sharing etc.) are much more efficient than computationally secure alternatives, such as homomorphic public-key encryption. Such techniques are used in both [3] and [12], making them much less efficient in practice than our construction.

Thus, our result is incomparable to previous work, and we believe it provides a new tradeoff between security properties that is attractive in practice. We later give more exact numeric evidence of the efficiency.

Our protocol is based on Beaver's well known circuit randomization techniques, where one creates in a preprocessing phase shared random values a, b, c with $ab = c$. We show two techniques for generating these triples efficiently. One is a variant of the protocol from [2], the other is based on pseudorandom secret sharing [8], it is much faster for a small number of players, but only gives computational security. Both protocols are actually synchronous, but we handle this via a new technique that may be of independent interest, namely a general method by which – if one accepts that the protocol may abort – a synchronous protocol can be executed in an asynchronous fashion, using a single synchronization point to decide if the protocol succeeded.

A crucial observation we make is that if the protocol is based on Shamir secret sharing with threshold less than $n/3$, then the computation phase can be done asynchronously and still guarantee termination, if the preprocessing succeeded.

A final contribution of our paper is a software framework called VIFF, short for Virtual Ideal Functionality Framework. It provides a platform on which general MPC protocols can be implemented, and we use it later in the paper to benchmark our protocol. Protocols implemented in VIFF can be compositions of basic primitives like addition and multiplication of secret-shared values, or one can implement new primitives. VIFF is basically asynchronous and operates on the principle that players proceed whenever possible (but can handle synchronization points when asked to do so). This allows us to provide all protocol implementations with automatic parallel scheduling of the operations, i.e., the programmer does not have to explicitly use multithreading, for instance, or specify any explicit timing of operations.

When players distributed across a large network execute a large protocol, it is very important to be able to run as much as possible in parallel in order to lower the cost per operation of the network delays. Round-trip times across the Internet are typically in the order of 100–200 milliseconds, but when executing many multiplications in parallel we are able to obtain an average time of just 2 milliseconds per secure multiplication of 32-bit numbers, using a standard implementation based on Shamir secret-sharing, for 3 players and passive security.

Furthermore, the ability to program asynchronously is very important towards having simpler code: If the protocol to be implemented is synchronous, one has to implement waiting to make sure that all messages from the current round have arrived, and the actual waiting time has to be chosen correctly with respect to the network we use. This means that the software now depends on the underlying network which is clearly undesirable, as it creates an extra source of errors, insecurity, or both.

2 Preliminaries

For an element $x \in \mathbb{F}$ we let $[x]_d$ denote a set of Shamir shares [14] of x computed using threshold/degree d. We use the shorthand $[x]$ for sharings $[x]_t$ where t is the number of corrupted players, so that $t < n/3$. We use the notation $[x] + a[y]$ where a is a public constant to denote the set of shares obtained by locally adding the share of x to the share of y times a. Since Shamir sharing is linear, we have $[x] + a[y] = [x + ay]$.

When in the following, we say that x is *publicly reconstructed* from $[x]_t$, where at most $t < n/3$ players are actively corrupted, this simply means that each player sends his share to all other players. This allows all honest players to reconstruct x using standard decoding techniques since $t < n/3$. We may also *privately open* x to player P_i by sending shares only to him.

3 Overview and Security Model

The goal of the protocol is to securely compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$. For notational convenience we assume that all inputs and outputs are single field elements. In addition each y_i can assume the value $y_i = \perp$, which indicates to P_i that the computation failed.

3.1 Overview of the Protocol

Our protocol consists of two phases, the *preprocess and input phase* and the *computation phase.*

Preprocessing and input phase. In the preprocessing phase, we can make use of any protocol that can generate a given number of multiplication triples, i.e., random secret-shared values $[a], [b], [c]$ where $ab = c$. In addition, for each player P_i, it should construct a secret sharing $[r_i]$ where r_i is random and reveal r_i privately to P_i. The protocol ends by outputting "success" or "failure" to all players, depending on whether the required values were successfully constructed or not. The purpose of $[r_i]$ is to allow P_i to contribute his input x_i securely by broadcasting $r_i + x_i$.

Instead of attempting to build such a protocol directly for the asynchronous model, it is much easier to design a protocol for the synchronous model with broadcast, we give two examples of this in Section 4. We then show below a special way to run any such protocol in an asynchronous way, i.e., we can avoid the use of timeouts after each communication round and we avoid having to implement broadcast. The price we pay for this is that the adversary can force the preprocessing to fail.

The basic idea is that in each round all parties just wait for messages from all other parties and progress to the next round immediately if and when they all arrived. Some extra technicalities are needed to make sure there is agreement at the end on whether the preprocessing succeeded, and to make sure that no information on the inputs is revealed prematurely.

To emulate a synchronous protocol with R rounds, each P_j proceeds as follows:

1. Wait for an input $\mathsf{begin\ preprocess}$. Let $r = 1$ and for each P_i compute the message $m_{j,i,1}$ to be sent to P_i in the first round of the synchronous protocol. Also compute the message $m_{j,1}$ to be broadcast in the first round.
2. Send $(m_{j,i,1}, m_{j,1})$ to P_i.
3. While $r \leq R$:
   (a) Wait until a message $(m_{i,j,r}, m_{i,r})$ arrived from all P_i.
   (b) From the incoming messages $((m_{1,j,r}, m_{1,r}), \ldots, (m_{n,j,r}, m_{n,r}))$ compute the messages $(m_{j,1,r+1}, \ldots, m_{j,n,r+1})$ that the preprocessing protocol wants to send in the next round, and the message $m_{j,r+1}$ to be broadcast.
   (c) $r := r + 1$.
4. Let $g_j \in \{\mathsf{preprocess\ success}, \mathsf{preprocess\ failure}\}$ denote the output of the preprocessing protocol and let M_j consist of the broadcast messages $m_{i,r}$ for $i = 1, \ldots, n$ and $r = 1, \ldots, R$. Send $(\mathsf{check}, g_j, M_j)$ to all parties.

5. Wait until all $n-1$ other parties P_i send $(\mathsf{check}, g_i, M_i)$. If all P_i sent $g_i =$ $\mathsf{preprocess\ success}$ and $M_i = M_j$, then send $s_j = x_j + r_j$ to all parties.
6. Wait to receive s_i from all other parties, let $S_j = (s_1, \ldots, s_n)$ and send S_j to all parties.
7. If all $n-1$ other parties P_i sent some S_i before the timeout and all $S_i = S_j$, then let $q_i = \mathsf{success}$. Otherwise, let $q_i = \mathsf{failure}$.
8. Run a Byzantine agreement (BA) on the q_i to agree on a common value $q \in \{\mathsf{failure}, \mathsf{success}\}$. Being a BA this protocol ensures that if $q_i = \mathsf{success}$ for all honest parties, then $q = \mathsf{success}$, and if $q_i = \mathsf{failure}$ for all honest parties, then $q = \mathsf{failure}$.

We assume that the preprocessing phase is started enough in advance of the time-out to guarantee that it will terminate successfully on time when there is no cheating. However, as mentioned in the introduction, the adversary can stop the preprocessing, in particular if a corrupted party does not send a message the preprocessing dead-locks.

Note that if just one honest party outputs $q_i = \mathsf{success}$, then the preprocessing protocol terminated successfully before the timeout and all the values s_i were consistently distributed. In particular, if $q = \mathsf{success}$, then $q_i = \mathsf{success}$ for at least one honest P_i, and therefore the preprocessing and inputting were successful.

As for security, if after each communication round in Step 3 the parties compared the messages $m_{i,r}$ and terminated if there was disagreement, then it is clear that a secure synchronous protocol[1] run asynchronously this way is again secure. The only loss is that the adversary can now deprive some parties of their input. The reason why it is secure to postpone the check of consistency of the broadcasted message until Step 5 is that the inputs x_i do not enter the computation until Step 6 and that there are no other secrets to be leaked, like secret keys. Sending inconsistent broadcast messages before Step 6 will therefore yield no information leakage. After Step 5 it is known that the preprocessing was an emulation of a consistent synchronous execution, at which point it becomes secure to use the result r_i to mask x_i.

This way to emulate a synchronous protocol in an asynchronous environment is generic and does not just apply to our protocols here.

Computation phase. If $q = \mathsf{failure}$, then all parties output $y_i = \perp$. If $q = \mathsf{success}$, then the parties compute $[x_i] = s_i - [r_i]$ for all P_i and run the asynchronous protocol described below which compute sharings $[y_i]$ of the outputs from the sharings $[x_i]$, making use of the multiplication triples from the preprocessing. Finally the shares of $[y_i]$ are sent privately to P_i which computes y_i.

We may assume that for each multiplication we have to do, a triple $[a], [b], [c]$ as described above is constructed in the preprocessing. To handle any arithmetic circuit describing the desired function, we then only need to describe how to deal with linear combinations and multiplications of shared values.

Linear Combinations: Shamir sharing is linear, and any linear function of shared values can therefore be computed locally by applying the same linear function to the shares.

[1] The synchronous security should be against a rushing adversary.

Multiplication: Consider a multiplication gate in the circuit and let $[a], [b], [c]$ be the triple constructed for this gate. Assume we have computed sharings of the two input values $[x]$ and $[y]$, so we now wish to compute $[xy]$. Note that

$$\begin{aligned} xy &= ((x-a)+a)((y-b)+b) \\ &= de + db + ae + ab, \end{aligned}$$

where $d = x - a$ and $e = y - b$. We may now publicly reconstruct d and e, since they are just random values in $\mathbb{F}$. The product can then be computed locally as

$$[xy] = de + d[b] + [a]e + [c] \ .$$

The overall cost of this multiplication is the cost of two public reconstructions and a constant number of local arithmetic operations.

A crucial observation is that this protocol (assuming the triples are given) can be executed in a completely asynchronous fashion, and is guaranteed to terminate: At each multiplication gate, each player simply waits until he has received enough shares of d and e and then reconstructs them. More precisely, we need that at least $n - t$ shares of each value have arrived, and that at least $n - t$ of them are consistent with some polynomial. Since there are $n - t$ honest players, $n - t$ consistent shares will eventually arrive. Moreover, if $n - t$ shares are found to be consistent, since $t < n/3$, these must include at least $t + 1$ shares from honest players, and so the correct value is always reconstructed. One can test if the conditions are satisfied using standard error correction techniques.

3.2 Security Model

The security of our protocol can be phrased in the UC framework [6]. For the protocol we assume the standard asynchronous communication model of the UC model, except that we let the timeout of P_i be called by the adversary by inputting timeout to that party, and that we assume secure point-to-point channels where the adversary can decide when a message sent is delivered. Our protocols are secure and terminate no matter when the timeouts are called. They provide outputs, $\neq \perp$, if all parties behave honestly in the preprocessing and the timeouts are called after the preprocessing succeeded at all honest parties. We formalize that by implementing an ideal functionality.

For a function $f \colon \mathbb{F}^n \to \mathbb{F}^n$, let $\mathcal{F}_{\text{FSFE}}^{f}$ be the following ideal functionality for fair secure function evaluation.

1. On input begin preprocess from P_i, output $(P_i, \text{begin preprocess})$ to the adversary.
2. On input x_i from P_i, output $(P_i, \text{gave input})$ to the adversary.
3. If the adversary inputs early timeout, then output $y_i = \perp$ to all P_i, and terminate.

4. If all P_i have input both $\mathsf{begin\ preprocess}$ and x_i and the adversary then inputs $\mathsf{late\ timeout}$, then compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and output y_i to all P_i, and terminate.

Note that the adversary can always make the evaluation fail, but must do so in a fair way: Either no party learns anything, or all parties learn a correct output. Our protocol securely implements this ideal functionality when $t < n/3$ parties are corrupted. If the BA is modeled as an ideal functionality, then our implementation is perfectly secure. We will not give the full simulation proofs below, as they follow more or less straightforwardly using known techniques.

On a high level, however, the simulation proceeds as follows: First the simulator simulates the first 4 steps while emulating the algorithms of honest players as specified in the protocol. This is possible as the secret inputs of honest players are not used in these steps. We write $\bar{P}_i$ for the simulator's "copy" of honest P_i.

If some honest $\bar{P}_j$ computed $g_j = \mathsf{preprocess\ failure}$, then the simulator inputs $\mathsf{early\ timeout}$ to $\mathcal{F}^f_{\text{FSFE}}$, which will make it output $y_i = \perp$ to all players. Clearly the same happens in the real execution since P_j sends $g_j = \mathsf{preprocess\ failure}$ to all honest parties.

If all honest $\bar{P}_j$ compute $g_j = \mathsf{preprocess\ success}$, then the preprocessing was secure. This ensures that the sharings $[r_i]$ are consistent, and since the simulator knows the shares of all $\bar{P}_j$, it can compute all r_i. From the s_i broadcast by the corrupted parties in the simulation it computes $x_i = s_i - r_i$ and inputs these to $\mathcal{F}^f_{\text{FSFE}}$ on behalf of the corrupted parties. It broadcasts random s_i's on behalf of honest players.

Then the simulator finishes the execution of the preprocess and input phase. If during this the adversary cheats or calls the timeouts at a time which makes the BA terminate with $q = \mathsf{failure}$, then the simulator inputs $\mathsf{early\ timeout}$ to $\mathcal{F}^f_{\text{FSFE}}$, which will make it output $y_i = \perp$ to all P_i. Clearly the same happens in the real execution.

If $q = \mathsf{success}$ in the simulation, then the simulator inputs $\mathsf{late\ timeout}$ to $\mathcal{F}^f_{\text{FSFE}}$, and learns the output for corrupted parties. It can now simulate the computation phase using standard techniques until all parties have computed their outputs[2]. Namely, since the computation phase is a sequence of public reconstructions, the simulator for each reconstruction selects the value to be opened, either a random value or a result y_i, as appropriate. It then computes shares to send on behalf of the honest players such that they are consistent with the opened value and the shares held by corrupted players.

4 Protocol for Preprocessing

In this section, we describe the techniques used in the preprocessing phase. One version of the preprocessing is obtained by simplifying in a straightforward way

[2] In this process, the simulator may need to control the time at which results are delivered to honest parties, depending on when the adversary chooses to deliver the messages in the simulated execution.

the protocols from Hirt and Beerliová-Trubíniová in [2], where *hyperinvertible* matrices are used to generate multiplication triples. Another version is based on pseudorandom secret-sharing [8].

4.1 Preprocessing Based on Hyperinvertible Matrices

In this subsection we will show how the preprocessing and input phase works. This amounts to showing how to generate the multiplication triples.

The key element in the way we generate triples is that while in [2], a player elimination step is run whenever a fault occurs, we accept the possibility that our protocol will not terminate. Therefore we can simplify and speed up the protocols considerably by cutting away the player elimination and simply aborting if a fault is detected. For completeness and readability, we will describe the most important protocols here, but refer to [2] for security proofs and some of the tools.

In order for us to be able to generate multiplication triples, we first need to be able to generate double sharings of random element – that is, two Shamir sharings of the same random element, possibly with different thresholds. In other words we wish to generate for a random $r \in \mathbb{F}$ sharings $[r]_d$ and $[r]_{d'}$, where d and d' are the degrees or thresholds. A more compact notation for the double sharing is $[r]_{d,d'}$.

We will need some facts from [2] on reconstructing shared values, namely how to reconstruct a value robustly to one player using $\mathcal{O}(nk)$ bits of communication and how to reconstruct up to $T = n - 2t$ values publicly using $\mathcal{O}(n^2 k)$ bits, where k is the size of a field element.

The following is based on the concept of hyperinvertible matrices. "Hyperinvertible" is defined as in [2], where a straightforward way to construct such a matrix is also presented:

Definition 1. *An $m \times n$ matrix M is* hyperinvertible *if for any selection $R \subseteq \{1, \ldots, m\}$ of rows and $C \subseteq \{1, \ldots, n\}$ of columns such that $|R| = |C| > 0$, the square matrix M_C^R consisting of the intersections between rows in R and columns in C is invertible.*

The protocol for generating $T = n - 2t$ double sharings now works as follows (it assumes the existence of an publicly known $n \times n$ hyperinvertible matrix M):

1. Each player P_i Shamir shares a random value s_i to the others using both d and d' as degrees. Every P_i now knows shares of $[s_1]_{d,d'}, \ldots, [s_n]_{d,d'}$, but shares from corrupted players may be incorrect.
2. The players locally compute

$$([r_1]_{d,d'}, \ldots, [r_n]_{d,d'}) = M([s_1]_{d,d'}, \ldots, [s_n]_{d,d'}) \ .$$

   Note that there are actually two vectors here, and the matrix is applied to both, creating two new vectors.
3. All sharings $[s_i]_{d,d'}$ are verified for $i = T + 1, \ldots, n$. They are verified by having each P_j send his share of $[s_i]_{d,d'}$ to P_i. Each P_i that is given shares must then check whether they are consistent and that both parts of the

double sharing represent the same value. If not, P_i sets an *unhappy* flag to indicate the fault.

4. The double sharings $[r_1]_{d,d'}, \ldots, [r_T]_{d,d'}$ are the output.

The double sharing protocol is guaranteed to either output $T = n - 2t$ correct and random double sharings that are unknown to the adversary or make at least one honest player unhappy. This is proved in [2], along with the fact that the communication complexity is $\mathcal{O}(n^2 k)$ bits. In our case, if an honest player becomes unhappy at any point, all other players are informed and the honest players will abort, as described in the Section 3. That is, we skip the player elimination used in [2].

If we only wanted to generate a set of T single Shamir sharings, it is easy to see that we can use the protocol above but considering only sharings using degree d for each step. The complexity of this is half that of creating double sharings. This is used for generating the sharings $[r_i]$ of a random r_i for each player P_i, that we promised in the Section 3.

Generating Multiplication Triples. Given sharings

$$[a_1]_t, \ldots, [a_T]_t, [b_1]_t, \ldots, [b_T]_t$$

and

$$[r_1]_{t,2t}, \ldots, [r_T]_{t,2t}$$

of random and independent numbers $a_i, b_i, r_i \in \mathbb{F}$, we can generate T multiplication triples as follows:

1. The players compute $[a_i]_t[b_i]_t - [r_i]_{2t} = [a_i b_i - r_i]_{2t}$ for $i = 1, \ldots, T$.[3] They then attempt to publicly reconstruct all of the $a_i b_i - r_i$. If the reconstruction of any of the values fails, an honest player becomes unhappy and we abort.
2. The players locally compute $[a_i b_i]_t = a_i b_i - r_i + [r_i]_t$. All honest players now own shares of the $[a_i b_i]_t$, the $[a_i]_t$ and the $[b_i]_t$ for $i = 1, \ldots, T$.

This protocol is clearly secure, assuming that the sharings we start from have been securely constructed. The simulator would choose random values s_i to play the role of $a_i b_i - r_i$, it would then expand the set of shares known by corrupt players of $[a_i b_i - r_i]$ to a complete set consistent with s_i and use these shares as those sent by honest players. Please see [2] for more details.

The communication complexity is $\mathcal{O}(n^2 k)$ bits for the reconstructions and therefore a total of $\mathcal{O}(n^2 k)$ bits including the generation of the double sharings. That is, we can reconstruct $T = n - 2t = \Theta(n)$ shares with a communication complexity of $\mathcal{O}(n^2 k)$, where k is the bit length of the field elements.

4.2 Preprocessing Based on Pseudorandom Secret-Sharing

We show here how to do the preprocessing based on pseudorandom secret-sharing. The techniques used are described in detail in [9], but we present here an overview for completeness.

[3] The notation $[a_i]_t[b_i]_t$ means that each player locally multiplies its shares $[a_i]_t$ and $[b_i]_t$. This gives a $2t$ sharing of $a_i b_i$.

Pseudorandom Secret-Sharing. Let A be a set of players of size $n-t$. We can create a random, shared secret by defining for each set A a random value r_A and give it to all players in A. The secret is then given by

$$s = \sum_A r_A \ .$$

Now every maximal unqualified set $\{1, \ldots, n\} \setminus A$ misses exactly one value, namely r_A.

Keeping the above in mind, pseudorandom secret-sharing (PRSS) is then based on the observation that we can create many random shared secrets by distributing once and for all one set of r_A values.

The trick is to use a pseudorandom function ψ_{r_A} with r_A as its key. If the parties agree on some publicly known value a, they can generate the random values they need as $\psi_{r_A}(a)$. So the secret is now

$$s = \sum_A \psi_{r_A}(a) \ .$$

What we actually want, however, is a Shamir sharing. This can be fixed as follows. Define a degree at most t polynomial f_A by $f_A(0) = 1$ and $f_A(i) = 0 \quad \forall i \in \{1, \ldots, n\} \setminus A$. Now each player P_i computes its share

$$s_i = \sum_{\substack{A \subset \{1,\ldots,n\}: \\ |A|=n-t, i \in A}} \psi_{r_A}(a) f_A(i) \ .$$

This is in fact a Shamir sharing of s, since it defines the polynomial

$$f(\mathtt{x}) = \sum_{\substack{A \subset \{1,\ldots,n\}: \\ |A|=n-t}} \psi_{r_A}(a) f_A(\mathtt{x}) \ .$$

It is easy to see that this polynomial has degree at most t and that

$$f(0) = \sum_{\substack{A \subset \{1,\ldots,n\}: \\ |A|=n-t}} \psi_{r_A}(a) = s \ ,$$

which means that it shares the right secret. It is also clear that $s_i = f(i)$, which means that our sharing is a correct Shamir sharing.

Pseudorandom Zero-Sharing. We will need one more tool to be able to generate multiplication triples, namely what is defined in [9] as pseudorandom zero-sharing (PRZS).

Like PRSS, it creates a Shamir sharing using only local computations, but in this case it is a sharing of 0. We will need a sharing of degree $2t$ in the following, but the approach works just as well with other thresholds. First, for a set $A \subseteq \{1, \ldots, n\}$ of size $n-t$ we define the set

$$G_A = \{g \in \mathbb{Z}_p[x] \mid \deg(g) \leq 2t \wedge g(0) = 0 \wedge (j \notin A \Rightarrow g(j) = 0)\} \ .$$

This is a subspace of the vector space of polynomials of degree at most $2t$. Because every polynomial in the set has $t+1$ zeros, the subspace must have dimension $2t+1-(t+1)=t$. The construction from [9] needs a basis for this subspace, but no explicit construction was given there. We suggest to use the following:

$$(g_A^1, \dots, g_A^i, \dots, g_A^t) = (xf_A, \dots, x^i f_A, \dots, x^t f_A),$$

where the f_A is defined as above. It is a basis because it has t elements of G_A which are all of different degrees and therefore linearly independent. Exactly as for PRSS, we assume that we have values r_A known (only) by players in A. Now we define the share at player j as

$$s_j = \sum_{\substack{A \subset \{1,\dots,n\}: \\ |A|=n-t, j \in A}} \sum_{i=1}^{t} \psi_{r_A}(a,i) g_A^i(j) \ .$$

Note here that the inner sum is a pseudorandom choice of a polynomial from G_A, evaluated in the point j. Now we want to show that this leads to a Shamir sharing of 0, so we define the corresponding polynomial as

$$g_0(\mathtt{x}) = \sum_{\substack{A \subset \{1,\dots,n\}: \\ |A|=n-t}} \sum_{i=1}^{t} \psi_{r_A}(a,i) g_A^i(\mathtt{x}).$$

The degree of g_0 is clearly at most $2t$, and it is also easy to see that it is consistent with the shares above and that $g_0(0)=0$.

Making triples using PRSS and PRZS. In order to make multiplication triples, we already know that it is enough if we can build random sharings $[a]_t$, $[b]_t$, and a double sharing $[r]_{t,2t}$.

Using PRSS, it is easy to construct the random degree t sharings. A double sharing can be constructed as follows: Create using PRSS a random sharing $[r]_t$ and use PRZS to create a sharing of zero $[0]_{2t}$. Now

$$[r]_{2t} = [r]_t + [0]_{2t}$$

is clearly a sharing of r of degree $2t$. We can therefore use pseudorandom secret sharing and pseudorandom zero sharing to locally compute all the values needed to make multiplication triples. The only interaction needed is one public opening for each triple as described in Section 4.1.

This is faster than using hyperinvertible matrices for a small number of players, but does not scale well: Since $n-t=\Theta(n)$, the local computation is exponential in n, as clearly seen from the benchmark results in Section 6. The break-even point between PRSS and hyperinvertible matrices depends both on local computing power and on the cost of communication.

Security of the PRSS approach. We claim that the overall protocol is secure against a computationally bounded and static adversary, when based on PRSS.

To argue this, consider some adversary who corrupts t players, and let A be the set of $n-t$ honest players. Now let π_{random} be the protocol that runs as described above, but where the function ψ_{r_A} is replaced with a truly random function.[4]

When we execute PRSS or PRZS in π_{random}, all secrets and sets of shares held by the honest players are uniformly random, with the only restriction that they are consistent with the shares held by corrupt players. We can therefore use the proof outlined in Section 3.2 to show that π_{random} implements $\mathcal{F}^{f}_{\text{FSFE}}$ (with perfect security).

For the rest of the argument, we refer to the protocol using the pseudorandom function as π_{pseudo}. We claim that real-world executions of π_{random} and π_{pseudo} are computationally indistinguishable. Assume for contradiction that there exists some computationally bounded environment $\mathcal{Z}$ that can distinguish between the two with a non-negligible advantage.

From $\mathcal{Z}$ we can now build a new machine $\mathcal{M}$, which gets oracle access to some function f and outputs its guess of whether the function is pseudorandom or truly random.

$\mathcal{M}$ simply runs the protocol with f inserted in the place of ψ_{r_A} (i.e., it runs either π_{random} or π_{pseudo}) for $\mathcal{Z}$. If $\mathcal{Z}$ outputs "π_{random}", M outputs "truly random", otherwise it outputs "pseudorandom". Clearly, M can distinguish between a pseudorandom function and a truly random function with a non-negligible advantage, breaking the security of our PRF.

Combining this with the fact that π_{random} securely realizes $\mathcal{F}$, we see that the same holds for π_{pseudo} (with computational security): The simulator that works for π_{random} also works for π_{pseudo}.

5 VIFF

The Virtual Ideal Functionality Framework, VIFF, is a new software library with building blocks for developing cryptographic protocols. The goal of the library is to provide an efficient and high-level basis on which practical applications using MPC can be built. It is also our hope that the framework offered by VIFF will help facilitate rapid prototyping of new protocols by researchers and so lead to more protocols with practical applications. The source code and documentation is therefore freely available from the VIFF homepage: `http://viff.dk/`.

VIFF aims to be usable by parties connected by real world networks. Such networks are all asynchronous by nature, which means that no upper bound can be given on the message delivery time. A well-known example is the Internet where the communication from A to B must go through many hops, each of which introduces an unpredictable delay. Targeting networks with this kind of worst-case behavior from the beginning means that VIFF works well in all environments, including local area networks which typically behave in a much more synchronous manner.

[4] This can be formalized by assuming an ideal functionality that gives oracle access to the function for the honest players as soon as the adversary has corrupted a set of players initially.

To be efficient in the asynchronous setting, the VIFF runtime system tries to avoid waiting unless it is explicitly asked to do so. In a synchronous setting all parties wait for each other at the end of each round, but VIFF has no notion of "rounds". What determines the order of execution is solely the inherent dependencies in a given program. If two parts of a program have no mutual dependencies, then their relative ordering in the execution is unpredictable. This assumes that the calculations remain secure when executed out-of-order. Protocols written for asynchronous networks naturally enjoy this property since the adversary can delay packets arbitrarily, which makes the reordering done by VIFF a special case.

As an example, consider the simple program in Figure 1 for three players, $n = 3$. It starts by defining the field $\mathbb{Z}_{1031}$ where the toy-calculation will take place and a list with the IDs of all players. The user is then prompted for input (an integer). All three players then take part in a Shamir sharing of their respective inputs, this results in three Share objects being defined. A fourth Share object is generated using pseudorandom secret sharing [8].

```
# (Standard program setup not shown.)

Zp = GF(1031)
all = [1, 2, 3]

input = int(raw_input("Your input: "))
a, b, c = rt.shamir_share(all, Zp, input)
d = rt.prss_share_random(Zp)

x = a * b
y = c * d
z = x + y
```

Fig. 1. Program, rt is a Runtime object

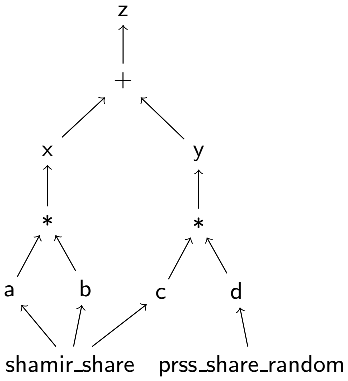


Fig. 2. Expression tree

Here all variables represent secret-shared values – VIFF supports Shamir secret sharing for when $n \geq 3$ and additive secret shares for when $n = 2$. The execution of the above calculation is best understood as the evaluation of a tree, please see Figure 2. Arrows denote dependencies between the expressions that result in the calculation of the variable z.

The two variables x and y are mutually independent, and so one cannot reliably say which will be calculated first. But more importantly: We may calculate x and y in *parallel*. It is in fact very important for efficiency reasons that we calculate x and y in parallel. The execution time of a multiparty computation is limited by the speed of the CPUs engaged in the local computations and by the delays in the network. Network latencies can reach several hundred milliseconds, and will typically dominate the running time.

When we say *parallel* we mean that when the calculation of x is blocked and waits on network communication from other parties, then it is important that the calculation of y gets a chance to begin its network communication. This puts maximum load on both the CPU and the network.

5.1 Implementing VIFF

VIFF is written in Python, a modern object-oriented procedural language. Programs using VIFF are normal Python programs, and yet we have described how operations may be executed in a random order. This is possible by using a technique where we only work with *deferred* results and never with the results themselves. A deferred result is something that will eventually materialize, but we do not know when and the only available operation is to add *callbacks* to the result. Callbacks are simply function pointers, and each deferred result keeps a list of such pointers. The callbacks will be executed when the result is ready, typically when some share has arrived over the network. This programming style is well-known from graphical user interfaces where the programmer also attaches callbacks to certain events in the user interface. In VIFF this is implemented using the Twisted network library, specifically using the Deferred class provided by Twisted. An example of how Twisted works is this program which retrieves a page from the Internet and prints out the contents:

```
def print_contents(contents):
    print "The Deferred has called us with:"
    print contents

deferred = getPage('http://example.net/')
deferred.addCallback(print_contents)
```

The getPage function returns a Deferred which will eventually hold the HTML of the fetched page. When it does, it will call its callbacks in sequence. If we had retrieved several pages, and attached different callbacks to each, then the execution of those callbacks would depend on which page arrives first.

VIFF uses the Deferred class extensively. In Figure 2 the variables are Share objects, a sub-class of Deferred. Using suitable operator overloading we are able to allow the programmer to do arithmetic with Share objects and so treat them like normal numbers. Key to the implementation of VIFF is a function gather_shares which takes a list of Share objects as input and returns a new Share. This Share will call its callbacks when all Share objects in the list have called their callbacks. We use this to make Share objects that *wait* on other Share objects. Figure 4 in Appendix A shows the implementation of the standard passively secure BGW multiplication protocol [4] in VIFF, and uses gather_shares to make the product wait on the two operands to the multiplication.

The big advantage of this system is that it automatically runs the operations in parallel: The calculations implicitly create the tree shown in Figure 2, and this tree is destroyed as fast as possible when operands become ready. There is no predefined execution order in the tree – it is deconstructed from the leaves inwards at the same speed as the needed operands arrive.

Please note that this simple system for parallel scheduling automatically extends to all levels in the program, i.e., from primitives like addition and multiplication up to compound protocols like comparisons or even entire auctions. This is a very important consequence since it allows us to build a larger protocol π by combining smalller protocols and still be sure that several instances of π can be executed in parallel. It is the uniform interface enforced by always working on deferred values which enables this kind of modularity. At runtime, a new protocol π will simply correspond to a subtree in Figure 2 and its leaf nodes will be executed in parallel with all other leaf nodes in the tree.

Executing things in this way changes the semantics of a program using VIFF from that of a normal Python program. Each statement is no longer executed when it is encountered, it is merely scheduled for execution and then executed later when the operands are available. The semantics of a program using VIFF is thus more like that of a declarative programming language where you declare your intentions but where the compiler takes care of scheduling the calculations in the optimal order.

6 Benchmark Results

In order to measure the efficiency of our implementation, we have run a number of tests using the techniques described above on a set of computers on a fast local area network. The computers had Intel Celeron CPUs with a clock speed of 2.40 GHz, 1 GiB of RAM and were running Red Hat Enterprise Linux 5.2, Python 2.4.3, and VIFF 0.7.

We ran benchmarks with $n = 4, 7, \ldots, 25$ corresponding to thresholds $t = 1, 2, \ldots, 8$, respectively. In each test we secret-shared 2,000 random 32-bit numbers and multiplied the 1,000 pairs in parallel. The results in Table 1 is the average online time spent per multiplication (columns 2, 3, and 5) and the average offline time spent per multiplication triple (columns 4 and 6).

Table 1 also includes a column giving the ratio between the online time for the multiplication protocol described here using multiplication triples, and the time for the standard BGW multiplication protocol which is only secure against passive adversaries [4]. The passively secure multiplication protocol consists of

Table 1. Benchmark results

(n, t)	Passive	Active PRSS		Active Hyper		Ratio
$(4, 1)$	2 ms	4 ms	5 ms	4 ms	20 ms	2.6
$(7, 2)$	3 ms	6 ms	22 ms	6 ms	42 ms	2.2
$(10, 3)$	4 ms	8 ms	130 ms	8 ms	82 ms	2.0
$(13, 4)$	6 ms	10 ms	893 ms	10 ms	136 ms	1.7
$(16, 5)$	8 ms	—	—	12 ms	208 ms	1.6
$(19, 6)$	10 ms	—	—	14 ms	287 ms	1.5
$(22, 7)$	12 ms	—	—	17 ms	377 ms	1.3
$(25, 8)$	15 ms	—	—	19 ms	501 ms	1.3

a local multiplication followed by a resharing in which everybody communicates with everybody else. The actively secure multiplication, as described above, consists of local multiplications and two openings, which also involves quadratic communication.

The average online time per multiplication appears to grow linearly in the number of players, both in the case of passive and active adversaries. The total amount of network traffic is quadratic in the number of players (in both protocols), but the work done by each player grows only linearly. Our results therefore suggest that the players are CPU bound instead of being slowed down by the network. In the test setup all 25 machines were located on a fast LAN with ping times of about 0.1 ms, so this is to be expected. We hope to setup a better test environment with a controllable network delay in order to do more realistic testing in the future.

One would expect that the average preprocessing time per multiplication triple (produced via hyperinvertible matrices) would also grow linearly. Instead it seems to grow quadratically (see Figure 3). The curve $f(n) = an^2 + bn + c$ with $a = 0.8$, $b = -1$, $c = 10$ is the best fit. We do not have a good explanation for this at the moment, but hope to find out based on a more controllable test environment.

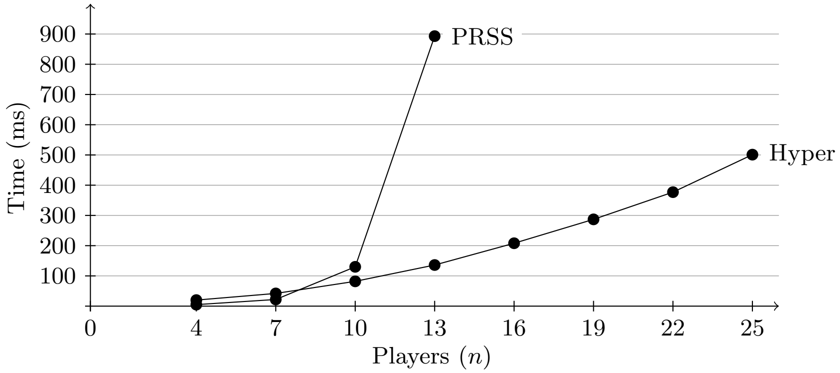


Fig. 3. Average preprocessing time per multiplication triple as a function of the number of players

As expected, the PRSS based preprocessing is faster for a small number of players but does not scale well, and we had to abandon it for $n > 13$. The amount of work per player depends on the number of subsets of size $n-t$ and with $\binom{n}{n-t}$ subsets this gives an exponential growth.

7 Conclusion

We have presented an efficient protocol for general multiparty computation secure against active and adaptive adversaries. The protocol provides a new tradeoff between guaranteeing termination and efficiency which we believe is relevant in practice. To demonstrate this we have implemented the protocol in a framework for secure multiparty computation called VIFF. This allowed us to show that achieving active security costs only about a factor of two in online time, if one is willing to accept that the preprocessing step might fail without revealing any private data. We believe this to be well-suited for practical applications where the parties typically have a much stronger incentive to participate in the computation than to halt it.

Even though the cost of preprocessing is larger than the online cost, it is certainly not prohibitive: For instance, for 4 players, 1000 multiplications can be prepared in 5 seconds.

Currently VIFF supports the fast arithmetic using standard Shamir shares for the case with three or more players, and has support for much slower arithmetic with additive shares in the two player case. Using the additively homomorphic Paillier public key cryptosystem [13], our benchmarks show an average time per multiplication of 300 ms for 32-bit numbers.[5] This is with a straightforward implementation of the cryptosystem in Python and we expect to gain some performance by reimplementing it as a C extension instead.

In the two player case we have $t = n - 1$, also known as *self trust* since every player only need to trust himself to be honest. We would like to develop protocols for $t = n - 1$, but for $n > 2$. Such a high threshold will again require public key cryptography, so we expect this to be expensive, but nevertheless interesting since there might be some situations where the parties are willing to wait longer in return for this level of security.

VIFF can be freely downloaded from `http://viff.dk/` and it is hoped that others can verify our measurements and expand them it with other protocols.

References

1. Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. ACM (1988)
2. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008)
3. Beerliová-Trubíniová, Z., Hirt, M., Nielsen, J.B.: Almost-asynchronous multi-party computation with faulty minority (manuscript, 2008)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC [1], pp. 1–10
5. Bogetoft, P., Christensen, D.L., Damgard, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Multiparty computation goes live. Cryptology ePrint Archive, Report 2008/068 (2008), `http://eprint.iacr.org/`
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE, Los Alamitos (2001)

[5] The implementation actually allows multiplication of much larger numbers, up to about 500 bits with a marginal performance penalty.

7. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC [1], pp. 11–19 (1988)
8. Cramer, R., Damgård, I.B., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 342–362. Springer, Heidelberg (2005)
9. Cramer, R., Damgård, I.B., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 342–362. Springer, Heidelberg (2005)
10. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game – a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229. ACM, New York (1987)
11. Hirt, M., Maurer, U.M.: Robustness for free in unconditional multi-party computation. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 101–118. Springer, Heidelberg (2001)
12. Hirt, M., Nielsen, J.B., Przydatek, B.: Asynchronous multi-party computation with quadratic communication. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 473–485. Springer, Heidelberg (2008)
13. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
14. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)

A Multiplication in VIFF

As an example of real VIFF code, we have included the implementation of the standard BGW multiplication protocol [4] which is secure against passive adversaries, see Figure 4.

The code handles both local multiplication and multiplication involving network traffic. First, if either share_a or share_b is a not a Share object, i.e., one of them is a constant integer or a FieldElement, then we do a quick local multiplication. Assume that share_a is the constant and share_b is the Share (lines 5–10). We cannot simply multiply share_a and share_b since share_b is a Deferred and might not have a value yet. The solution is to clone share_b and add a callback to it. This callback is simply a lambda expression (an anonymous function) that takes care of the correct multiplication when share_b eventually gets a value (line 9). The opposite case is handled in the same way (lines 11–15). If it is established that both share_a and share_b are Share objects we create a new Share which waits on both of them (line 19). We then add several callbacks: First we multiply, then we reshare, and finally we recombine. These three operations will be executed in sequence when both share_a and share_b have received their values due to incoming network traffic. The last two callbacks involve network traffic, and must be added using a more expensive mechanism to ensure agreement on the labels put on the data as it is sent over the network.

```
   def mul(self, share_a, share_b):
 2     assert isinstance(share_a, Share) or isinstance(share_b, Share), \
           "Either share_a or share_b must be a Share."
 4
       if not isinstance(share_a, Share):
 6         # Then share_b must be a Share => local multiplication. We
           # clone first to avoid changing share_b.
 8         result = share_b.clone()
           result.addCallback(lambda b: share_a * b)
10         return result
       if not isinstance(share_b, Share):
12         # Likewise when share_b is a constant.
           result = share_a.clone()
14         result.addCallback(lambda a: a * share_b)
           return result
16
       # At this point both share_a and share_b must be Share objects. We
18     # wait on them, multiply, reshare, and recombine.
       result = gather_shares([share_a, share_b])
20     result.addCallback(lambda (a, b): a * b)
       self.schedule_callback(result, self._shamir_share)
22     self.schedule_callback(result, self._recombine, threshold=2*self.threshold)
       return result
```

Fig. 4. The standard multiplication protocol for passive adversaries

In all three cases the mul method returns result to the caller (lines 10, 15, or 23). Note that result probably does not have a value at this point, but result is a Share that we have prepared in such a way that it *will* receive the correct value at some point in the future. All VIFF methods follow this pattern.

Multi-Party Computation with Omnipresent Adversary

Hossein Ghodosi[1] and Josef Pieprzyk[2]

[1] School of Mathematics, Physics and Information Technology
James Cook University, Townsville, Qld 4811, Australia
[2] Department of Computing
Center for Advanced Computing – Algorithms and Cryptography
Macquarie University, Sydney, NSW 2109 Australia

Abstract. Secure multi-party computation (MPC) protocols enable a set of n mutually distrusting participants $P_1, \ldots, P_n$, each with their own private input x_i, to compute a function $Y = F(x_1, \ldots, x_n)$, such that at the end of the protocol, all participants learn the correct value of Y, while secrecy of the private inputs is maintained. Classical results in the unconditionally secure MPC indicate that in the presence of an active adversary, every function can be computed if and only if the number of corrupted participants, t_a, is smaller than $n/3$. Relaxing the requirement of perfect secrecy and utilizing broadcast channels, one can improve this bound to $t_a < n/2$.

All existing MPC protocols assume that uncorrupted participants are truly honest, i.e., they are not even curious in learning other participant secret inputs. Based on this assumption, some MPC protocols are designed in such a way that after elimination of all misbehaving participants, the remaining ones learn all information in the system. This is not consistent with maintaining privacy of the participant inputs. Furthermore, an improvement of the classical results given by Fitzi, Hirt, and Maurer indicates that in addition to t_a actively corrupted participants, the adversary may simultaneously corrupt some participants passively. This is in contrast to the assumption that participants who are not corrupted by an active adversary are truly honest.

This paper examines the privacy of MPC protocols, and introduces the notion of an *omnipresent adversary*, which cannot be eliminated from the protocol. The omnipresent adversary can be either a passive, an active or a mixed one. We assume that up to a minority of participants who are not corrupted by an active adversary can be corrupted passively, with the restriction that at any time, the number of corrupted participants does not exceed a predetermined threshold. We will also show that the existence of a *t-resilient* protocol for a group of n participants, implies the existence of a *t'-private* protocol for a group of n' participants. That is, the elimination of misbehaving participants from a *t-resilient* protocol leads to the decomposition of the protocol.

Our adversary model stipulates that a MPC protocol never operates with a set of truly honest participants (which is a more realistic scenario). Therefore, privacy of all participants who properly follow the protocol will be maintained. We present a novel disqualification protocol to avoid a loss of privacy of participants who properly follow the protocol.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 180–195, 2009.

Keywords: Multi-Party Computation, Omnipresent Adversary, Proactive Secret Sharing, *t-resilient* Protocols, *t-private* Protocols.

1 Introduction

Multi-party computation (MPC) protocols provide a general model for secure computation of arbitrary function whose arguments (inputs) are held by a group of participants. The concept of MPC was introduced by Yao [18] for two-party computations and then generalized by Goldreich, Micali, and Wigderson [12] for an arbitrary number of participants. A secure MPC protocol enables a set of n mutually distrusting participants $P_1, \dots, P_n$, each with their own private input x_i, to compute a function $Y = F(x_1, \dots, x_n)$, such that at the end of the protocol, all participants learn the correct value of Y, while the confidentiality of the private inputs x_i is maintained. The design of secure MPC protocols has been the subject of investigations by many researchers, and many solutions have been published in the literature. From the security point of view, these protocols can be classified into two broad categories: (i) *computationally secure* MPC protocols, and (ii) *unconditionally secure* MPC protocols. For computationally (conditionally) secure MPC protocols, we assume that the adversary is polynomially bounded. More precisely, breaking the security of the protocol implies that the adversary is able to solve efficiently (in polynomial time) a problem that is believed to be intractable. Unconditionally secure MPC protocols are intrinsically secure, that is, no matter how much time and computing power is available to the adversary, they cannot break the system better than by guessing private inputs. In both computationally and unconditionally secure MPC protocols, the security model includes the adversary, who may corrupt some participants. Two types of adversaries, namely *passive* and *active* have been studied in the literature.

Passive Adversary. Participants who are corrupted by a passive adversary properly follow the protocol but try to learn private information of others. That is, a passive adversary has access to the information of corrupted participants, but will not control their behaviour. In other words, a passive adversary threatens the privacy of uncorrupted participants. Note however, that the correctness of the protocol is preserved. Corrupted participants are also called *honest-but-curious.*

A commonly used parameter to measure the level of security obtained in a multi-party protocol with n participants, is the maximum number of participants that can be corrupted by a passive adversary while the privacy of uncorrupted participants still holds. This parameter is determined by a threshold t ($t < n$). Protocols that can tolerate up to t corrupted participants, are called *t-private.*

Definition 1. *A multi-party protocol is t-private if after completion of the protocol, any subset of up to t participants cannot learn more information (about honest participant private inputs) than what they could derive from their private inputs and the output of the protocol.*

Active Adversary. A more serious threat for the security of MPC protocols are corrupted participants who not only try to learn additional information but may also wish to disrupt the protocol. This type of participants are called *malicious* and they are said to be corrupted by an *active adversary*, who has access to all information of the corrupted participants, and controls their behaviour. Participants who are corrupted by an active adversary may behave arbitrarily, and may deviate from the protocol at any time.

The main challenge in designing secure MPC protocols in the presence of an active adversary, is to equip the protocol with mechanisms that can detect misbehaving participants and eliminate them from the protocol without influencing the correctness of the protocol. Such protocols are called *robust*. A measure for expressing the level of robustness is determined by a threshold parameter t ($t < n$), that is the maximum number of participants that can be corrupted, without an impact on the correctness of the protocol. A protocol that can tolerate up to t malicious participants is called *t-resilient* and is defined as follows.

Definition 2. *A multi-party protocol is t-resilient if no set of up to t malicious participants can influence the correctness of the output produced by the protocol.*

1.1 Preliminaries

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants who wish to compute a function $Y = F(x_1, \ldots, x_n)$, where P_i holds private input x_i. Without loss of generality, we will assume that all input variables are elements of a finite field E, and the function F can be computed by a circuit over E using the field operations $+$, $\times$, the inverse operations and constants from E.

The model of computation is a complete synchronous network of n participants. The broadcast and pairwise communication channels between participants are secure, that is, they cannot be read or tampered with by other participants.

MPC Protocols with a Passive Adversary

A generic MPC protocol consists of the following three phases [4].

1. **Initialization.** Each participant P_i ($i = 1, \ldots, n$) utilizes Shamir's secret sharing scheme [17], and distributes his private input x_i amongst all participants in a t-private manner. More precisely, P_i chooses a random polynomial $f_i(x) = x_i + a_1 x + \ldots, + a_t x^t$, and gives $s_{j,i} = f_i(j)$ to participant P_j for $j = 1, \ldots, n$.
2. **Computation.** Let $s_{i,k}$ and $s_{i,\ell}$ be P_i's shares, associated with polynomials $f_k(x)$ and $f_\ell(x)$ for the secrets x_k and x_ℓ, respectively. The computation of every linear function is straightforward. In order to compute $x_k + x_\ell$, each cooperating participant, P_i, computes $s_i^{k+\ell} = s_{i,k} + s_{i,\ell}$, which is the share of P_i determined by polynomial $h(x) = f_k(x) + f_\ell(x)$. Since $h(x)$ is a polynomial of degree (at most) t, a set of at least $t + 1$ participants who properly follow the protocol can reconstruct the polynomial, and thus retrieve the constant term of the polynomial $h(x)$, which is $x_k + x_\ell$. Similarly, $c \times x_k$, where c

is a known scalar, can be computed by $t+1$ participants (each participant P_i calculates $c \times s_{i,k}$ as its share of $c \times x_k$). That is, for $n > t$, there is a non-interactive protocol for computing every linear function $F(x_1, \ldots, x_n)$.

Computation of non-linear functions, however, is not so straightforward. Assume that we want to compute $x_k \times x_\ell$. Although P_i can compute $s_i^{k \times \ell} = s_{i,k} \times s_{i,\ell}$, where $s_i^{k \times \ell}$ is P_i's share of $x_k \times x_\ell$, there are two problems. The first problem is that $s_i^{k \times \ell}$ is the share of P_i determined by the polynomial $h(x) = f_k(x) \times f_\ell(x)$ whose degree is (at most) $2t$. The second problem is that $h(x)$ is not a random polynomial.

Assuming that $n > 2t$, the computation can be carried out if the participants collectively redistribute the constant term of the $h(x)$ polynomial, in a t-private manner, amongst themselves. This process, also called degree reduction, is necessary, otherwise further multiplications will raise the degree, and once the degree of polynomial is equal to or larger than the number of participants in the system, participants will not have a sufficient number of points to perform the necessary interpolation.

Computation of an additive inverse is straightforward. To compute the additive inverse of x_i, every participant P_j computes the additive inverse of his share, $s_{j,i}$. Computing a multiplicative inverse, however, implies cooperation of all shareholders. Catalano, Gennaro, and Halevi [6] have shown how to compute multiplicative inverses.

3. **Reconstruction of the function value.** The function $F(x_1, \ldots, x_n)$ can be represented as a polynomial containing sum of products and the participants can collectively evaluate first products (product gates) and then sums (sum gates) getting finally the shares of the function value Y. In order to reconstruct Y, a set of a sufficiently large set of participants can pool their shares and recover the value Y.

Thus, in the presence of a passive adversary, a set of n participants can compute every *n-variate* function, in a *t-private* manner, as long as $n > 2t$.

MPC Protocols with an Active Adversary

The main challenge in designing MPC protocols in the presence of an active adversary is how to deal with malicious participants. In general, robust MPC protocols first identify misbehaving participants and then disqualify them. Two different disqualification techniques are being used:

1. *Ignoring the information associated with malicious participants* – This strategy is used in MPC protocols that can be completed without utilizing the information coming fom malicious participants.
2. *Reconstructing the information associated with malicious participants* – This strategy is used in MPC protocols that cannot be completed without using the information owned by corrupted participants. So, after detection of misbehaving participant P_i, other participants reconstruct the private information of P_i, and re-share it amongst themselves.

1.2 Background

In 1987, Goldreich, Micali, and Wigderson [12] gave a solution to the general MPC problem assuming that one-way functions with trapdoor exist (i.e. their protocol is computationally secure). They have shown that in the presence of passive adversaries, every function can be computed by n participants, in such a way that no subset of less than n participants can learn any additional information apart from the function value. They have also shown that if Byzantine faults are allowed (i.e. an active adversary may corrupt some participants and control their behaviour), every function can be computed by n collaborating participants, as long as the majority of participants is honest.

In 1988, Ben-Or, Goldwasser, and Wigderson [4] and Chaum, Crépeau, and Damgård [7], independently studied unconditionally secure MPC protocols. They have shown that:

(a) In the presence of a passive adversary, no set of size $t < n/2$ of participants learns any additional information, other than the function value.
(b) If Byzantine faults are allowed, no set of size $t < n/3$ can learn any additional information or disrupt the protocol.

Relaxing the requirement of perfect security and assuming that broadcast channels exists, Rabin and Ben-Or [16] have shown that in the presence of Byzantine faults, MPC protocols exist if the majority of participants is honest. The privacy achieved is unconditional (with error probability $\epsilon > 0$, which can be exponentially small), and does not rely on any assumption about computational intractability. Beaver [2] utilized the verifiable secret sharing (VSS) scheme of [16] and achieved similar results.

The MPC protocols from [4,7] are determined for $n \geq 3t + 1$ participants, where up to t of them can be corrupted. The disqualification method used in these protocols simply ignores the misbehaving participants, since at any time, there exists at least $2t + 1$ honest participants who properly follow the protocol.

In contast, the protocols from [16,2] are defined for $n \geq 2t + 1$ participants. After each multiplication, the polynomial associated with the multiplied shares is of degree at most $2t$. If a malicious participant P_i does not cooperate properly, the remaining participants must reconstruct all information in the hands of P_i, otherwise they cannot interpolate the associated $2t$-degree polynomial. Reconstruction of information in the hands of a participant is possible since the share of each participant is re-shared via the second level of sharing. This procedure, however, reveals one share associated with the secret input of every participant[1]. In these protocols, after detection and elimination of any malicious participant the degree of threshold parameter is decreased by one. So after elimination of t malicious participants, the threshold parameter drops to zero (i.e., all private information is disclosed to the remaining participants). One may argue that this is not a security problem, since the remaining participants are assumed to be honest.

[1] Similar problem in shared generation of digital signatures has been considered in [1].

Fitzi, Hirt, and Maurer [10] improved the classical results in unconditionally secure MPC by considering a *mixed adversary.* They have shown that in addition to $t_a < n/3$ actively corrupted participants, privacy can be guaranteed against additionally $t_p \leq n/6$ passively corrupted participants. They have also introduced the concept of (t_a, t_p)-secure MPC protocols. In a (t_a, t_p)-secure MPC protocol, correctness of the protocol is guaranteed if up to t_a participants are corrupted actively, and privacy of the participants is ensured if (in addition to t_a actively corrupted participants) up to t_p participants are corrupted passively.

1.3 Motivation

All existing MPC protocols with active adversaries assume that uncorrupted participants are truly honest, i.e., they are not even curious in learning private inputs of others. Based on this assumption, some MPC protocols (e.g. [16] [2], [3], [11], [14], etc.) are designed in such a way that after elimination of all misbehaving participants, the remaining ones learn all private information.

Furthermore, an improvement of the classical results provided by Fitzi, Hirt, and Maurer indicates that in addition to t_a actively corrupted participants, the adversary may simultaneously corrupt some participants passively. This is in contrast to the assumption that participants who are not corrupted by an active adversary are truly honest.

The paper examines the privacy of MPC protocols and argues that the assumption about participants that are not corrupted are truly honest is unrealistic. We will introduce the notion of the *omnipresent adversary.* An omnipresent adversary cannot be eliminated from the protocol and can be either passive, active, or mixed. More precisely, we assume that that up to a minority of participants who are not corrupted by an active adversary can be corrupted passively, with the restriction that at any time, the number of corrupt participants does not exceed a predetermined threshold. We will also show that the existence of a *t-resilient* protocol for a group of n participants, implies the existence of a *t'-private* protocol for a group of n' participants. That is, elimination of misbehaving participants from a *t-resilient* protocol leads to the decomposition of the protocol, and converts it to the *t'-private* protocol.

Our adversary model stipulates that a MPC protocol never operates with a set of truly honest participants (which is a more realistic scenario). Therefore, the privacy of all participants who properly follow the protocol will be maintained. In order to achieve these goals in the existing MPC protocols, we present a novel disqualification protocol that avoids exposing the privacy of participants who properly follow the protocol.

Our results are as follows:

Theorem 1. *Given a set of $n = 2t + 1$ participants in the computationally secure setting, then there exists a t-resilient and t-private MPC protocol provided that, at every stage of the protocol, the total number of actively and passively corrupted participants is not larger than t. That is, only one participant may not be corrupted by adversary throughout the execution of the protocol.*

Theorem 2. *Given a set of $n = 3t+1$ participants in the unconditionally secure setting with perfect secrecy, then there exists a t-resilient and t-private MPC protocol. That is, up to $2t$ participants may be corrupted by the adversary provided that, at every stage of the protocol, the total number of corrupted participants is not larger than t.*

Theorem 3. *Given a set of $n = 2t + 1$ participants in unconditionally secure setting with a negligible failure probability, then there exists a t-resilient and t/2-private MPC protocol. In other words, only t/2 + 1 participants may not be corrupted, provided that, at every stage of the protocol, the total number of corrupted participants does not exceed the threshold parameter.*

The rest of this paper is organized as follows. In Section 2, we will give an overview of MPC protocols in the presence of an omnipresent adversary. In Section 3, we will present our approach to disqualification of malicious participants in MPC protocols with an honest majority. In Section 4, we will study the transformation of *t-resilient* protocols into *t'-private* protocols. In Sections 5 and 6 we will show how to modify the existing protocols, in order to simultaneously achieve correctness and privacy. Section 7 gives concluding remarks.

2 An Overview of MPC with Omnipresent Adversary

Designing MPC protocols under an assumption that participants are truly honest is unrealistic. If we could assume that a single trusted party existed, then the designing of MPC protocols would be easy. In this case, all participants first handed their inputs to the trusted party who would compute the function and announce the result to each participant. This scenario, which is known as an *ideal process*, has been studied in order to evaluate the security of *real-life* MPC protocols (see, e.g., [5]). Real-life MPC protocols allow n mutually distrusting participants to evaluate a function for their private inputs assuming that some participants are corrupted. Note that as passively corrupted participants follow the protocol, they cannot be eliminated from it. On the other side, malicious participants deviate from the protocol so they can be identified and eliminated from the protocol.

An omnipresent adversary can be seen as an entity which attempts to break either privacy or correctness of MPC protocols by trying to corrupt (passively or actively) some participants. In the case of passive corruption, the adversary has access to all the information held by the corrupted participant while the participant follows the protocol honestly. In the case of active corruption, the adversary has full control over the behavior of the participant who may deviate from the protocol in an arbitrary way. Observe that if a corrupted participant does not follow the protocol, she can be identified and removed from it.

A good example of an omnipresent adversary is a powerful enemy (such as rogue states, terrorist organizations, intelligence agencies, etc.) who is using its large resources to break MPC protocols by trying to corrupt the participants. Clearly, the adversary is not going to be involved in the protocol directly but it will use the corrupted participants to achieve its goals. From the omnipresent adversary point of view, it would like to achieve its goals with minimum expenses. The expenses are proportional to the number of participants that need to be corrupted. It may also be assumed that a passive corruption may be easier and cheaper than an active corruption.

Our adversarial model is more general and powerful. In particular, it is dynamic so the composition of corrupted participants may change throughout the protocol execution. The number of participants that can be corrupted is larger than the threshold parameter, with the restriction that at any time, the number of corrupted participants does not exceed the threshold parameter. That is, if some actively corrupted participants are detected and eliminated, the adversary is allowed to corrupt some other participants. Strictly speaking, in the computationally secure MPC protocols (e.g. [12]), a *t-resilient* protocol works if $n \geq 2t+1$. That is, up to t participants can be corrupted actively, and the remaining $t+1$ participants are assumed to be honest. In our model, up to t participants can be corrupted actively, and up to t participants can be corrupted passively (only one participant may not be corrupted). In the unconditionally secure protocols with perfect secrecy (e.g. [4]), a *t-resilient* protocol works if $n \geq 3t+1$, where up to $2t+1$ participants are honest. In our model, up to t participants can be corrupted actively, and up to t participants can be corrupted passively (i.e., there are at least $t+1$ honest participants). Similarly, in majority-honest MPC protocols with small probability of error (e.g., [16,2]), a *t-resilient* protocol works if $n \geq 2t+1$, where up to $t+1$ participants are honest. In our model, up to t participants can be corrupted actively, and up to $t/2$ participants can be corrupted passively, (i.e. there are at least $t/2+1$ honest participants).

2.1 Proactive Secret Sharing Scheme

One can see that the number of corrupted participants in the protocol life-time can be greater than the threshold parameter t. If our protocols are implemented using a static secret sharing, then the adversary who learns more than t shares will be able to recover the private information. To prevent this, we employ the well-known *proactive secret sharing* technique [13]. A proactive secret sharing ensures the privacy of a secret by periodically renewing the shares of participants, without changing the secret, in such a way that information gained by an adversary in one time period is useless for the adversary in another time period. In other words, in an ordinary *t-private* secret sharing, its privacy is assured if, throughout the entire life-time of the secret, the adversary is not able to compromise more than t shares. In contrast, for a *t-private* proactive secret sharing, its privacy is guaranteed if at any time period (between two consecutive renewals), the adversary does not compromise more than t shares.

The proactive secret sharing of [13] consists of n participants, where each participant is connected to a common broadcast channel C, where messages sent on C instantly reach each party connected to it. The time is divided into time-periods (e.g. a day, a week, etc.). At the beginning of each time-period the participants update their shares using an interactive share renewal protocol.

The adversary can corrupt participants at any moment. If a participant P_j is corrupted during an update phase T_{i+1}, it will be considered as corrupted during both time-periods T_i and T_{i+1}. If the adversary leaves a corrupted participant P_j before the update phase T_{i+1}, then the adversary will not have any control over the communications of P_j, and thus has no information about the updated shares of P_j (i.e. P_j is no longer corrupted). The underlying secret sharing scheme is the Shamir [17] threshold scheme. The number of participants is $n = 2t + 1$, where during each time-period, the adversary can corrupt up to t participants. Assume that a secret x_i is shared amongst the set of n participants, in a *t-private* manner. In the update phase, each participant P_j distributes $s_{j,i}\varPi_{i=1,i\neq j}^{n}\frac{j}{j-i}$ amongst all participants in a *t-private* manner. Each participant P_ℓ adds all new shares received during the update phase, and takes it as his share of the secret x_i, and deletes the old share $s_{\ell,i}$ plus all partial shares. This process is correct, because $x_i = \varSigma_{j=1}^{n} s_{j,i}\varPi_{i=1,i\neq j}^{n}\frac{j}{j-i}$. The verifiable secret sharing (VSS) used in [13] is computationally secure and based on the Feldman VSS [9]. However, unconditional security is achievable by utilizing the Pedersen VSS from [15].

There are some differences between the proactivization used in [13], and in the MPC protocols with omnipresent adversary. They are as follows.

(a) The purpose of the update phase in [13] is to correct the shares of the participants that have been corrupted by an active adversary or alternatively by errors caused by other problems such as system crashes, for instance. Note that for the randomization and degree-reduction, we do not correct the shares of corrupted participants. Instead, we identify the corrupted participants and eliminate them from the protocol.

(b) The update phase of [13] is performed at the beginning of each time-period, while in our MPC protocols, it is done at the randomization and degree reduction stage.

(c) Similarly to [13], we allow the adversary to leave some corrupted participants alone until the execution of the randomization and degree reduction protocols. After proactivization, actively corrupted participants are eliminated from the protocol, while passively corrupted participants who have not been controlled by the adversary will have new shares that are not known to the adversary, and therefore they are not considered as corrupted participants any more. Now, the adversary may wish to corrupt some new participants (from the set of all remaining participants). That is, in our MPC protocols, the set of corrupted participants is dynamic.

It is worth mentioning that the proactivization process will not add too much overhead to our MPC protocols. This is due to the fact that in MPC protocols, the degree reduction procedure is, indeed, a proactivization of the participant shares. The only information that needs to be re-shared is the share of each participant from the other participant secret information. We observe that this is also done in MPC protocols with an honest majority, since after the threshold parameter is decreased, all information is re-shared using the new threshold. The overhead applies only to MPC protocols with fixed threshold parameter.

3 Disqualification in MPC with Honest Majority

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants who wish to compute a function $Y = F(x_1, \ldots, x_n)$, where participants P_i hold their private inputs x_i, assuming that $n = 2t + 1$, and the initial threshold parameter is t. Disqualification of a malicious participant P_i requires the reconstruction of the information in the hand of P_i, otherwise the protocol cannot be completed. The elimination of malicious participants, however, has to reduce the threshold parameter, otherwise the current number of participants cannot interpolate the polynomial associated with their shared information. In the existing MPC protocols, elimination of each malicious participant decreases the threshold parameter by one. After elimination of t malicious participants, the threshold parameter becomes zero, i.e., the remaining participants learn all private information.

In this section we will present a new disqualification technique that preserves the privacy of all participants who properly follow the protocol. Let D denote the number of eliminated participants from the system (initially, $D = 0$). After detecting a malicious participant, say P_i $(1 \leq i \leq n)$, increase the value of D by one and perform the following steps.

1. If D is an odd integer, private information of P_i is reconstructed by the other participants. If this occurs in the initialization phase, no further action is required. If this occurs in the computation phase, relevant computations (i.e. multiplication of relevant shares, randomization, and degree reduction procedure) associated with P_i will be performed publicly. This process reveals the private input x_i (which is not an issue, since it is a random value), and one share associated with the private input of each participants who properly follow the protocol. After the threshold parameter is reduced by one and all values are re-shared (see the next item), the knowledge of these shares is redundant.
2. If D is an even integer, only the secret input x_i is reconstructed, and the threshold parameter is decreased by 1. If this occurs in the initialization phase, the remaining participants repeat the initialization phase using a threshold parameter $t' = t - 1$. If this occurs in the computation phase, the remaining participants re-share their partial results using a new threshold parameter $t' = t - 1$, and continue the protocol using this new threshold parameter t'. Implicitly, this is a proactivization of a (t, n)-threshold scheme to a $(t - 1, n - 2)$-threshold scheme.

As the result of applying our disqualification technique, after occurrence of t faults, the number of remaining participants in the system is $t + 1$, and the threshold parameter is $t' = t - t/2 = t/2$. Therefore, no subset of up to $t/2$ participants learn any additional information about the secret input of participants who have properly followed the protocol.

4 Decomposition of *t-resilient* Protocols

According to the definition of *t-resilient* protocols, after elimination of all malicious participants, the remaining participants must be able to complete the protocol. In other words, in the absence of all eliminated participants (even if they voluntarily withdrew from the protocol), the remaining participants must be able to complete the protocol.

Theorem 4. *Let a t-resilient MPC protocol π realizes task T for a group of n participants, then there exists a t'-private MPC protocol π' that realizes task T for a group of n' participants, where $n' \geq n - t$ and $t/2 \leq t' \leq t$.*

Proof. If no participant misbehaves, the *t-resilient* MPC protocol π realizes task T in a secure manner (they obtain the correct result, where privacy of all inputs is maintained in a *t-private* manner). That is, a *t-resilient* MPC protocol is necessarily a *t-private* MPC protocol. Note that the inverse statement is not true. Having a *t-private* MPC protocol that realizes a task T, does not imply that we can design a *t-resilient* MPC protocol for the task. If all (or some) of the malicious participants are eliminated from the protocol, the remaining participants must be able to complete the protocol. Completion of the protocol means that remaining participants continue to perform the protocol π. If all malicious participants are eliminated, the number of remaining participants will be at most $n' = n - t$ (since at most t participants are eliminated). That is, protocol π continues with n' participants, where no fault occurs. In this case, π is not necessarily a *t-private* protocol, since the threshold parameter may have been reduced to t', where $t/2 \leq t' \leq t$. This completes the proof, assuming that protocol π' is a version of protocol π in which the verification procedures are omitted.

Indeed, a common practice in designing a *t-resilient* protocol is to equip a *t'-private* protocol with mechanisms that can manage malicious participants. That is, a *t-resilient* protocol can be decomposed into two phases, namely, detection and elimination of malicious participants and then the execution of a *t'-private* protocol π' for a group of n' participants.

For example, consider computationally secure MPC protocols given by Goldreich et al. in [12]. They proved that there exists a *t-resilient* MPC protocol for a group of at least $2t+1$ participants, and there exists a *t-private* MPC protocol for a group of at least $n' = t + 1$ participants. In other words, their *t-resilient* protocol can be decomposed into two sub-protocols, namely, detection of malicious participants and running their *t-private* protocol for the participants that honestly follow the protocol.

In an unconditionally secure setting, the protocols studied in [4] and [7] indicate that their *t-resilient* MPC protocol works for a group of at least $3t + 1$ participants, and their *t-private* protocol works for a group of at least $2t+1$ participants. That is, their *t-resilient* protocols can be decomposed in similar way, first detection and elimination of malicious participants and next the execution of a *t-private* protocol.

In MPC protocols with honest majority (alternatively, with faulty minority), their *t-resilient* MPC protocol works with a group of at least $2t+1$ participants. Although they have not discussed the case of passive adversary, classical results indicate that a *t-private* requires at least $2t+1$ cooperating participants. However, in their *t-resilient* MPC protocol with at least $2t+1$ participants, after elimination of t misbehaving participants, the remaining number of participants is $t+1$. That is, decomposition of their *t-resilient* protocol gives a *t/2-private* protocol.

5 Perfectly Secure MPC with Omnipresent Adversary

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants who wish to compute a function $Y = F(x_1, \ldots, x_n)$, where each participant P_i holds her private input x_i. To construct a perfectly secure MPC with an omnipresent adversary, we employ the MPC protocol from [4]. In the case of passively corrupted participants, if $n \geq 2t+1$, their *t-private* MPC protocol provides perfect privacy. In the case of actively corrupted (malicious) participants, if $n \geq 3t+1$, their *t-resilient* MPC protocol provides perfect privacy, assuming that up to t participants can be malicious and other participants honestly follow the protocol. In their *t-resilient* protocol with $3t+1$ participants, after elimination of all misbehaving participants, there will be $2t+1$ remaining participants in the system, which is large enough to construct a *t-private* protocol. That is, decomposition of their *t-resilient* protocol should lead to a *t-private* protocol. Since the threshold parameter is fixed, the number of corrupt participants at any time must not exceed the threshold parameter t. That is, we modify their disqualification procedure as follows (the rest of the protocol remains unchanged):

1. After the detection and elimination of a malicious participant P_i (for detail procedure, see [4]), the remaining participants perform proactivization of their shares from the other participant secret inputs.
2. After proactivization, the adversary is allowed to corrupt a new participant, either passively or actively, subject to the condition that the number of actively corrupted participants in the life-time of the protocol does not exceed t.
3. The remaining participants continue the protocol as in [4]. After elimination of at most t malicious participants, the system consists of at least $2t+1$ participants, where up to t participants are corrupted passively. Classical results indicate that an unconditionally secure *t-private* MPC protocol exists for this set of participants. That is, a *t-resilient* protocol is converted to a *t-private* protocol.

Remark 1. In computationally secure MPC, the *t-resilient* protocol from [12] starts with $2t+1$ participants. Performing their protocol in the presence of an omnipresent adversary, and utilizing the above elimination technique, after elimination of t malicious participants, the system consists of $t+1$ participants, where t of them are corrupted passively. Results of [12] indicate that a computationally secure *t-private* MPC protocol exists for this set of participants. That is, a *t-resilient* protocol is converted to a *t-private* protocol.

5.1 Security Discussion

This modified protocol is as secure as the original MPC protocol from [4]. This is because the adversary cannot learn any additional information (due to proactivization, information obtained in one time period is useless for another time period, and at each time period the scheme is *t-private*). Also, correctness of the result will not be affected, because up to t additional passively corrupted participants honestly follow the protocol.

Moreover, if t_a and t_p denote the number of actively and passively corrupted participants at any time period, the following conditions always hold:

(a) $3t_a + t_p < n$.
Considering the fact that $t_a \le t$ and after elimination of any misbehaving participant, one participant will be corrupted passively, if k $(0 \le k \le t)$ participants are eliminated, $3t_a + t_p \le 3(t-k) + k = 3t - 2k \le n - k$.

(b) $2t_a + 2t_p < n$.
Similarly, $2t_a + 2t_p \le 2(t-k) + 2(k) = 2t < n - k$.

That is, at every stage, our protocol satisfies the results of [10].

Theorem 5. *Given an MPC protocol defined in [4]. A set of n participants can compute every function perfectly (t_a, t_p)-securely if and only if $3t_a + t_p < n$ and $2t_a + 2t_p < n$. The computation is polynomial in n and linear in the size of the circuit. This holds whether a broadcast channel is available or not.*

6 Honest Majority MPC with Omnipresent Adversary

In MPC protocols with honest majority, a set of $\mathcal{P} = \{P_1, \ldots, P_n\}$ participants wish to compute a function $Y = F(x_1, \ldots, x_n)$, where each participant P_i holds her private input x_i. Although existing MPC with honest majority do not consider the case of a passive adversary, the condition $n \ge 2t + 1$ is the tight bound for designing a *t-private* MPC protocol, even if a broadcast channel is available. In the case of an active adversary, assuming that a public channel exists, a *t-resilient* MPC is achievable if $n \ge 2t + 1$. The secrecy of these protocols is unconditional, with error probability ϵ, which can be exponentially small.

To construct an MPC with honest majority in the presence of an omnipresent adversary, we employ the protocol from [16]. However, we utilize our disqualification technique (see Section 3), that ensures the privacy of participants who properly follow the protocol. So, our construction for MPC with honest majority works as follows:

1. After the detection and elimination of every two misbehaving participants, the threshold parameter is reduced by one (see our disqualification in Section 3). The process of re-sharing all information with a new threshold parameter $t' = t - 1$, implicitly, is a proactivization of the shares associated with the function value and the participant inputs.

2. After reducing the threshold, the adversary is allowed to corrupt a new participant, either in passive or active mode, provided that the number of actively corrupted participants in the life-time of the protocol does not exceed t.
3. The remaining participants continue the protocol as in [16], using the new threshold t'. After elimination of at most t misbehaving participants, the system consists of at least $t+1$ participants, where up to $t/2$ participants are corrupted passively. Classical results indicate that an unconditionally secure *t/2-private* MPC protocol exists for this set of participants. That is, a *t-resilient* protocol is converted to a *t/2-private* protocol.

We observe that the "honest majority" (alternatively "faulty minority") title is more suitable for our protocol (see above), since at every stage of the protocol, *only* the minority/majority of participants in the system are corrupt/honest. While in [16,2], after elimination of corrupt players, all remaining participants are honest.

6.1 Security Discussion

The modified protocol is as secure as the original MPC protocol from [16]. This is true because the adversary cannot learn any additional information since at each time period the scheme is *t'-private*. Also, correctness of the result will not be affected, because up to $t/2$ additional passively corrupted participants honestly follow the protocol.

Moreover, if t_a and t_p denote the number of actively and passively corrupted participants at any time period, the condition $2t_a + 2t_p < n$, is satisfied. This is because $2t_a + 2t_p \leq 2(t-k) + 2k/2 \leq 2t - k < n - k$. So, at every stage, our protocol satisfies the results of [10].

Theorem 6. *Given the MPC protocol defined in [16]. Then allowing an negligible failure probability and given a broadcast channel, a set of n participants can compute every function (t_a, t_p)-securely if and only if $2t_a + 2t_p < n$. The computation is polynomial in n and linear in the size of the circuit.*

7 Conclusions

We have investigated the privacy of MPC protocols in the presence of omnipresent adversary. The omnipresent adversary can be either passive, active, or mixed. We have shown that up to a minority of participants who are not corrupted actively, can be corrupted passively, with the restriction that at any time, the number of corrupt participants does not exceed a predetermined threshold.

Our adversary model stipulates that MPC protocols never run with a set of truly honest participants (which is a more realistic assumption). Therefore, privacy of all participants who properly follow the protocol will be maintained.

Remark 2. In this paper we have used the protocols from [4] and [16] (that are perfectly secure with a negligible failure probability) and showed how the omnipresent adversary works for these protocols. For the perfect security case, the

Table 1. Comparison of existing *t-resilient* MPC protocols and MPC protocols with an omnipresent adversary

Security Model	Adversary model	Number of participants	Actively corrupted	Passively corrupted	Uncorrupted participants
Computational	Active	$n = 2t + 1$	t	0	$t + 1$
	Omnipresent	$n = 2t + 1$	t	t	1
Unconditional without broadcast channel	Active	$n = 3t + 1$	t	0	$2t + 1$
	Omnipresent	$n = 3t + 1$	t	t	$t + 1$
Unconditional with broadcast channel	Active	$n = 2t + 1$	t	0	$t + 1$
	Omnipresent	$n = 2t + 1$	t	$t/2$	$t/2 + 1$

maximum number of corrupted participants at any time is t, and for MPC protocols with a negligible failure probability, the maximum number of corrupted participants at any time is t' ($t/2 \leq t' \leq t$), see Table 1. Applying the protocol from [10] improves these bounds, but will not increase the total number of participants that can be corrupted in the protocol life-time. This is because, in the presence of a passive adversary, $n > 2t$ (or $n \geq 2t+1$) is a tight bound regardless of a type of the protocol.

Acknowledgments

We are grateful to the anonymous referees for their constructive comments. The second co-author was supported by Australian Research Council grant DP0663452.

References

1. Almansa, J., Damgård, I., Nielsen, J.: Simplified Threshold RSA with Adaptive and Proactive Security. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 593–611. Springer, Heidelberg (2006)
2. Beaver, D.: Multiparty protocols tolerating half faulty processors. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 560–572. Springer, Heidelberg (1990)
3. Beaver, D.: Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. Journal of Cryptology 4, 75–122 (1991)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorem for Non-Cryptographic Fault-Tolerant Distributed Computation. In: Proceedings of the 20th ACM Annual Symposium on the Theory of Computing (STOC 1988), pp. 1–10 (1988)
5. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology 13, 143–202 (2000)
6. Catalano, D., Gennaro, R., Halevi, S.: Computing Inverses over a Shared Secret Modulus. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 190–206. Springer, Heidelberg (2000)
7. Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Proceedings of the 20th ACM Annual Symposium on the Theory of Computing (STOC 1988), pp. 11–19 (1988)

8. Dolev, D., Dwork, C., Waarta, O., Yung, M.: Perfectly Secure Message Transmission. Journal of the ACM 40, 17–47 (1993)
9. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: 28^{th} IEEE Symposium on Foundations of Computer Science, pp. 427–437 (October 1987)
10. Fitzi, M., Hirt, M., Maurer, U.: Trading correctness for privacy in unconditional multi-party computation. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 121–136. Springer, Heidelberg (1998)
11. Gennaro, R., Rabin, M., Rabin, T.: Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography. In: 17th Annual ACM Symposium on Principles of Distributed Computing, pp. 101–111 (1998)
12. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game. In: Proceedings of the 19th ACM Annual Symposium on the Theory of Computing (STOC 1987), May 25–27, pp. 218–229 (1987)
13. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
14. Hirt, M., Maurer, U., Przydatek, B.: Efficient Secure Multi-party Computation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 143–161. Springer, Heidelberg (2000)
15. Pedersen, T.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
16. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: Proceedings of the 21th ACM Annual Symposium on the Theory of Computing (STOC 1989), pp. 73–85 (1989)
17. Shamir, A.: How to Share a Secret. Communications of the ACM 22, 612–613 (1979)
18. Yao, A.: Protocols for Secure Computations. In: The 23rd IEEE Symposium on the Foundations of Computer Science, pp. 160–164 (1982)

Blind and Anonymous Identity-Based Encryption and Authorised Private Searches on Public Key Encrypted Data

Jan Camenisch[1], Markulf Kohlweiss[2], Alfredo Rial[2], and Caroline Sheedy[3]

[1] Zurich Research Lab
IBM Research
jca@zurich.ibm.com
[2] ESAT-COSIC / IBBT
Katholieke Universiteit Leuven
{markulf.kohlweiss,alfredo.rialduran}@esat.kuleuven.be
[3] School of Computing
Dublin City University
csheedy@computing.dcu.ie

Abstract. Searchable encryption schemes provide an important mechanism to cryptographically protect data while keeping it available to be searched and accessed. In a common approach for their construction, the encrypting entity chooses one or several keywords that describe the content of each encrypted record of data. To perform a search, a user obtains a trapdoor for a keyword of her interest and uses this trapdoor to find all the data described by this keyword.

We present a searchable encryption scheme that allows users to privately search by keywords on encrypted data in a public key setting and decrypt the search results. To this end, we define and implement two primitives: public key encryption with *oblivious* keyword search (PEOKS) and *committed blind anonymous* identity-based encryption (IBE). PEOKS is an extension of public key encryption with keyword search (PEKS) in which users can obtain trapdoors from the secret key holder without revealing the keywords. Furthermore, we define committed blind trapdoor extraction, which facilitates the definition of authorisation policies to describe which trapdoor a particular user can request. We construct a PEOKS scheme by using our other primitive, which we believe to be the first blind and anonymous IBE scheme.

We apply our PEOKS scheme to build a public key encrypted database that permits authorised private searches, i.e., neither the keywords nor the search results are revealed.

Keywords: Blind identity-based encryption, searchable encryption, public key encryption with keyword search.

1 Introduction

Vast quantities of sensitive personal data are retained for the purpose of network forensics and cyber investigations [1]. The advantages of the availability of such data for the investigation of serious crimes and the protection of national security are considerable.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 196–214, 2009.

However, these advantages must be counterpoised by the dangers that such data could fall into the wrong hands.

The encryption of retained data is a desirable counter measure against data theft. But how, then, can the investigator, such as the police or a secret service, search the data without having to decrypt the whole database? What if the investigator should only be given access to data that fulfills certain criteria? This seems to be a hard problem, as the criteria themselves may be sensitive and thus requiring protective measures, such as encryption. Moreover, a secret service is often reluctant to reveal the type of queries it wants to run on the encrypted database.

We consider a scenario in which an investigator searches for data described by multiple keywords without revealing the keywords or the search results to the database server. This scenario is akin to the private searching of streaming data presented in [2]. While in [2] the data is searched as it is generated (and can thereafter be discarded), in our scenario data is first stored in encrypted form and can be searched at a later stage. To provide a high level of security we make use of asymmetric cryptography. The database server only possesses the public encryption key (and cannot decrypt the retained data itself). In this way, data that is already encrypted remains secure even against a strong adversary that breaks into the database server. The decryption key is stored by a security server, which will only be involved when executing search queries.

As the details of queries made are to be obscured even from the security server, it is necessary to impose some restrictions on the investigator. Thus we introduce some checks and balances to avoid abuse by overzealous or malicious investigators. One obvious restriction is in the number of queries that the investigator can make. An unreasonable number of requests may be an indication of abuse. Another restriction that we consider is to involve a judge in granting search warrants to the investigator. The keyword is still hidden, but the security server is guaranteed that a judge (or another authority figure) has approved the search for a specific keyword.

In [3] the authors build an encrypted and searchable audit log. They propose two schemes, one based on symmetric encryption and one based on asymmetric encryption. They conclude that asymmetric encryption provides better security, as it reduces the trust in the encrypting entity. Our work can be seen as an extension of their asymmetric scheme with the possibility to obliviously search the encrypted database. For the symmetric case, in which the audit log server knows all the information needed for decrypting the database, the problem of performing oblivious searches is covered by [4,5]. The problem of oblivious searching on public key encrypted data is more difficult.

Outline of our solution. In [3], the asymmetric searchable encryption scheme is based on identity-based encryption (IBE) [6]. The keywords themselves are used to encrypt the database, i.e., they are the identity strings of the IBE scheme. The anonymity property of Boneh-Franklin IBE scheme [6] ensures that a ciphertext does not leak the identity string used to generate the encryption. The security server holds the master secret key that is used to derive the secret keys corresponding to the keywords that are needed for searching. A similar technique for searchable encryption was formalized as public key encryption with keyword search (PEKS) by [7]. In PEKS, the derived keys are referred to as search trapdoors, which can be given to third parties to grant them search rights.

When trying to build an oblivious search mechanism for such a database we have to address two difficulties: hiding the keywords from the security server and hiding the search results from the database. For the former, we present two new cryptographic primitives. The first one is *committed blind anonymous* IBE. In this context, *anonymous* means that the ciphertext does not leak the key (identity) under which it was encrypted [8,9] and *blind* means that a user can request the decryption key for a given identity without the key generation entity learning the identity [10]. The work of [10] describes how to construct blind key derivation protocols for [11] and [12,13], but these schemes are not anonymous. Moreover, it is much harder to derive a blind key derivation protocol for the Boneh-Franklin IBE scheme [6] used in [3], and we are interested in IBE schemes that do not require random oracles for their security proofs. (As shown by [14,15,16], a scheme may be insecure even if proven secure in the random oracle model.) As a corollary to our results, we obtain the first instantiation of [3] secure without random oracles.

We design a committed blind anonymous IBE scheme based on the anonymous IBE scheme due to [9]. As the scheme in [9] is only selective ID secure [11], we extend it with adaptive ID security [17] and prove the modified scheme secure. For the modified scheme we design a blind key extraction protocol. This leads to the first blind anonymous IBE scheme we are aware of. We extend the definition of blind IBE to allow for the derivation of a secret key for a committed identity. This allows the key generation entity to enforce authorisation policies on the identities for which a secret key is requested, as described in [18].

The second primitive we present is public key encryption with oblivious keyword search (PEOKS), which we implement using our committed blind anonymous IBE scheme. First, we extend the definition of PEKS to incorporate the encryption of a secret message when computing a searchable encryption. This secret message can contain a symmetric key, which allows PEKS to be used directly in settings such as [3]. Then we define blind key extraction with committed keywords, which facilitates the use of a policy that states for which keywords a trapdoor can be extracted while still keeping them hidden from the trapdoor generation entity.

In order to hide the search results from the database one could in theory download the whole database and then use PEOKS to do the search. This is inefficient. We describe a data structure that allows to use private information retrieval (PIR) [19] to improve the communication efficiency of the search.

Our contribution. We define and construct the first blind anonymous IBE scheme. We generalize PEKS to be usable in settings such as [3], and we extend it to incorporate the facility to perform oblivious keywords searches (PEOKS). Both our blind anonymous IBE scheme and our PEOKS scheme support committed blind key extraction and thus allow for complex policies. Finally, we describe the first public key encrypted database that allows for oblivious searches, i.e., both the keywords and the search results remain hidden.

Outline of the paper. In Sect. 2 we introduce basic concepts and security assumptions and in Sect. 3 we define committed blind anonymous IBE and PEOKS. We construct a committed blind anonymous IBE scheme and we show how to apply it to build a PEOKS scheme in Sect. 4. In Sect. 5, we describe the use of PEOKS to construct a privacy-preserving searchable encrypted database. Finally, Sect. 6 draws a conclusion and discusses future work.

2 Technical Preliminaries

A function ν is *negligible* if, for every integer c, there exists an integer K such that for all $k > K$, $|\nu(k)| < 1/k^c$. A problem is said to be *hard* (or *intractable)* if there exists no probabilistic polynomial time (p.p.t.) algorithm on the size of the input to solve it. ϵ denotes the empty string.

Bilinear Maps. Let G_1, G_2 and G_T be groups of prime order p. A map $e : G_1 \times G_2 \rightarrow G_T$ must satisfy the following properties:

(a) *Bilinearity.* A map $e : G_1 \times G_2 \rightarrow G_T$ is bilinear if $e(a^x, b^y) = e(a, b)^{xy}$;
(b) *Non-degeneracy.* For all generators $g \in G_1$ and $h \in G_2$, $e(g, h)$ generates G_T;
(c) *Efficiency.* There exists an efficient algorithm $\mathsf{BMGen}(1^k)$ that outputs $(p, G_1, G_2, G_T, e, g, h)$ to generate the bilinear map and an efficient algorithm to compute $e(a, b)$ for any $a \in G_1, b \in G_2$.

The security of our scheme is based on the following number-theoretic assumptions:

Definition 1 (Decision BDH). *Given $g, g^a, g^b, g^c \in G_1$, $h, h^a, h^b \in G_2$, and $Z \in G_T$ for random exponents $a, b, c \in Z_p$, decide whether $Z = e(g, h)^{abc}$ or a random element from G_T. The Decision BDH assumption holds if all p.p.t algorithms have negligible advantage in solving the above problem.*

Definition 2 (Decision Linear). *Given $g, g^a, g^b, g^{ac}, g^{bd}, Z \in G_1$, $h, h^a, h^b \in G_2$ for random exponents $a, b, c, d \in Z_p$, decide whether $Z = g^{c+d}$ or a random element in G_1. The Decision Linear assumption holds if all p.p.t algorithms have negligible advantage in solving the above problem.*

Commitment Schemes. A *commitment scheme* is a two phase scheme that allows a user to *commit* to a hidden value, while preserving the ability of the user to *reveal* the committed value at a later stage. The properties of a commitment scheme are *hiding*: the value committed to must remain undiscovered until the reveal stage, and *binding*: the only value which may be revealed is the one that was chosen in the commit stage.

We use the perfectly hiding commitment scheme proposed by Pedersen [20]: Given a group G of prime order p with generators g and h, generate a commitment C to $x \in Z_p$ by choosing at random $open_x \leftarrow Z_p$ and computing $C = g^x h^{open_x}$. The commitment is opened by revealing x and $open_x$.

Proofs of Knowledge. We use several existing results to prove statements about discrete logarithms; (1) proof of knowledge of a discrete logarithm modulo a prime [21], (2) proof of knowledge of the equality of some element in different representations [22], (3) proof that a commitment opens to the product of two other committed values [23,24,25], and (4) proof of the disjunction or conjunction of any two of the previous [26]. These results are often given in the form of Σ-protocols but they can be turned into zero-knowledge protocols using efficient zero-knowledge compilers [27,28].

When referring to the proofs above, we follow the notation introduced by Camenisch and Stadler [29] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms.

$$PK\{(\alpha, \beta, \delta) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers α, β, and δ such that $y = g^\alpha h^\beta$ and $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta$ holds*", where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$ that have the same order. (Note that some elements in the representation of y and $\tilde{y}$ are equal.) The convention is that letters in the parenthesis, in this example α, β, and δ, denote quantities whose knowledge is being proven, while all other values are known to the verifier. There exists a knowledge extractor which can extract these quantities from a successful prover.

3 Definitions of Committed Blind Anonymous IBE and PEOKS

3.1 Anonymous Identity-Based Encryption

We recall the definition of identity-based encryption [6]. An IBE scheme Π consists of the algorithms ($\mathsf{IBESetup}, \mathsf{IBEExtract}, \mathsf{IBEEnc}, \mathsf{IBEDec}$):

$\mathsf{IBESetup}(1^k)$ outputs parameters $params$ and master secret key msk.
$\mathsf{IBEExtract}(params, msk, id)$ outputs the secret key sk_{id} for identity id.
$\mathsf{IBEEnc}(params, id, m)$ outputs ct encrypting m under id.
$\mathsf{IBEDec}(params, sk_{id}, ct)$ outputs message m encrypted in ct.

An IBE scheme is *anonymous* [30], if it is not possible to associate the identity id used to encrypt a message m with the resulting ciphertext ct (in the context of public key encryption this is also known as key privacy [31]).

Abdalla et al. [30] define anonymity through a security game in which the adversary receives a ciphertext encrypted with an identity that is randomly picked from two identities of his choosing. The adversary has to guess the identity used to encrypt the ciphertext. As in [8], we combine this game with the standard chosen plaintext security game for IBE in which the adversary needs to guess which message out of two possible messages was encrypted.[1]

Definition 3 (Secure Anonymous IBE [30]). *Let k be a security parameter. An anonymous IBE scheme Π is secure if every p.p.t. adversary $\mathcal{A}$ has an advantage negligible in k in the following game:*

Setup. *The game runs $\mathsf{IBESetup}(1^k)$ to generate $(params, msk)$.*

Phase 1. *$\mathcal{A}$ may query an oracle $\mathcal{O}_{\mathsf{IBEExtract}}(params, msk, id)$ polynomially many times with input id. The oracle then runs $\mathsf{IBEExtract}(params, msk, id)$ and returns the associated sk_{id}.*

Challenge. *$\mathcal{A}$ presents the simulator two target identities id_0, id_1, which have not*

[1] We define an adaptive identity security game as this is required by our IBE to PEOKS transformation.

been queried in Phase 1, and two challenge messages m_0, m_1. The simulator selects two random bits b_1 and b_2, and returns to $\mathcal{A}$ the challenge ciphertext $ct = \mathsf{IBEEnc}(params, id_{b_1}, m_{b_2})$.

Phase 2. *$\mathcal{A}$ may again query oracle $\mathcal{O}_{\mathsf{IBEExtract}}(params, msk, id)$ polynomially many times on id provided id is not id_0 or id_1.*

Guess. *$\mathcal{A}$ outputs b_1', b_2'. We define the advantage of $\mathcal{A}$ as $|Pr[b_1' = b_1 \wedge b_2' = b_2] - 1/4|$.*

3.2 Committed Blind Anonymous IBE

In standard IBE schemes, a key generation centre $\mathcal{KGC}$ executes the *key extraction* algorithm IBEExtract that returns the secret key sk_{id} corresponding to input identity id. Green and Hohenberger [10] propose extracting the secret key in a *blinded* manner. The blinding action obscures the identity from the $\mathcal{KGC}$. We extend this concept by proposing a *committed blind anonymous* IBE scheme, where the $\mathcal{KGC}$ is given a commitment to the requested identity. A user can reveal partial information about the identity or prove statements about it using efficient zero-knowledge proofs about commitments [32,25].[2]

A committed blind anonymous IBE scheme consists of the algorithms Π of an IBE scheme, a secure commitment scheme Commit, and the protocol IBEBlindExtract:

$\mathsf{IBEBlindExtract}(\mathcal{U}(params, id, open_{id}), \mathcal{KGC}(params, msk, C))$ generates the secret decryption key sk_{id} for $\mathcal{U}$'s identity id in an interactive key issuing protocol between $\mathcal{U}$ and the $\mathcal{KGC}$. If $C = \mathsf{Commit}(id, open_{id})$, $\mathcal{U}$'s output is a decryption key sk_{id} and the output of the $\mathcal{KGC}$ is empty. Otherwise both parties output $\perp$.

Green and Hohenberger [10] construct a security argument for blind-IBE by defining two properties for the IBEBlindExtract protocol: leak freeness and selective-failure blindness. Leak freeness requires that IBEBlindExtract is a secure two-party computation that does not leak any more information than IBEExtract.[3] Selective-failure blindness requires that a potentially malicious authority does not learn anything about the user's identity during the IBEBlindExtract protocol. Additionally, it cannot cause the IBEBlindExtract protocol to selectively fail depending on the user's choice of identity. We provide adapted versions of these properties for committed blind anonymous IBE.

Definition 4 (Leak Freeness [10]). *An* IBEBlindExtract *protocol of an* IBE *scheme is leak free if, for all efficient adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for every value k, no efficient distinguisher $\mathcal{D}$ can determine whether it is playing Game Real or Game Ideal with non-negligible advantage, where*

Game Real: *Run* $\mathsf{IBESetup}(1^k)$. *As many times as $\mathcal{D}$ wants, he picks a commitment C and $\mathcal{A}$'s input state. $\mathcal{A}$ runs* $\mathsf{IBEBlindExtract}(\mathcal{A}(params, state), \mathcal{KGC}(params, msk, C))$ *with the $\mathcal{KGC}$. $\mathcal{A}$ returns the resulting view to D.*

[2] Technically this can be seen as restricting the blind key derivation queries to a certain language, membership of which is proven in zero-knowledge.

[3] It also implies that the user is required 'to know' the id for which she needs a key to be extracted. We also require that she knows the opening to the commitment.

Game Ideal: *Run* $\mathsf{IBESetup}(1^k)$*. As many times as* $\mathcal{D}$ *wants, he picks a commitment* C *and initial input state.* $\mathcal{S}$ *obtains* $(params, state)$ *and may choose values* id *and* $open_{id}$ *to query an oracle* $\mathcal{O}_{\mathsf{IBEExtract}}$ *that knows* msk *and is parameterized with* C*. If* $C = \mathsf{Commit}(id, open_{id})$*, the oracle returns key* $sk_{id} \leftarrow \mathsf{IBEExtract}(params, msk, id)$*, otherwise* $\perp$*.* $\mathcal{S}$ *returns a simulated view to* $\mathcal{D}$*.*

Definition 5 (Selective-Failure Blindness [33]). *An* $\mathsf{IBEBlindExtract}$ *protocol is said to be selective-failure blind if every adversary* $\mathcal{A}$ *has a negligible advantage in the following game:* $\mathcal{A}$ *outputs params and a pair of identities* id_0, id_1*. A random bit* $b \in \{0,1\}$ *is chosen, and* $\mathcal{A}$ *is given two fresh commitments* C_b*,* C_{1-b} *and black-box access to two oracles:* $\mathcal{U}(params, id_b, open_{id_b})$ *and* $\mathcal{U}(params, id_{1-b}, open_{id_{1-b}})$*. The* $\mathcal{U}$ *algorithms produce* sk_b, sk_{1-b} *respectively. If* $sk_b \neq \perp$ *and* $sk_{1-b} \neq \perp$*,* $\mathcal{A}$ *receives* (sk_0, sk_1)*; if only* $sk_{1-b} = \perp$*,* $(\epsilon, \perp)$*; if only* $sk_b = \perp$*,* $(\perp, \epsilon)$*; and if* $sk_b = sk_{1-b} = \perp$*,* $\mathcal{A}$ *receives* $(\perp, \perp)$*. Finally,* $\mathcal{A}$ *outputs his guess* b'*. The advantage of* $\mathcal{A}$ *in this game is* $|Pr[b' = b] - 1/2|$.

Following [10], we define a secure committed blind anonymous IBE as follows.

Definition 6 (Secure Committed Blind Anonymous IBE). *A committed blind anonymous IBE scheme* $(\Pi, \mathsf{IBEBlindExtract}, \mathsf{Commit})$ *is secure if and only if: (1) The underlying* Π *is a secure anonymous IBE scheme, (2)* Commit *is a secure commitment scheme, and (3)* $\mathsf{IBEBlindExtract}$ *is leak free and selective-failure blind.*

3.3 Public Key Encryption with Oblivious Keyword Search

We recall and extend the definition of PEKS [7]. A PEKS scheme $\Upsilon = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$ consists of the algorithms:

$\mathsf{KeyGen}(1^k)$ outputs a public key A_{pub} and secret key A_{priv}.
$\mathsf{PEKS}(A_{pub}, W, m)$ outputs a searchable encryption S_W of m under keyword W.
$\mathsf{Trapdoor}(A_{pub}, A_{priv}, W)$ outputs a trapdoor T_W that allows to search for the keyword W.
$\mathsf{Test}(A_{pub}, S_W, T_{W'})$ outputs the message m encoded in S_W, if $W = W'$; otherwise it outputs $\perp$.

This definition of PEKS extends the standard definition [7] by encoding a secret m into the PEKS element S_W generated by the PEKS algorithm. Test outputs this secret when a match occurs.

A secure PEKS scheme must be *chosen plaintext attack* (CPA) secure and *consistent* [30]. CPA security requires that an attacker cannot distinguish two PEKS elements generated for keywords and messages of his choice, even if given oracle access to $\mathsf{Trapdoor}$ for other keywords. Consistency requires that if the searchable encryption and the trapdoor are computed using different keywords, then algorithm Test should output $\perp$ upon such input.

In PEKS, the party holding the secret key A_{priv} runs the $\mathsf{Trapdoor}$ algorithm to obtain the trapdoor T_W for a keyword W. Public key encryption with oblivious keyword search (PEOKS) is an extension of PEKS in which a user $\mathcal{U}$ performing a search can obtain in a *committed* and *blinded* manner the trapdoor T_W from the trapdoor generation entity $\mathcal{TGE}$. The $\mathcal{TGE}$ only learns a commitment to the search term.

A PEOKS scheme consists of the algorithms Υ of a PEKS scheme, a secure commitment scheme Commit used to commit to keywords, and the following $\mathsf{BlindTrapdoor}$ protocol:

$\mathsf{BlindTrapdoor}(\mathcal{U}(A_{pub}, W, open_W), \mathcal{TGE}(A_{pub}, A_{priv}, C))$ generates a trapdoor T_W for a keyword W in an interactive protocol between $\mathcal{U}$ and $\mathcal{TGE}$. If C=$\mathsf{Commit}(W, open_W)$, $\mathcal{U}$'s output is the trapdoor T_W and the output of $\mathcal{TGE}$ is empty. Otherwise both parties output $\perp$.

Leak freeness and *selective-failure blindness* can be defined for $\mathsf{BlindTrapdoor}$ following the definition for $\mathsf{IBEBlindExtract}$.[4] We define the security of PEOKS similarly to that of a committed blind anonymous IBE scheme, assuming a secure underlying PEKS scheme.

Definition 7 (Secure PEOKS). *A* PEOKS *scheme (Υ,* $\mathsf{BlindTrapdoor}$, Commit*) is secure if and only if: (1) The underlying Υ is a secure PEKS scheme, (2)* Commit *is a secure commitment scheme, and (3)* $\mathsf{BlindTrapdoor}$ *is leak free and selective-failure blind.*

4 Construction of a Committed Blind Anonymous IBE Scheme and a Transformation to PEOKS

4.1 The Underlying Anonymous IBE Scheme

We present an anonymous IBE scheme that is adaptive identity secure in the standard model, based on the anonymous IBE scheme proposed by Boyen-Waters [9]. The Boyen-Waters scheme is selective identity secure. We use a transformation due to Naccache [12], a variant of that of Waters [17], to achieve the required adaptive identity security. The use of such a transformation was proposed by Boyen-Waters [9]. We provide what we believe to be the first proof of security for this variant. Our scheme supports asymmetric bilinear maps, allowing the use of a wider range of potentially more efficient implementations using different pairing types [34]. Let identity $id \in \{0,1\}^{\ell \times n}$ and let $id_1 \| \ldots \| id_n = id$ be the separation of id into ℓ bit integers id_i. Let $H_1(id) = g_0 \prod_{i=1}^{n} g_i^{id_i}$ and $H_2(id) = h_0 \prod_{i=1}^{n} h_i^{id_i}$. Our anonymous IBE scheme $\Pi = (\mathsf{IBESetup}, \mathsf{IBEExtract}, \mathsf{IBEEnc}, \mathsf{IBEDec})$ consists of the following algorithms :

$\mathsf{IBESetup}(1^k)$. Run $\mathsf{BMGen}(1^k)$ to obtain a bilinear map setup $(p, G_1, G_2, G_T, e, g, h)$. Choose values $\alpha, z_0, z_1, \ldots, z_n, t_1, t_2, t_3, t_4 \leftarrow Z_p^*$ and keep $msk = (\alpha, t_1, t_2, t_3, t_4)$ as the master key. Compute the system parameters as

$$params = \Big(\Omega = e(g,h)^{t_1 t_2 \alpha}, g, h, g_0 = g^{z_0}, \ldots, g_n = g^{z_n}, v_1 = g^{t_1}, \ldots, v_4 = g^{t_4}, h_0 = h^{z_0}, \ldots, h_n = h^{z_n}\Big).$$

[4] The inputs are mapped 1-to-1. KeyGen is used instead of $\mathsf{IBESetup}$ and $\mathsf{Trapdoor}$ instead of $\mathsf{IBEExtract}$.

$\mathsf{IBEExtract}(params, msk, id)$. Choose two random values $\tilde{r}_1, \tilde{r}_2 \leftarrow Z_p^*$ and compute the key

$$sk_{id} = \Big(h^{\tilde{r}_1 t_1 t_2 + \tilde{r}_2 t_3 t_4}, h^{-\alpha t_2} H_2(id)^{-\tilde{r}_1 t_2}, h^{-\alpha t_1} H_2(id)^{-\tilde{r}_1 t_1}, H_2(id)^{-\tilde{r}_2 t_4}, H_2(id)^{-\tilde{r}_2 t_3}\Big) .$$

$\mathsf{IBEEnc}(params, id, msg)$. To encrypt a message $msg \in G_T$, choose $s, s_1, s_2 \leftarrow Z_p$, and generate the ciphertext

$$ct = \Big(\Omega^s \cdot msg, H_1(id)^s, v_1^{s-s_1}, v_2^{s_1}, v_3^{s-s_2}, v_4^{s_2}\Big).$$

$\mathsf{IBEDec}(params, sk_{id}, ct)$. Parse sk_{id} as $(d_0, d_1, d_2, d_3, d_4)$ and ct as $(c', c_0, c_1, c_2, c_3, c_4)$ and return

$$msg = c' \cdot e(c_0, d_0) \cdot e(c_1, d_1) \cdot e(c_2, d_2) \cdot e(c_3, d_3) \cdot e(c_4, d_4).$$

Theorem 1. *The scheme Π is an adaptive identity secure anonymous IBE scheme under the DBDH and DLIN assumptions.* Please see the full version for the proof.

4.2 Blind Extraction Protocol

We introduce an interactive blind key extraction protocol IBEBlindExtract, which extends algorithm IBEExtract.

Intuition behind our construction. Generating a randomly distributed secret key by means of the IBEBlindExtract protocol requires the values $\tilde{r}_1, \tilde{r}_2$ to be jointly chosen by the user and the key issuer in a manner which prevents either party from learning anything about the other's randomness. This prevents a user that learns the issuer's randomness from potentially decrypting messages of other users and an issuer that learns a user's randomness from potentially breaking the blindness of the key issued.

The key issuer, $\mathcal{KGC}$, chooses random values $\hat{r}_1, \hat{r}_2 \leftarrow Z_p^*$, and the user $\mathcal{U}$ picks random values $r_1', r_2' \leftarrow Z_p^*$. The key generation protocol may be implemented using standard secure two-party computation techniques [35], as a protocol in which the user inputs r_1', r_2' and the $\mathcal{KGC}$ inputs $\alpha, t_1, t_2, t_3, t_4, \hat{r}_1, \hat{r}_2$. The user's output in the protocol is a secret key

$$sk_{id} = (h^{\tilde{r}_1 t_1 t_2 + \tilde{r}_2 t_3 t_4}, h^{-\alpha t_2} H_2(id)^{-\tilde{r}_1 t_2}, h^{-\alpha t_1} H_2(id)^{-\tilde{r}_1 t_1}, H_2(id)^{-\tilde{r}_2 t_4}, H_2(id)^{-\tilde{r}_2 t_3}) ,$$

with $\tilde{r}_1 = \hat{r}_1 r_1'$ and $\tilde{r}_2 = \hat{r}_2 r_2'$. The $\mathcal{KGC}$ learns nothing further, and outputs nothing. By decomposing this protocol into sub-protocols, whose results only require simple arithmetic operations (addition and multiplication), we obtain an efficient protocol.

Construction. Our committed blind anonymous IBE scheme consists of the algorithms Π of the underlying IBE scheme, the Pedersen commitment scheme Commit, and the following IBEBlindExtract protocol:

$\mathsf{IBEBlindExtract}(\mathcal{U}(params, id, open_{id}) \leftrightarrow \mathcal{KGC}(params, msk, C))$.

1. $\mathcal{KGC}$ chooses at random $\hat{r}_1, \hat{r}_2 \leftarrow Z_p^*$, and the user $\mathcal{U}$ chooses at random $u_0, u_1, u_2 \leftarrow Z_p$ and $u_3, r_1', r_2' \leftarrow Z_p^*$. Implicitly, $\tilde{r}_1 = \hat{r}_1 r_1'$ and $\tilde{r}_2 = \hat{r}_2 r_2'$. $\mathcal{U}$ computes $C_{u_3} = \mathsf{Commit}(u_3, \mathit{open}_{u_3})$, and $\mathcal{KGC}$ computes $C_{\hat{r}_1} = \mathsf{Commit}(\hat{r}_1, \mathit{open}_{\hat{r}_1})$ and $C_{\hat{r}_2} = \mathsf{Commit}(\hat{r}_2, \mathit{open}_{\hat{r}_2})$. $\mathcal{KGC}$ and $\mathcal{U}$ make use of a two-party protocol for simple arithmetics modulo p (parameterized by C_{u_3}, $C_{\hat{r}_1}$, and $C_{\hat{r}_2}$). $\mathcal{U}$ inputs $u_0, u_1, u_2, u_3, \mathit{open}_{u_3}, r_1', r_2'$ and $\mathcal{KGC}$ inputs $\alpha, t_1, t_2, t_3, t_4, \hat{r}_1, \mathit{open}_{\hat{r}_1}, \hat{r}_2, \mathit{open}_{\hat{r}_2}, \mathit{open}_{x_0}, \mathit{open}_{x_1}, \mathit{open}_{x_2}$. If $C_{u_3} = \mathsf{Commit}(u_3, \mathit{open}_{u_3})$, $C_{\hat{r}_1} = \mathsf{Commit}(\hat{r}_1, \mathit{open}_{\hat{r}_1})$, and $C_{\hat{r}_2} = \mathsf{Commit}(\hat{r}_2, \mathit{open}_{\hat{r}_2})$ the output of $\mathcal{KGC}$ is

$$\begin{aligned} x_0 &= (\hat{r}_1 r_1' t_1 t_2 + \hat{r}_2 r_2' t_3 t_4) + u_0 \pmod p \ , \\ x_1 &= -(u_3/r_1' \cdot \alpha t_2) + u_1 \pmod p \ , \\ x_2 &= -(u_3/r_1' \cdot \alpha t_1) + u_2 \pmod p). \end{aligned}$$

   Provided that $\mathcal{KGC}$ does not abort at that moment, $\mathcal{U}$ obtains $C_{x_0} = \mathsf{Commit}(x_0, \mathit{open}_{x_0})$, $C_{x_1} = \mathsf{Commit}(x_1, \mathit{open}_{x_1})$ and $C_{x_2} = \mathsf{Commit}(x_2, \mathit{open}_{x_2})$ as output. Otherwise, both parties output $\perp$. In Sect. 4.3 we show how to efficiently realise such a protocol.
2. $\mathcal{U}$ computes $ID' = H_2(id)^{u_3}$, where u_3 is a blinding value, and sends ID' to $\mathcal{KGC}$. $\mathcal{U}$ proves that the identity in ID' corresponds to C_{id} and that ID' is well-formed using C_{u_3}. $\mathcal{KGC}$ returns $\perp$ if the proof fails. Details about this proof of knowledge can be found in Appendix A.
3. $\mathcal{KGC}$ computes

$$sk_{id}' = (h^{x_0}, h^{x_1} ID'^{-\hat{r}_1 t_2}, h^{x_2} ID'^{-\hat{r}_1 t_1}, ID'^{-\hat{r}_2 t_4}, ID'^{-\hat{r}_2 t_3}).$$

4. $\mathcal{KGC}$ sends the blinded key $sk_{id}' = (d_0', d_1', d_2', d_3', d_4')$ to $\mathcal{U}$, and engages in a proof of knowledge that it is correctly constructed. The proof assures $\mathcal{U}$ that $\mathcal{KGC}$'s chosen values $\hat{r}_1, \hat{r}_2, \mathit{open}_{\hat{r}_1}, \mathit{open}_{\hat{r}_2}, t_1, t_2, t_3, t_4, x_0, x_1, x_2, \mathit{open}_{x_0}, \mathit{open}_{x_1}, \mathit{open}_{x_2}$ correspond to sk_{id}' and to the commitments $C_{\hat{r}_1}$, $C_{\hat{r}_2}$, C_{x_0}, C_{x_1} and C_{x_2} (see Appendix A). If the proof fails, $\mathcal{U}$ returns $\perp$. Otherwise, she computes

$$sk_{id} = (d_0, d_1, d_2, d_3, d_4) = (d_0' h^{-u_0}, (d_1' h^{-u_1})^{r_1'/u_3}, (d_2' h^{-u_2})^{r_1'/u_3}, d_3'^{\,r_2'/u_3}, d_4'^{\,r_2'/u_3}) \ .$$

Theorem 2. *The* IBEBlindExtract *protocol provides a* leak-free *and* selective-failure blind *committed blind extraction protocol for the adaptive identity secure anonymous IBE scheme.*

Proof. Leak freeness: Note that the simulator $\mathcal{S}$ can rewind an instance of the adversary $\mathcal{A}$ that he runs internally. He simulates the communication between the distinguisher $\mathcal{D}$ and $\mathcal{A}$ by passing $\mathcal{D}$'s input to $\mathcal{A}$ and $\mathcal{A}$'s output to $\mathcal{D}$.

In the two party protocol $\mathcal{S}$ can provide random input. Using rewinding techniques, $\mathcal{S}$ extracts $\mathcal{A}$'s input r_1', r_2', and u_0, u_1, u_2, u_3 to the two party computation protocol. In the next step of the blind issuing protocol $\mathcal{A}$ must send $ID' = H_2(id)^{u_3}$ together with a proof of knowledge of a correct representation of ID' and C_{id}. $\mathcal{S}$ uses its rewinding access to $\mathcal{A}$ in order to also extract id, and open_{id}.

Next $\mathcal{S}$ submits id, $open_{id}$ to $\mathcal{O}_{\mathsf{IBEExtract}}$ to obtain a valid secret key $sk_{id} = (d_0, d_1, d_2, d_3, d_4)$. $\mathcal{S}$ returns $(d_0{\cdot}h^{u_0}, d_1^{u_3/r_1'}h^{u_1}, d_2^{u_3/r_1'}h^{u_2}, d_3^{u_3/r_2'}, d_4^{u_3/r_2'})$ to $\mathcal{A}$. These values are distributed in the same way as in $\mathsf{IBEBlindExtract}$.

Selective-failure blindness: $\mathcal{A}$ provides $params$, and two identities id_0, id_1. The game chooses a random bit b. $\mathcal{A}$ is given commitments $C_b = \mathsf{Commit}(id_b, open_b)$ and $C_{1-b} = \mathsf{Commit}(id_{1-b}, open_{1-b})$. $\mathcal{A}$ has blackbox access to two oracles $\mathcal{U}(params, id_{1-b}, open_{1-b})$ and $\mathcal{U}(params, id_b, open_b)$.

Note that once an oracle $\mathcal{U}$ is activated, $\mathcal{A}$ can run a two-party protocol with the oracle, the result of which are three randomly distributed values in Z_p (x_0, x_1, x_2). In the next step, the oracle provides a randomly distributed value in G_2 (ID'), to $\mathcal{A}$. Then the oracle performs a zero-knowledge proof with $\mathcal{A}$.

Suppose that $\mathcal{A}$ runs one or both of the oracles up to this point. Up to now the distributions of the two oracles are computationally indistinguishable. (Otherwise we could break the security of the two party computation, the hiding property of the commitment scheme or the witness indistinguishability of the zero-knowledge proof. The latter is implied by the zero-knowledge property of the proof system.)

$\mathcal{A}$ must provide values $(d_0', d_1', d_2', d_3', d_4')$ and a proof that these values were correctly computed. We can assume that $\mathcal{A}$ chooses these values using an arbitrary complex strategy. We show that any adversary $\mathcal{A}$ can predict the output sk_i of $\mathcal{U}$ without further interaction with the oracles:

1. $\mathcal{A}$ does the proof of Step 4 internally with itself. If the proof fails, it records $sk_0 = \perp$. Otherwise, the adversary temporarily records sk_0=$\mathsf{IBEExtract}(params, msk, id_0)$.
2. In turn, $\mathcal{A}$ generates different $(d_0', d_1', d_2', d_3', d_4')$ and executes a second proof of knowledge (again internally), now for the second oracle. It performs the same checks and recordings for sk_1 and id_1.
3. Finally the adversary predicts (sk_0, sk_1), if both $sk_0 \neq\perp$ and $sk_1 \neq\perp$; $(\epsilon, \perp)$, if only $sk_1 =\perp$; $(\perp, \epsilon)$, if only $sk_0 =\perp$; and $(\perp, \perp)$, if $sk_0 = sk_1 =\perp$.

These predictions result in the same distributions as that returned by the oracle, as the same checks are performed. Moreover, note that for the case that keys are returned by the game they are in both cases equally distributed random keys because of the random values r_1' and r_2' contributed by the oracles.

4.3 Two-Party Protocol for Modulo Arithmetics

The protocol uses a public key additive homomorphic encryption scheme with encryption and decryption functions HEnc and HDec, such that the following hold: $\mathsf{HEnc}(x) \otimes y = \mathsf{HEnc}(xy)$ and $\mathsf{HEnc}(x) \oplus \mathsf{HEnc}(y) = \mathsf{HEnc}(x + y)$. In addition, the encryption should be verifiable [36], meaning it should allow for efficient proofs of knowledge about the encrypted content. The key pair is generated by the $\mathcal{KGC}$ and is made available to $\mathcal{U}$.

We describe an efficient committed two-party computation protocol for computing algebraic terms with addition and multiplication modulo a prime p that generalises ideas presented in [37]. The round complexity of the protocol is 3 if non-interactive proofs of knowledge are used and 12 if interactive proofs of knowledge are used.[5]

[5] The round complexity can be reduced by interleaving the proofs and piggybacking some of the messages.

Let $x_1, \ldots, x_N, open_{x_1}, \ldots, open_{x_N} \in Z_p$ and $y_1, \ldots, y_M, open_{y_1}, \ldots, open_{y_M} \in Z_p$ be the secret input variables and openings of $\mathcal{U}$ and $\mathcal{KGC}$ respectively and let $C_{x_1}, \ldots, C_{x_N}$ and $C_{y_1}, \ldots, C_{y_M}$ be public commitments to the x_i and y_i. We provide a protocol for computing the multivariate polynomial $\sum_{\ell=1}^{L} a_\ell \prod_{n=1}^{N} x_n^{u_{\ell n}} \prod_{m=1}^{M} y_m^{v_{\ell m}}$ where $u_{11}, \ldots, u_{LN}, v_{11}, \ldots, v_{LM} \in \{0, 1\}$ and $a_\ell \in Z_p$ are publicly known values.

The parties can do parts of the computation locally: $\mathcal{U}$ sets $X_\ell = a_\ell \prod_{n=1}^{N} x_n^{u_{\ell n}} \mod p$ and $\mathcal{KGC}$ sets $Y_\ell = \prod_{m=1}^{M} y_m^{v_{\ell m}} \mod p$. To prove that the computation was done correctly $\mathcal{U}$ computes commitment $C_{X_\ell} = \mathsf{Commit}(X_\ell, open_{X_\ell})$ and $\mathcal{KGC}$ computes $C_{Y_\ell} = \mathsf{Commit}(Y_\ell, open_{Y_\ell})$.

The parties can complete the computation using homomorphic encryption as described in the following protocol (The message space of the homomorphic encryption needs to be at least $2^k \ell p^2$). The $\bigoplus$ operator denotes the homomorphic addition of multiple ciphertexts.

$$\begin{array}{rcl}
\mathcal{U} & \longleftrightarrow & \mathcal{KGC} \\
 & \xleftarrow{\{e_\ell\}_{\ell=1}^{L}} & \{e_\ell = \mathsf{HEnc}(Y_\ell)\}_{\ell=1}^{L} \\
e_x = \left(\bigoplus_{\ell=1}^{L} (e_\ell \otimes X_\ell)\right) & \overset{\mathsf{PK}_1}{\longleftrightarrow} & \\
\oplus(\mathsf{HEnc}(r) \otimes p) & \xrightarrow{e_x} & \\
 & \overset{\mathsf{PK}_2}{\longleftrightarrow} & \\
 & & x = Dec(e_x) \\
 & & open_x \leftarrow Z_p \\
 & \xleftarrow{C_x} & C_x = g^x h^{open_x} \\
 & \overset{\mathsf{PK}_3}{\longleftrightarrow} &
\end{array}$$

The $\mathcal{KGC}$ encrypts each Y_ℓ and sends it to the user. The proof PK_1 assures $\mathcal{U}$ that C_{Y_ℓ} was computed correctly using the values in commitments C_{y_i} and that the e_ℓ are encryptions of the values committed to in the C_{Y_ℓ}.

Next, $\mathcal{U}$ computes the encrypted result. The term $r \cdot p$, $0 < r < (2^k - 1)\ell p$ is added to avoid possible modulo overflows from revealing any statistically significant information about $\mathcal{U}$'s input. $\mathcal{U}$ proves to $\mathcal{KGC}$ in PK_2 that C_{X_ℓ} was computed correctly using the values in commitments C_{x_i} and that e_x was computed correctly using the values committed to in the C_{Y_ℓ}.

As a last step, $\mathcal{KGC}$ decrypts e_x, does a single modulo p reduction to obtain the result of the computation, and commits to the result in commitment C_x. In PK_3 $\mathcal{KGC}$ proves to the user that C_x contains the same value modulo p as encrypted in e_x. For details on how to do the proofs $\mathsf{PK}_1, \ldots, \mathsf{PK}_3$ we refer to [29,38]. An efficient implementation of such a protocol is presented in [37] using the Paillier homomorphic encryption scheme [39].

4.4 Transformation to PEOKS

We construct a suitable PEKS scheme for our application scenario using the anonymous IBE scheme presented in Sect. 4.1. We follow a generic transformation by Abdalla et al. [30] from IBE to PEKS. The transformation takes as input the algorithms Π of a secure IBE scheme and returns a PEKS scheme $\Upsilon = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$. Note that our scheme differs from preexisting schemes as Test returns a secret message in case of a match:

$\mathsf{KeyGen}(1^k)$ runs algorithm $\mathsf{IBESetup}(1^k)$ and returns the key pair (A_{pub}, A_{priv}), the $(params, msk)$ of the IBE scheme.

$\mathsf{PEKS}(A_{pub}, W, msg)$ takes as input public key A_{pub}, keyword W and message msg. It outputs a searchable encryption S_W of message msg under keyword W as follows:

1. Generate a random value $C_2 \in \{0,1\}^k$.
2. Compute $C_1 = \mathsf{IBEEnc}(A_{pub}, W, msg \| C_2)$.
3. Output the tuple $S_W = (C_1, C_2)$.

$\mathsf{Trapdoor}(A_{pub}, A_{priv}, W)$ outputs a trapdoor $T_W = \mathsf{IBEExtract}(A_{pub}, A_{priv}, W)$ that enables a search for the keyword W.

$\mathsf{Test}(A_{pub}, S_W, T_{W'})$ parses S_W as (C_1, C_2) and computes $M = \mathsf{IBEDec}(A_{pub}, T_{W'}, C_1)$. If $M = msg \| C_2$, it outputs the message msg encoded in S_W; if there is no match, it outputs $\perp$.

In order to achieve the oblivious property in our PEOKS scheme, we extend algorithm Trapdoor to a BlindTrapdoor protocol. Our PEOKS scheme is thus composed of the algorithms Υ of the PEKS scheme, a secure commitment scheme Commit, and a BlindTrapdoor protocol where

$\mathsf{BlindTrapdoor}(\mathcal{U}(A_{pub}, W, open_W), \mathcal{TGE}(A_{pub}, A_{priv}, C))$ generates a trapdoor T_W for a keyword W by running protocol $\mathsf{IBEBlindExtract}(\mathcal{U}(A_{pub}, W, open_W), \mathcal{KGC}(A_{pub}, A_{priv}, C))$.

5 Authorised Private Searches on Public Key Encrypted Data

We describe a public key encrypted database that enables oblivious searches. Our construction is similar to the audit log presented in [3]. Each data record is encrypted using a fresh random symmetric key and associated with several searchable encryptions. Each searchable encryption is generated using input of a keyword that describes the content of the record, and a secret message that contains the symmetric key. Once an investigator obtains a trapdoor that matches a searchable encryption (i.e., both were computed on input the same keyword), she is returned the symmetric key that allows her to decrypt the record.

In constructing authorised private searches, we ensure that neither the keywords of interest for the investigator nor the search results are revealed. For the first property, we employ the PEOKS scheme. The investigator runs protocol BlindTrapdoor with the trapdoor generation entity ($\mathcal{TGE}$) in order to retrieve a trapdoor for a committed keyword in a blind manner. The committed blind extraction allows the $\mathcal{TGE}$ to construct policies detailing the data that a particular investigator can obtain. To enforce these restrictions, the $\mathcal{TGE}$ requires the investigator to prove in zero-knowledge that the keyword used to compute the commitment belongs to a certain language. We also consider a party (such as a judge) in charge of deciding which keywords can be utilized by the investigator, and describe how the investigator obtains a search warrant from the judge and shows it to the $\mathcal{TGE}$. The judge and the $\mathcal{TGE}$ are only involved in providing search warrants and trapdoors respectively, and can remain off-line when not required to perform these tasks.

To obscure the search results, we describe a data structure that allows the use of a PIR scheme and that integrates concepts from [40] to improve the efficiency of the searches[6]. Since the PIR queries are made over encrypted data, we also ensure that the investigator does not obtain any information about data described by keywords for which she was not authorised to retrieve a trapdoor. It should also be noted that, due to the public key setting, the database only stores the public key of the PEOKS scheme. Thus, in the event that it gets corrupted, records encrypted prior to corruption remain secure (forward secrecy).

Details on data storage. We describe a data structure in which only one searchable encryption per keyword is computed, while still allowing each data record to be described by several keywords. Once the investigator finds the searchable encryption that matches her trapdoor, she receives the information needed to decrypt all the data records described by the corresponding keyword. This mechanism of data storage allows for an efficient search (not all the searchable encryptions need to be tested) and is privacy enhancing in so far as it hides the number of keywords that describe a record from the investigator.

We use encrypted linked lists and store the encrypted nodes at random positions in the PIR database to hide which node belongs to which linked list, as introduced in [40]. We construct one linked list per keyword. Each node in the linked list contains the information required to retrieve and decrypt one record associated with the keyword. A node contains a PIR query index P_R for the data record and the key K_R used to encrypt the record. It also stores a PIR query index to the next node on the list, and the key used to encrypt it. To encrypt the nodes and the records of data, we employ a symmetric encryption algorithm Enc.

When the data holder adds a keyword W for which no searchable encryption has previously been computed, he chooses a symmetric key K_{N_1} and runs algorithm PEKS $(A_{pub}, W, K_{N_1}||P_{N_1})$ to compute the searchable encryption. P_{N_1} is the PIR query index to the first node of the list and K_{N_1} is the symmetric key used to encrypt this node. He then builds the node $N_1 = (P_R, K_R, P_{N_2}, K_{N_2})$, computes $\mathsf{Enc}(K_{N_1}, N_1)$, and stores the node in the position given by P_{N_1}. Finally, he deletes P_{N_1} and K_{N_1} from his memory but keeps values P_{N_2} and K_{N_2}. P_{N_2} and K_{N_2} are the PIR query index and the key for the next node in the list. In position P_{N_2} a flag is stored to indicate the end of the list.

When the data holder chooses this keyword to describe another record R', it builds the second node $N_2 = (P_{R'}, K_{R'}, P_{N_3}, K_{N_3})$, runs $\mathsf{Enc}(K_{N_2}, N_2)$, and stores the encrypted node in the position given by P_{N_2}. It deletes P_{N_2} and K_{N_2} from his memory but keeps P_{N_3} and K_{N_3} to facilitate adding another node to the list. He also stores the flag in P_{N_3}. This iterative procedure is applied as many times as required.

If a data record is described by several keywords, one node per keyword is generated and stored in its corresponding linked list. All these nodes contain the same PIR query index to the data record and the same key used to encrypt the record.

[6] The amount of PIR queries may give some indication about the number of records retrieved. This information can be hidden through dummy transactions up to an upper limit on the number of matching records.

Authorizing and performing private searches. An investigator that wants to search on the encrypted database follows the procedure:

1. The investigator requests authorisation from the judge to perform a search on a given database for a particular keyword W. Assuming the investigator holds the relevant credentials, the judge grants a warrant. In practice, this means that the investigator runs a protocol GetCredential with the judge, which returns to the investigator a credential $cred$ with attribute W from the judge.
2. The investigator requests a trapdoor from the $\mathcal{TGE}$. This is a three step process:
   (a) The investigator has a commitment $C = Commit(W, open_W)$ to the keyword W for which she wants to receive a trapdoor, and sends C to the $\mathcal{TGE}$.
   (b) The investigator and the $\mathcal{TGE}$ run an interactive protocol, ShowCredential. This verifies the validity of the credential presented by the investigator and the claim that the keyword used to compute the commitment is the same as the keyword contained in the credential's attributes.
   (c) The investigator and the $\mathcal{TGE}$ execute the BlindTrapdoor protocol, with investigator input $A_{pub}, W, open_W$ and $\mathcal{TGE}$ input A_{pub}, A_{priv}, C. The protocol returns no output to the $\mathcal{TGE}$, and a trapdoor T_W to the investigator.
3. The investigator downloads the list of PEKS elements for all the keywords.
4. If an investigator performs a successful Test for a PEKS element (using the correct trapdoor), the algorithm returns the key and PIR query index pair that correspond to the first node of the list. The investigator uses the PIR scheme to retrieve the node and the first record. As above, each node returns sufficient information to link to the next node, until all data related to the keyword have been returned.

Remark. GetCredential and ShowCredential can be implemented using conventional signatures: during GetCredential the judge signs $C = Commit(W, open_W)$ to create the credential $cred$ (a signature on C); in the ShowCredential protocol the investigator sends $cred$ together with C and the $\mathcal{TGE}$ verifies the signature. More sophisticated credential protocols [41,42,43,44,45] allow the implementation of more complex policies, such as, e.g., the *time restricted searches* described below.

Time restricted searches. In PEKS, the notion of temporary keyword search [30] implies that searchable encryptions and trapdoors are related to a specific time period in such a way that, even if the keyword used to compute them is the same, they do not match if the time period is different. The simplest way to build PEKS with temporary keyword search is to concatenate keywords W and time periods t when computing searchable encryptions and trapdoors. When applying this solution to our database, multiple linked lists are generated for the same keyword concatenated, each corresponding to a different time frame.

This function is useful to provide searches in which the investigator is allowed to obtain all records described by a specific keyword that were stored within a restricted period of time. In this case, the credential issued by the judge is extended to contain two additional attributes t_1 and t_2 corresponding to a start time-stamp and end time-stamp which limit the period of an investigation. When showing the credential to the $\mathcal{TGE}$, the investigator computes a commitment to $W||t$ and also proves that $t_1 \leq t \leq t_2$. This can for instance be done using the techniques described in [32].

6 Conclusion and Future Work

We have defined and implemented a searchable encryption scheme, PEOKS, that allows for oblivious searches on public key encrypted data. For this purpose, we have extended the PEKS primitive by adding blind trapdoor extraction with committed keywords. In order to implement PEOKS, we have defined committed blind anonymous IBE and we have provided a construction of such a scheme. Finally, we applied PEOKS to build a public key encrypted database that permits authorised private searches.

As future work we leave the design of a blind key extraction protocol secure under concurrent execution. Furthermore, more efficient anonymous identity-based encryption schemes with more light weight key derivation protocols would translate directly into highly efficient PEOKS. Unfortunately, the scheme in [8] does not seem fit for our purposes as it uses stateful randomness in the secret key generation phase.

We observe that in a practical application it is likely that an investigator would want to search for data described by a predicate formed by conjunctions and disjunctions of keywords. Future work would focus on using attribute-hiding predicate encryption [46] to build a scheme that permits oblivious searches on encrypted data by specifying predicates of keywords.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 216483. It has also been funded by a Science Foundation of Ireland Basic Research Grant, project number 04/BR/CS0692.

References

1. Directive 2006/24/ec of the european parliament and of the council. Official Journal of the European Union (April 2006)
2. Ostrovsky, R., Skeith III, W.E.: Private searching on streaming data. J. Cryptology 20(4), 397–430 (2007)
3. Waters, B.R., Balfanz, D., Durfee, G., Smetters, D.K.: Building an encrypted and searchable audit log. In: NDSS, The Internet Society (2004)
4. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords (1998)
5. Ogata, W., Kurosawa, K.: Oblivious keyword search. J. Complexity 20(2-3), 356–371 (2004)
6. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Proceedings of Eurocrypt, vol. 4 (2004)
8. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
9. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)

10. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
11. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
12. Naccache, D.: Secure and practical identity-based encryption. Information Security, IET 1(2), 59–64 (2007)
13. Chatterjee, S., Sarkar, P.: Trading time for space: Towards an efficient ibe scheme with short(er) public parameters in the standard model. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 424–440. Springer, Heidelberg (2006)
14. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
15. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. J. ACM 50(6), 852–921 (2003)
16. Goldwasser, S., Kalai, Y.T.: On the (in)security of the fiat-shamir paradigm. In: FOCS, p. 102. IEEE Computer Society, Los Alamitos (2003)
17. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
18. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. Cryptology ePrint Archive, Report 2008/474 (2008), `http://eprint.iacr.org/`
19. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS, pp. 41–50 (1995)
20. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
21. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)
22. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
23. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number n is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
24. Camenisch, J.L.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich (1998)
25. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997)
26. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
27. Damgård, I.: Concurrent zero-knowledge is easy in practice. Available online at Theory of Cryptography Library (June 1999)
28. Damgård, I.: On σ-protocols (2002), `http://www.daimi.au.dk/~ivan/Sigma.ps`
29. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich (March 1997)

30. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
31. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
32. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
33. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
34. Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006), http://eprint.iacr.org/
35. Yao, A.C.: Protocols for secure computations. In: Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 160–164 (1982)
36. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000)
37. Camenisch, J., Koprowski, M., Warinschi, B.: Efficient blind signatures without random oracles. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 134–148. Springer, Heidelberg (2005)
38. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
39. Paillier, P.: Public-key cryptosystems based on composite residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–239. Springer, Heidelberg (1999)
40. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 79–88. ACM, New York (2006)
41. Brands, S.: Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy. PhD thesis, Eindhoven Inst. of Tech. The Netherlands (1999)
42. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
43. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
44. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
45. Bangerter, E., Camenisch, J., Lysyanskaya, A.: A Cryptographic Framework for the Controlled Release Of Certified Data. In: 12th International Workshop on Security Protocols 2004, Cambridge, England, April 26, 2004, pp. 20–42. Springer, Heidelberg (2004)
46. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. Cryptology ePrint Archive, Report 2007/404 (2007), http://eprint.iacr.org/

A Proofs of Knowledge of Correct Key Derivation

Proof for Step 2. The $\mathcal{KGC}$ has commitment C_{u_3} to u_3, and C_{id} to the user's choice of id. In Step 2 of our $\mathsf{IBEBlindExtract}$ protocol the user does the following proof of knowledge to convince the $\mathcal{KGC}$ that her message ID' is well formed:

$$\begin{aligned}
PK\{(&id_1, \ldots, id_n, u_3, id_1 \cdot u_3, \ldots, id_n \cdot u_3, open_{id}, open_{u_3},\\
&open_{id} \cdot u_3) : C_{id} = (\prod_{i=1}^{n} (h_0^{2^{l(i-1)}})^{id_i}) h_1^{open_{id}} \wedge && (1)\\
&\bigwedge_{i=1}^{n} 0 \leq id_i < 2^l \wedge C_{u_3} = h_0^{u_3} h_1^{open_{u_3}} \wedge && (2)\\
&1 = C_{id}^{u_3} (\prod_{i=1}^{n} ((1/h_0)^{2^{l(i-1)}})^{id_i \cdot u_3})(1/h_1)^{open_{id} \cdot u_3} \wedge && (3)\\
&ID' = h_0^{u_3} \prod_{i=1}^{n} h_i^{id_i \cdot u_3}\} \ . && (4)
\end{aligned}$$

The user proves that id is correctly encoded in ID'. This is done in two steps: (1) and (2) prove that id is correctly split up into its n components id_i; (3) and (4) prove that ID' contains a blinded version of $H_2(id)$. This requires to prove multiplicative relations between u_3 and the id_i in (3).

Proof for Step 4. The user has commitments $C_{\hat{r}_1}$, $C_{\hat{r}_2}$ and C_{x_0}, C_{x_1}, and C_{x_2}. In Step 4 of our $\mathsf{IBEBlindExtract}$ protocol the $\mathcal{KGC}$ does the following proof of knowledge to convince the user that the blinded key $(d'_0, d'_1, d'_2, d'_3, d'_4)$ it returns is well formed:

$$\begin{aligned}
PK\{(&\hat{r}_1, \hat{r}_2, open_{\hat{r}_1}, open_{\hat{r}_2}, t_1, t_2, t_3, t_4, x_0, x_1, x_2, open_{x_0},\\
&open_{x_1}, open_{x_2}, -\hat{r}_1 t_1, -\hat{r}_1 t_2, -\hat{r}_2 t_3, -\hat{r}_2 t_4) :\\
&C_{\hat{r}_1} = h_0^{\hat{r}_1} h_1^{open_{\hat{r}_1}} \wedge C_{\hat{r}_2} = h_0^{\hat{r}_2} h_1^{open_{\hat{r}_2}} \wedge v_1 = g^{t_1} \wedge v_2 = g^{t_2} \wedge v_3 = g^{t_3} \wedge\\
&v_4 = g^{t_4} \wedge C_{x_0} = h_0^{x_0} h_1^{open_{x_0}} \wedge C_{x_1} = h_0^{x_1} h_1^{open_{x_1}} \wedge C_{x_2} = h_0^{x_2} h_1^{open_{x_2}} \wedge\\
&1 = (1/v_1)^{\hat{r}_1} (1/g)^{-\hat{r}_1 t_1} \wedge 1 = (1/v_2)^{\hat{r}_1} (1/g)^{-\hat{r}_1 t_2} \wedge 1 = (1/v_3)^{\hat{r}_2} (1/g)^{-\hat{r}_2 t_3} \wedge\\
&1 = (1/v_4)^{\hat{r}_2} (1/g)^{-\hat{r}_2 t_4} \wedge d'_0 = h^{x_0} \wedge d'_1 = h^{x_1} ID'^{-\hat{r}_1 t_2} \wedge d'_2 = h^{x_2} ID'^{-\hat{r}_1 t_1} \wedge\\
&d'_3 = ID'^{-\hat{r}_2 t_4} \wedge d'_4 = ID'^{-\hat{r}_2 t_3}\} \ .
\end{aligned}$$

By means of this proof the $\mathcal{KGC}$ demonstrates to the user that it uses the correct values for $x_0, x_1, x_2, t_1, t_2, t_3, t_4, \hat{r}_1, \hat{r}_2$ when it computes $(d'_0, d'_1, d'_2, d'_3, d'_4)$. The proof involves proving the multiplicative relations $-\hat{r}_1 t_1, -\hat{r}_1 t_2, -\hat{r}_2 t_3, -\hat{r}_2 t_4$ between $t_1, t_2, t_3, t_4, \hat{r}_1, \hat{r}_2$.

Anonymous Hierarchical Identity-Based Encryption with Constant Size Ciphertexts

Jae Hong Seo[1,⋆], Tetsutaro Kobayashi[2], Miyako Ohkubo[2],
and Koutarou Suzuki[2]

[1] Department of Mathematical Sciences and ISaC-RIM, Seoul National University,
Seoul, Korea
jhsbhs@gmail.com
[2] NTT Information Sharing Platform Labs, Tokyo, Japan
{kobayashi.tetsutaro,ookubo.miyako,suzuki.koutarou}@lab.ntt.co.jp

Abstract. We propose an anonymous Hierarchical Identity-Based Encryption (anonymous HIBE) scheme that has constant size ciphertexts. This means the size of the ciphertext does not depend on the depth of the hierarchy. Moreover, our scheme achieves the lowest computational cost because during the decryption phase the computational cost of decryption is constant. The security can be proven under reasonable assumptions without using random oracles because it is based on the composite order bilinear group. Our scheme achieves selective-ID security notion.

1 Introduction

Identity-Based Encryption (IBE) is a topic of focus as a useful technique. Studies are proceeding in various directions, and numerous applications using IBE have been presented. A searchable encryption scheme has been discussed. At first, schemes allowed keywords that were not encrypted. However, in such schemes, an anonymous request for a keyword cannot be satisfied by using simple IBE schemes. To provide this function, anonymous IBE was proposed. The anonymous IBE scheme provides a very useful function, i.e., anonymity of ID. An anonymous IBE ciphertext does not leak any information about the receiver's identity. Such a useful property can be applied to keyword searchable encryption while maintaining anonymity of the keyword [1,5,11].

Anonymous Hierarchical Identity-Based Encryption (anonymous HIBE), which handles IDs hierarchically maintaining the anonymity of an ID and keys, can be delegated even if a blinding ID is used. Anonymous HIBE allows some protocols using anonymous HIBE to be extended; for example, by applying keyword-searchable encryption, keywords can be treated hierarchically while maintaining anonymous keyword information.

⋆ This work was done while the first author was visiting NTT Information Sharing Platform Labs. He was supported by the Science Research Center(SRC) program of the Korea Science and Engineering Foundation (KOSEF) with a grant funded by the Korean govement (Ministry of Science and Technology (MOST), Grant R11-2007-035-01002-0).

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 215–234, 2009.

1.1 Related Works: ID-Based Encryption Algorithms

After Horwitz and Lynn defined the notion of Hierarchical ID-Based Encryption (HIBE)[18], many efficient and provably secure HIBE schemes were proposed. Gentry and Silverberg proposed an efficient and secure HIBE scheme, that achieves full-ID CPA (chosen plaintext attack) security; however, it was proven with a random oracle (GS-HIBE)[17]. Canetti, Halevi and Katz[13] suggested a weaker security notion, called selective-ID, and they also proposed a selective-ID secure HIBE without using random oracles; however, their scheme is an inefficient one. An efficient and selective-ID secure HIBE scheme in the standard model was proposed by Boneh and Boyen (BB-HIBE) [2]. However the ciphertext of the BB-HIBE scheme is depends on the depth of the hierarchy. To improve the efficiency, HIBE with constant size ciphertexts was presented by Boneh, Boyen and Goh (BBG-HIBE)[3]. In their scheme, a private key can be delegated while maintaining a constant ciphertext size. BBG-HIBE was proven without using random oracles, and it achieves selective-ID security. Full-ID secure schemes that do not use random oracles were presented by Waters[23], Chatterjee and Sarkar [15]. All of the above mentioned HIBE schemes were proposed assuming that IDs are known to everyone, so they cannot provide anonymity of ID. We call such HIBE schemes non-anonymous HIBE schemes.

On the other hand, the concepts of anonymous IBE and anonymous HIBE were shown by Abdalla et al. [1], and formal definitions of them were also given in that paper. However, a concrete construction of anonymous HIBE was not proposed. A concrete constructions of anonymous IBE in the standard model was proposed by Gentry[16] and a concrete construction of anonymous HIBE was proposed by Boyen and Waters (BW-HIBE)[12]. Both of these schemes were proposed in the standard model and are selective-ID CPA secure, which can be proven without using random oracles. Shi and Waters proposed a delegatable hidden-vecor encryption (dHVE) whose definition is a generalization of anonymous HIBE, i.e., anonymous HIBE is a special case of the dHVE scheme (SW-dHVE) [22]. The scheme takes a composit order bilinear group, which was introduced by Boneh, Goh, and Nissim [7], to obtain property of anonoymity. However, in the BW-HIBE and SW-dHVE schemes, the ciphertext size depends on the hierarchy depth. The ciphertext size has a great impact on practicality, so while their results are very interesting, efficiency remains an open problem.

1.2 Our Results

Motivation: The size of ciphertext affects the efficiency and feasibility of various applications using HIBE schemes, and if the ciphertext size depends on the depth the hierarchy depth, the efficiency and feasibility of applications also depend on the hierarchy depth. The HIBE scheme with constant size ciphertexts can extend the feasibility and convenience of applications. Additionally, some applications need anonymity of ID. For example, Keyword-search encryption is one of such applications. In keyword-search encryption scheme, each keyword needs a ciphertext, therefore totally the size of ciphertexts for many keywords is significant impact on the efficiency of keyword search. In such a case the size of ciphertext is serious problem. However, none of the previous results for anonymous HIBE could provide constant size ciphertexts.

Contribution: We present an anonymous HIBE scheme with constant size ciphertexts. Our scheme achieves selective-ID CPA security without using random oracles, and is based on a new assumption, the ℓ-composite Diffie-Hellman assumption The details are shown in section 2.3. The technical highlight of our paper is the technique for achieving constant size ciphertexts even though the number of layers is increased in HIBE, keeping anonymity of IDs, and security proofs using game-based proof techniques. The difficulties in devising an efficient anonymous HIBE scheme are constructing a length of ciphertext that is independent of hierarchy depth and maintaining the anonymity of key delegation hierarchical ID. Our idea is effected by BBG-HIBE [3], which provides constant size ciphertexts. However, [3] does not satisfy the requirement of an anonymous ID. To attain ID anonymity, we need a randomizing method, keeping the property of key deligation. Therefore the proposed scheme takes a composite order bilinear group, and is using a technique improved upon that of in [11]. The HVE scheme in [11] is not anonymous HIBE; it is a scheme for keyword searchable encryption. However, the technique for providing keyword anonymity offers us a key idea for solving key delegation for anonymous HIBE.

There were two anonymous HIBE schemes, BW-HIBE and SW-dHVE[1] before our HIBE scheme. Comparing our construction with these previous two anonymous HIBE schemes, we see that the ciphertexts in both those schemes are $O(L)$ group elements, and private keys are $O(L^2)$ group elements, where L is the maximum hierarchy depth. In constract, our scheme uses only four group elements for a ciphertext, and the private key uses $O(L)$ group elements.

2 Background

2.1 Security Models

We briefly explain the informal security notions of anonymous HIBE. The formal security definitions may be found in the literature [6,1]. We use a weaker notion of security introduced by [13,14] in which the adversary commits ahead of time to the public *params* that it will attack; i.e., we use the selective security notion.

Semantic security(IND-sID(indistinguishability against selective identity)): The adversary outputs target identity ID* before public parameters are generated. It can make a private key derivation query for ID such that the ID is not a prefix of or equal to target identity ID*. It publishes target message Msg*. No poly-time adversary can distinguish between a ciphertext of target message Msg* with target identity ID* and a ciphertext of random message with target identity ID*.

[1] The dHVE and anonymous HIBE schemes were proposed as anonymous HIBE schemes [22]. Anonymous HIBE has a flaw, but since we can consider dHVE as an anonymous HIBE scheme, we compare dHVE with our scheme.

Anonymity(ANON-sID(anonymity against selective identity)): The adversary outputs target identity ID* before public parameters are generated. It can make private key derivation query for ID such that the ID is not a prefix of or equal to target identity ID*. It publishes target message Msg*. No poly-time adversary can distinguish between a ciphertext of target message Msg* with target identity ID* and a ciphertext of target message Msg* with random identity.

2.2 Bilinear Groups of Composite Order

We will use a bilinear group of composite order pq. Bilinear groups of composite order were introduced by Boneh, Goh, Nissim [7].

Let $\mathcal{G}$ be a group generation algorithm that takes security parameter 1^λ as input and outputs tuple $(p, q, \mathbb{G}, \mathbb{G}_T, e)$ where p and q are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order $n = pq$, and e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear map; i.e., e satisfies the following properties:

bilinear: For $\forall g_1, h_1 \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$, $e(g_1^a, h_1^b) = e(g_1, h_1)^{ab}$.
non-degenerate: For generator g_1 of $\mathbb{G}$, $e(g_1, g_1)$ generates $\mathbb{G}_T$.

We assume that group multiplication in $\mathbb{G}$, $\mathbb{G}_T$ and bilinear map e are all polynomial time computable in λ. Furthermore, we assume that descriptions of $\mathbb{G}$ and $\mathbb{G}_T$ contain generators as well as identity elements $1_\mathbb{G}$, $1_{\mathbb{G}_T}$ of $\mathbb{G}$ and $\mathbb{G}_T$, respectively. If there is no confusion, we use 1 for identity irrespective of the group.

We will use the notation $\mathbb{G}_p$ and $\mathbb{G}_q$ to denote the subgroups of $\mathbb{G}$ of order p and q, respectively, and we will use the notation $\mathbb{G}_{T,p}$ and $\mathbb{G}_{T,q}$ to denote subgroups of $\mathbb{G}_T$ of order p and q, respectively. Then $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q$ and $\mathbb{G}_T = \mathbb{G}_{T,p} \times \mathbb{G}_{T,q}$. If g_1 is a generator of $\mathbb{G}$, then g_1^q and g_1^p are generators of $\mathbb{G}_p$ and $\mathbb{G}_q$, respectively. We use the notation g_p and g_q to denote generators of $\mathbb{G}_p$ and $\mathbb{G}_q$, respectively.

Note that $e(h_p, h_q) = 1$ for all random elements $h_p \in \mathbb{G}_p$ and $h_q \in \mathbb{G}_q$ because $e(h_p, h_q) = e(g_p^a, g_q^b)$ for some integers a, b, and $e(g_p^a, g_q^b) = e(g_1^{qa}, g_1^{pb}) = e(g_1, g_1)^{pqab} = 1$ for some generator g_1 in $\mathbb{G}$.

2.3 Complexity Assumptions

ℓ-weak Bilinear Diffie-Hellman Inversion* assumption. The ℓ-Bilinear Diffie-Hellman Inversion (ℓ-BDHI) assumption has been used for constructing cryptographic schemes [21,2,3,4]. Boneh, Boyen and Goh introduced a slightly weaker assumption, ℓ-weak BDHI*, denoted by ℓ-wBDHI* to design HIBE with constant size ciphertexts in [4]. Our scheme use Decision ℓ-wBDHI* in bilinear groups of composite order to prove semantic security. We say that group generator $\mathcal{G}$ satisfies the (ϵ, t)-Decision ℓ-wBDHI* assumption if no t-time algorithm has advantage at least ϵ in solving the Decision ℓ-wBDHI* problem in groups generated by $\mathcal{G}$.

$$(prime : \{p, q\}, group : \{\mathbb{G}, \mathbb{G}_T\}, bilinearmap : \{e\}) \xleftarrow{R} \mathcal{G}(\lambda),$$

$$n \leftarrow pq, \quad g_p, h \xleftarrow{R} \mathbb{G}_p, \quad g_q \xleftarrow{R} \mathbb{G}_q, \quad a \xleftarrow{R} \mathbb{Z}_n$$

$$\mathbf{Z} \leftarrow ((n, \mathbb{G}, \mathbb{G}_T, e), g_q, g_p, h, g_p^a, g_p^{a^2}, \cdots, g_p^{a^\ell}),$$

$$T \leftarrow e(g_p, h)^{a^{\ell+1}}, \quad d \xleftarrow{R} \{0, 1\}.$$

Let $T' = T$ if d is 1; otherwise set T' to be a uniformly and independently chosen element from $\mathbb{G}_{T,p}$. We call ($\mathbf{Z}$, T') the challenge pair of the Decision ℓ-wBDHI*. Give the challenge pair to adversary $\mathcal{A}$. Then $\mathcal{A}$ outputs d', and succeeds if $d = d'$. The advantage of $\mathcal{A}$ in solving Devision ℓ-wBDHI* problem in groups generated by $\mathcal{G}$ is $|Pr[\mathcal{A}(\mathbf{Z}, T) = 1] - Pr[\mathcal{A}(\mathbf{Z}, R) = 1]|$, where the probability is over random coins in $\mathcal{G}$, a random choice of $R \in \mathbb{G}_{T,p}$ and the random coins of $\mathcal{A}$.

ℓ-composite Diffie-Hellman assumption. The anonymity of our construction is based on a new complexity assumption that we call the ℓ-composite Diffie-Hellman assumption (ℓ-cDH) in bilinear groups with composite order $n = pq$. We say that group generator $\mathcal{G}$ satisfies the (ϵ, t)-ℓ-cDH assumption if no t-time algorithm has advantage at least ϵ in solving the ℓ-cDH problem in groups generated by $\mathcal{G}$.

$$(prime : \{p, q\}, group : \{\mathbb{G}, \mathbb{G}_T\}, bilinearmap : \{e\}) \xleftarrow{R} \mathcal{G}(\lambda),$$

$$n \leftarrow pq, \quad g_p \xleftarrow{R} \mathbb{G}_p, \quad g_q, R_1, R_2, R_3 \xleftarrow{R} \mathbb{G}_q \quad a, b \xleftarrow{R} \mathbb{Z}_n$$

$$\mathbf{Z} \leftarrow ((n, \mathbb{G}, \mathbb{G}_T, e), g_q, g_p, g_p^a, g_p^{a^2}, \cdots, g_p^{a^\ell}, g_p^{a^{\ell+1}} \cdot R_1, g_p^{a^{\ell+1}b} \cdot R_2),$$

$$T \leftarrow g_p^b \cdot R_3, \quad d \xleftarrow{R} \{0, 1\}.$$

Let $T' = T$ if d is 1; otherwise set T' to be a uniformly and independently chosen element from $\mathbb{G}$. We call ($\mathbf{Z}$, T') the challenge pair of the ℓ-cDH. Give the challenge pair to adversary $\mathcal{A}$. Then $\mathcal{A}$ outputs d', and succeeds if $d = d'$. The advantage of $\mathcal{A}$ in solving the ℓ-cDH problem in groups generated by $\mathcal{G}$ is $|Pr[\mathcal{A}(\mathbf{Z}, T) = 1] - Pr[\mathcal{A}(\mathbf{Z}, R) = 1]|$, where the probability is over random coins in $\mathcal{G}$, a random choice of $R \in \mathbb{G}$ and the random coins of $\mathcal{A}$.

Our assumption holds in a generic model if the factorization of n is hard. According to the Master Theorem [20], in a generic model, if there is an algorithm $\mathcal{A}$ issuing at most q_s instructions and having advantage Adv in the above experiment, then $\mathcal{A}$ can be used to find a non-trivial factor of n with probability at least $Adv - O(q_s^2(\ell + 2)/(n^{1/2}))$. Therefore, if the factorization of n is hard, any polynomial time algorithm $\mathcal{A}$ has a negligible advantage in n.

Bilinear Subset Decision assumption. This assumption is implied by the ℓ-cDH assumption and was introduced by Boneh, Sahai, and Waters [10]. We say that group generator $\mathcal{G}$ satisfies the (ϵ, t)-Bilinear Subset Decision (BSD) assumption if no t-time algorithm has advantage at least ϵ in solving the BSD problem in groups generated by $\mathcal{G}$.

$$(prime : \{p, q\}, group : \{\mathbb{G}, \mathbb{G}_T\}, bilinearmap : \{e\}) \xleftarrow{R} \mathcal{G}(\lambda),$$

$$n \leftarrow pq, \quad g_p \overset{R}{\leftarrow} \mathbb{G}_p, \quad g_q \overset{R}{\leftarrow} \mathbb{G}_q,$$
$$\mathbf{Z} \leftarrow ((n, \mathbb{G}, \mathbb{G}_T, e), g_q, g_p),$$
$$T \overset{R}{\leftarrow} \mathbb{G}_{T,p}, \quad d \overset{R}{\leftarrow} \{0,1\}.$$

Let $T' = T$ if d is 1; otherwise set T' to be a uniformly and independently chosen element from $\mathbb{G}_T$. We call $(\mathbf{Z}, T')$ the challenge pair of the BSD problem. Give the challenge pair to adversary $\mathcal{A}$. Then $\mathcal{A}$ outputs d', and succeeds if $d = d'$. The advantage of $\mathcal{A}$ in solving BSD problem in groups generated by $\mathcal{G}$ is $|Pr[\mathcal{A}(\mathbf{Z}, T) = 1] - Pr[\mathcal{A}(\mathbf{Z}, R) = 1]|$, where the probability is over random coins in $\mathcal{G}$, a random choice of $R \in \mathbb{G}_T$ and the random coins of $\mathcal{A}$.

3 Anonymous HIBE with Constant Size Ciphertexts

In this section, we propose an anonymous hierarchical ID-based encryption with constant size ciphertexts secure under the Decision ℓ-wBDH* assumption and ℓ-cDH assumption. Our construction is based on BBG-HIBE. To attain anonymity, we construct HIBE over a bilinear group with composite order as HVE in [11] which can be considered as an anonymous IBE scheme. All non-random elements of our HIBE scheme are embedded in $\mathbb{G}_p$ or $\mathbb{G}_{T,p}$. A private key consists of only elements in $\mathbb{G}_p$. Public parameters and ciphertexts are blinded by random elements in $\mathbb{G}_q$ or $\mathbb{G}_{T,q}$. Since a pairing result between elements in $\mathbb{G}_p$ and elements in $\mathbb{G}_q$ is 1, blinding factors of ciphertexts are removed by calculating a pairing with a private key in the decryption procedure.

In delegation procedure of the BBG-HIBE scheme, the private key is re-randomized by using public parameters. However, we cannot use public parameters for private key re-randomization in our construction since a private key must be composed of only elements in $\mathbb{G}_p$. If we use public parameters which have blinding factors in $\mathbb{G}_q$ to re-randomize the private key, then the resulting private key will also be blinded by elements in $\mathbb{G}_q$. This private key can not decrypt any ciphertext because the blinding factors of ciphertexts cannot be removed by pairing with the private key in the decryption procedure. Therefore we add a re-randomization subkey, which is composed of elements in $\mathbb{G}_p$, to the private key, and the re-randomization procedure uses not only public parameters but also the re-randomization subkey.

3.1 Construction

Here, we present our HIBE construction.

Setup(λ, L)**:** The setup algorithm generates public system parameters, denoted by *params*, and the corresponding master secret key, denoted by *MK* by using a security parameter and the maximum hierarchy depth L. First, the setup algorithm generates $(p, q, \mathbb{G}, \mathbb{G}_T, e)$, as explained in the section 2.2. Next, it selects random elements

$$g, f, v, h_1, \cdots, h_L, w \in \mathbb{G}_p, \qquad R_g, R_f, R_v, R_1, \cdots, R_L, g_q \in \mathbb{G}_q.$$

It then, computes $G = gR_g$, $F = fR_f$, $V = vR_v$, $H_1 = h_1R_1, \cdots, H_L = h_LR_L$ and $E = e(g, w)$, and it publishes the description of a group $\mathbb{G}$ and *params* as:

$$\mathit{params} \leftarrow [g_q, g_p, G, F, V, H_1, \cdots, H_L, E]$$

and retains *MK* as the secret values:

$$\mathit{MK} \leftarrow [p,\ q,\ g,\ f,\ v,\ h_1,\ \cdots,\ h_L,\ w]$$

The group description contains n but not p, q.

KeyGenerate$(\mathit{MK}, \mathrm{ID})$**:** To generate a private key corresponding to $\mathrm{ID} = [\mathrm{I_1}, \mathrm{I_2}, \cdots, \mathrm{I_k}] \in (\mathbb{Z}_n)^k$, the *KeyGenerate* algorithm first takes *MK* and ID as input. Next, it picks random integers $r_1, r_2, s_1, s_2, t_1, t_2 \in \mathbb{Z}_n$ satisfying equations $s_1 \cdot t_2 - s_2 \cdot t_1 \not\equiv 0 \mod p$ and $\not\equiv 0 \mod q$. The algorithm randomly chooses integers and checks whether or not the equation holds. If it does not, the algorithm chooses other random integers and repeats this procedure until the equation does hold. Since the equation holds without probability $\frac{p+q-1}{n}$, this iteration will finish immediately. The private key $\mathrm{Pvk}^{\mathrm{ID}}$ consisting of two subkeys $\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}} \in (\mathbb{G}_p)^{L-k+3}$ and $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}} \in (\mathbb{G}_p)^{2(L-k+3)}$ is output. $\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}}$ is used for decryption and delegation, and $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}}$ is used for re-randomization.

$$\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}} \leftarrow [w(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}})^{r_1} f^{r_2},\ g^{r_1},\ g^{r_2},\ h_{k+1}^{r_1},\ \cdots\ h_L^{r_1}].$$

$$\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}} \leftarrow [[(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}})^{s_1} f^{s_2}, g^{s_1}, g^{s_2}, h_{k+1}^{s_1}, \cdots h_L^{s_1}], [(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}})^{t_1} f^{t_2}, g^{t_1}, g^{t_2}, h_{k+1}^{t_1}, \cdots h_L^{t_1}]].$$

Derive$(\mathrm{Pvk}^{\mathrm{ID}_{|\mathrm{k}-1}}, \mathrm{ID}_{|\mathrm{k}})$**:** The private key for $\mathbf{ID}_{|\mathbf{k}} \in (\mathbb{Z}_n)^k$, where $2 \le k \le L$, is derived from a given private key for the parent,

$$\mathrm{Pvk}^{\mathrm{ID}_{|\mathrm{k}-1}} = [\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}_{|\mathrm{k}-1}}, \mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}_{|\mathrm{k}-1}}]$$

$$= [\ [a_0, a_1, a_2, b_k, \cdots, b_L],\ [\ [\alpha_0, \alpha_1, \alpha_2, \beta_k, \cdots, \beta_L],\ [\alpha_0', \alpha_1', \alpha_2', \beta_k', \cdots, \beta_L']\ ]\ ].$$

To generate $\mathrm{Pvk}^{\mathrm{ID}_{|\mathrm{k}}}$, pick random integers γ_1, γ_2, γ_3, δ_1, δ_2, $\delta_3 \in \mathbf{Z}_n$ satisfying equations $g_p^{\gamma_2\cdot\delta_3-\gamma_3\cdot\delta_2} \not\equiv 1$ and $g_q^{\gamma_2\cdot\delta_3-\gamma_3\cdot\delta_2} \not\equiv 1$ holds. To select four integers satisfying the equations, uniformly and independently choose four integers from $\mathbb{Z}_n$ and check the equation. If the equation does not hold, then choose four other integers and repeat the procedure. Since four randomly chosen integers γ_2, γ_3, δ_2 and δ_3 satisfy the above equations without negligible probability $\frac{p+q-1}{n}$, this iteration will finish immediately. Therefore we consider four randomly chosen integers satisfying above equations as a random element in $GL_2(\mathbb{Z}_n)$. Lastly the *Derive* algorithm outputs $\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}_{|\mathrm{k}}}$ and $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}_{|\mathrm{k}}}$ as follows.

Step 1 (delegation procedure):

$$[\ \zeta_0, \zeta_1, \zeta_2, \eta_{k+1}, \cdots, \eta_L\ ] \leftarrow [\ a_0 \cdot b_k^{\mathrm{I_k}}, a_1, a_2, b_{k+1}, \cdots, b_L\ ]$$

$$[\ \theta_0, \theta_1, \theta_2, \phi_{k+1}, \cdots, \phi_L\ ] \leftarrow [\alpha_0 \cdot \beta_k^{\mathrm{I_k}}, \alpha_1, \alpha_2, \beta_{k+1}, \cdots, \beta_L],$$
$$[\ \theta'_0, \theta'_1, \theta'_2, \phi'_{k+1}, \cdots, \phi'_L\ ] \leftarrow [\alpha'_0 \cdot \beta'^{\mathrm{I_k}}_k, \alpha'_1, \alpha'_2, \beta'_{k+1}, \cdots, \beta'_L].$$

Step 2(re-randomization procedure):

$$\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}_{|\mathrm{k}}} \leftarrow [\ \zeta_0 \theta_0^{\gamma_1} \theta'^{\delta_1}_0, \zeta_1 \theta_1^{\gamma_1} \theta'^{\delta_1}_1, \zeta_2 \theta_2^{\gamma_1} \theta'^{\delta_1}_2, \eta_{k+1} \phi_{k+1}^{\gamma_1} \phi'^{\delta_1}_{k+1}, \cdots, \eta_L \phi_L^{\gamma_1} \phi'^{\delta_1}_L\ ]$$

$$\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}_{|\mathrm{k}}} \leftarrow [\ [\theta_0^{\gamma_2} \theta'^{\delta_2}_0, \theta_1^{\gamma_2} \theta'^{\delta_2}_1, \theta_2^{\gamma_2} \theta'^{\delta_2}_2, \phi_{k+1}^{\gamma_2} \phi'^{\delta_2}_{k+1}, \cdots, \phi_L^{\gamma_2} \phi'^{\delta_2}_L\ ],$$
$$[\theta_0^{\gamma_3} \theta'^{\delta_3}_0, \theta_1^{\gamma_3} \theta'^{\delta_3}_1, \theta_2^{\gamma_3} \theta'^{\delta_3}_2, \phi_{k+1}^{\gamma_3} \phi'^{\delta_3}_{k+1}, \cdots, \phi_L^{\gamma_3} \phi'^{\delta_3}_L]\ ].$$

We note that private keys generated by the *Derive* algorithm have the same structure and distribution as those generated by the *KeyGenerate* algorithm. Two random integers r_1 and r_2 of $\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID|k-1}}$ are re-randomized as follows:

$$\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} + \begin{pmatrix} s_1 & t_1 \\ s_2 & t_2 \end{pmatrix} \cdot \begin{pmatrix} \gamma_1 \\ \delta_1 \end{pmatrix}$$

Since we choose γ_1 and δ_1 uniformly and independently from $\in \mathbb{Z}_n$, the above value is also distributed uniformly in $(\mathbb{Z}_n)^2$. Therefore, $\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID|k}}$ has the same distribution as that of the private key generated by the *KeyGenerate* algorithm.

Random integers of $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID|k-1}}$ are re-randomized as follows:

$$A \cdot B \ \ \textit{where} \ \ A = \begin{pmatrix} s_1 & t_1 \\ s_2 & t_2 \end{pmatrix}, \ \ B = \begin{pmatrix} \gamma_2 & \gamma_3 \\ \delta_2 & \delta_3 \end{pmatrix}$$

Since $A \in GL_2(\mathbb{Z}_n)$ and B are uniformly chosen from $GL_2(\mathbb{Z}_n)$, $A \cdot B$ is also uniformly distributed in $GL_2(\mathbb{Z}_n)$. Therefore, the private key generated by the *Derive* algorithm has the same distribution as that of the private key generated by the *KeyGenerate* algorithm.

Encrypt$(\mathit{params}, \mathrm{ID}, \mathrm{Msg})$**:** First, pick a random integer $s \in \mathbb{Z}_n$ and random elements $Z_1, Z_2, Z_3 \in \mathbb{G}_q$ to encrypt message $\mathrm{Msg} \in \mathbb{G}_T$ for a given identity $\mathrm{ID} = [\mathrm{I}_1, \cdots, \mathrm{I_k}] \in (\mathbb{Z}_n)^k$. A random element of $\mathbb{G}_q$ can be chosen by raising g_q to random exponents from $\mathbb{Z}_n$. Next, the *Encrypt* algorithm outputs the ciphertext

$$\mathrm{CT} \ \leftarrow \ [\mathrm{Msg} \cdot E^s, \ G^s \cdot Z_1, \ F^s \cdot Z_2, \ (V \prod_{i=1}^{k} H^{\mathrm{I_i}})^s \cdot Z_3] \ \in \mathbb{G}_T \times \mathbb{G}^3.$$

Decrypt$(\mathrm{Pvk^{ID}}, \mathrm{CT})$**:** Consider $\mathrm{ID} = [\mathrm{I}_1, \cdots, \mathrm{I_k}]$. To decrypt ciphertext $\mathrm{CT} = [C_1, C_2, C_3, C_4]$, using the first three elements of subkey $\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}} = [a_0, a_1, a_2, b_{k+1}, \cdots, b_L]$ of the private key $\mathrm{Pvk^{ID}}$, output

$$\mathrm{Msg} \ \leftarrow \ C_1 \cdot \frac{e(a_1, C_4) \cdot e(a_2, C_3)}{e(a_0, C_2)}.$$

We can easily check the correctness of the *Decrypt* algorithm for valid ciphertext as follows.

$$C_1\frac{e(a_1,C_4)e(a_2,C_3)}{e(a_0,C_2)} = \mathrm{Msg}\cdot e(g,w)^s \frac{e(g^{r_1},(V\prod_{i=1}^{k} H^{\mathrm{I_i}})^s Z_3)e(g^{r_2},F^s Z_2)}{e(w(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}})^{r_1} f^{r_2}, G^s Z_1)}$$

$$= \mathrm{Msg}\cdot e(g,w)^s \frac{e(g^{r_1},(V\prod_{i=1}^{k} H^{\mathrm{I_i}})^s)e(g^{r_2},f^s)}{e(w(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}})^{r_1} f^{r_2}, g^s)} = \mathrm{Msg}$$

The second equality holds because $e(h_p,h_q)=1$ for all $h_p\in\mathbb{G}_p$ and $h_q\in\mathbb{G}_q$.

3.2 Proof of Security

In this section, we explain the security of our construction. Our construction is similar to that of BBG-HIBE except for the blinding factors and the re-randomization subkey of the private key. Since the re-randomization subkey does not contain an element of the master key, w, adding the re-randomization subkey does not effect the semantic security. Therefore, we can demonstrate the semantic security of our construction in a similar manner to that for BBG-HIBE.

To prove anonymity, we use hybrid steps similar to that of [11]. The security of HVE is based on the composite 3-party Diffie-Hellman assumption, however we use the L-cDH which is a stronger assumption than c3DH, where L is the maximum hierarchy depth. The reason we introduce and use the L-cDH assumption is like that the semantic securities of our scheme and BBG-HIBE scheme are based on the Decision L-BDHI* assumption which is a stronger assumption than the Decision BDH assumption and depends on the maximum hierarchy depth L. The private key of BBG-HIBE has delegation key elements whose number depends on the maximum hierarchy depth. Therefore it may be that an adversary attacking the BBG-HIBE scheme or our scheme can get more information from the private key extraction queries than from other HIBE schemes in which the private key does not contain delegation key elements and is secure under the Decision BDH assumption, for example, GS-HIBE and BB-HIBE. The Decision L-BDHI assumption and the L-cDH assumption guarantee that any computationally bounded adversary can get no information about the message and the identity, respectively, from the chosen private keys and the challenge ciphertext with reasonable constraints.

Theorem 1. *If group generator algorithm $\mathcal{G}$ satisfies the (t,ϵ_1)-Decision L-wBDHI* assumption and (t,ϵ_2)-L-cDH assumption, then our HIBE scheme with maximum hierarchy depth L is $(q_s,\hat{t_1},\hat{\epsilon_1})$*-IND-sID-CPA *secure and $(q_s,\hat{t_2},\hat{\epsilon_2})$-*ANON-sID-CPA *secure with $\hat{t_1}$, $\hat{t_2}=\Theta(t)$, $\hat{\epsilon_1}=\Theta(\epsilon_1+\epsilon_2)$, and $\hat{\epsilon_2}=\Theta(\epsilon_1+\epsilon_2/(1-\frac{p+q-1}{n})^{q_s})$.*

We prove Theorem 1 using hybrid experiments under the Decision L-wBDHI* assumption and L-cDH assumption.

$\mathrm{Game}_1 : \mathrm{CT}_1 = [C_1,\ C_2,\ C_3,\ C_4]$
$\mathrm{Game}_2 : \mathrm{CT}_2 = [C_1\cdot R_p,\ C_2,\ C_3,\ C_4]$
$\mathrm{Game}_3 : \mathrm{CT}_3 = [C_1\cdot R = R_1,\ C_2,\ C_3,\ C_4]$

$\mathrm{Game}_4 : \mathrm{CT}_4 = [R_1,\ R_2,\ C_3,\ C_4]$
$\mathrm{Game}_5 : \mathrm{CT}_5 = [R_1,\ R_2,\ R_3,\ R_4]$

where R_p is a randomly chosen element from $\mathbb{G}_{T,p}$; R, R_1 are uniformly distributed in $\mathbb{G}_T$; and R_2, R_3, R_4 are uniformly distributed in $\mathbb{G}$.

We show that under the Decision L-wBDHI* assumption and L-cDH assumption, there are no algorithms that distinguish between Game_1 and Game_2, or between Game_2 and Game_3, or between Game_3 and Game_4, or between Game_4 and Game_5. Challenge ciphertext CT_5 is composed of four random group elements, so it does not leak any information about the message or the identity. Therefore indistinguishability between games prove Theorem 1. First, we prove the indistinguishability between games, and next we complete the proof of Theorem 1.

Indistinguishability between Game_1 and Game_2.

Lemma 1. *If group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ)-Decision L-wBDHI* assumption, there is no adversary with running time t that distinguishes between* Game_1 *and* Game_2 *with advantage ϵ.*

Proof. We assume that there exists adversary $\mathcal{A}$ that distinguishes between Game_1 and Game_2 with advantage ϵ. We show that there is a simulator $\mathcal{B}$ using $\mathcal{A}$ to solve the Decision L-wBDH* problem with advantage ϵ. The proof is similar to the proof of the semantic security of BBG-HIBE except for the treatment of the re-randomization key in our proof.

The challenger makes a challenge pair $(\mathbf{Z},\ T')$ of the Decision L-wBDHI* assumption. which is defined in Section 2.3 and gives the challenge pair to simulator $\mathcal{B}$. Let $A_i = g_p^{a^i}$ where $g_p^{a^i}$ is defined in $\mathbf{Z}$ for $1 \leq i \leq L+1$.

Initialization. Adversary $\mathcal{A}$ chooses challenge identity $\mathrm{ID} = [\mathrm{I_1}, \mathrm{I_2}, \cdots, \mathrm{I_m}]$, and sends it to simulator $\mathcal{B}$. Then, $\mathcal{B}$ sets $\mathrm{I_{m+1}} = \cdots = \mathrm{I_L} = 0$. Hence, a simulator can always consider the length of the challenge identity as L.

Setup. $\mathcal{B}$ chooses random integers and random elements

$$\gamma, x, y, z, x_1, \cdots, x_L \in \mathbb{Z}_n, \qquad R_g, R_f, R_v, R_{h,1}, \cdots, R_{h,l} \in \mathbb{G}_q.$$

A random element of $\mathbb{G}_p(\mathbb{G}_q)$ can be chosen by raising $g_p(g_q$, respectively) to random exponents from $\mathbb{Z}_n$. $\mathcal{B}$ sets $G = g_p R_g$, $F = g_p^z R_f$, $V = (g_p^y \cdot \prod_{i=1}^{L} (A_{L-i+1})^{\mathrm{I_i}}) R_v$, $H_i = g_p^{x_i} / A_{L-i+1} R_{h,i}$ for $1 \leq i \leq L$, and $E = e(A_1, A_L \cdot g_p^{\gamma})$. Then, $\mathcal{B}$ publishes system parameters as

$$params \leftarrow [g_q, G, F, V, H_1, \cdots, H_L, E]$$

where *params* generated by $\mathcal{B}$ has the same distribution as that of an actual scheme. The master key w corresponding to the system parameters is $(A_L \cdot g_p^{\gamma})^a = A_{L+1} \cdot A_1^{\gamma}$. Since $\mathcal{B}$ does not have A_{L+1}, $\mathcal{B}$ does not know the master key.

Query Phase1. $\mathcal{A}$ queries the private key for $\mathrm{ID}^* = [\mathrm{I}_1^*, \mathrm{I}_2^*, \cdots, \mathrm{I}_\mathrm{u}^*]$, where $u \leq L$ is distinct from ID and all its prefixes. This private query is carried on an adaptively chosen identity by $\mathcal{A}$. Let k be the smallest integer such that $\mathrm{I_k} \neq \mathrm{I}_\mathrm{k}^*$.

Then, first $\mathcal{B}$ generates the private key corresponding to $[\mathrm{I}_1^*, \mathrm{I}_2^*, \cdots, \mathrm{I}_\mathrm{k}^*]$ and runs the *Derive* algorithm to make ID^*. $\mathcal{B}$ first chooses random integers $r_1, r_2 \in \mathbb{Z}_n$. We posit $\hat{r_1} = r_1 + \frac{a^k}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}$. Next, it generates $\mathrm{Pvk}^{\mathrm{ID}^*} = [\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}, \mathrm{Pvk}_\mathrm{r}^{\mathrm{ID}^*}]$. We observe the first component of $\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}$.

$$w(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}^*})^{\hat{r_1}} f^{r_2} = w \cdot (v\prod_{i=1}^{k} h_i^{\mathrm{I_i}^*})^{r_1} f^{r_2} \cdot (v\prod_{i=1}^{k} h_i^{\mathrm{I_i}^*})^{\frac{a^k}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}}$$

Since v is $g_p^y \cdot \prod_{i=1}^{L} (A_{L-i+1})^{\mathrm{I_i}}$ and h_i is $g_p^{x_i}/A_{L-i+1}$ and f is g_p^z which can be obtained by removing the blinding factor from V, H_i and F, respectively, and r_1, r_2 are chosen by simulator, $\mathcal{B}$ can compute the second term in the above expression. We focus on the product of the first and third terms in the above expression, $w \cdot (v\prod_{i=1}^{k} h_i^{\mathrm{I_i}})^{\frac{a^k}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}}$. Then

$$w \cdot (v\prod_{i=1}^{k} h_i^{\mathrm{I_i}^*})^{\frac{a^k}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}} = A_{L+1} A_1^{\gamma} \cdot (g_p^y \prod_{i=1}^{L} (A_{L-i+1})^{\mathrm{I_i}} \prod_{i=1}^{k} (g_p^{x_i}/A_{L-i+1})^{\mathrm{I_i}^*})^{\frac{a^k}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}}$$

$$= A_{L+1} A_1^{\gamma} \cdot (g_p^y A_{L-k+1}^{\mathrm{I_k} - \mathrm{I}_\mathrm{k}^*} \prod_{i=k+1}^{L} (A_{L-i+1})^{\mathrm{I_i}} \prod_{i=1}^{k} g_p^{x_i \mathrm{I}_\mathrm{i}^*})^{\frac{a^k}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}}$$

$$= A_{L+1} A_1^{\gamma} \cdot (A_k^y A_{L+1}^{\mathrm{I_k} - \mathrm{I}_\mathrm{k}^*} \prod_{i=k+1}^{L} (A_{L+k-i+1})^{\mathrm{I_i}} \prod_{i=1}^{k} A_k^{x_i \mathrm{I}_\mathrm{i}^*})^{\frac{1}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}}$$

$$= A_1^{\gamma} \cdot (A_k^y \prod_{i=k+1}^{L} (A_{L+k-i+1})^{\mathrm{I_i}} \prod_{i=1}^{k} A_k^{x_i \mathrm{I}_\mathrm{i}^*})^{\frac{1}{\mathrm{I}_\mathrm{k}^* - \mathrm{I}_\mathrm{k}}}$$

Since $\mathcal{B}$ knows all the terms in the above expression, it can compute the first component of $\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}$. Since the remaining elements in $\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}$ do not involve A_{L+1}, $\mathcal{B}$ can compute all of them. $\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}$ is distributed as if $\hat{r_1} = r_1 + \frac{a^k}{\mathrm{I_k} - \mathrm{I}_\mathrm{k}^*}$ and r_2 are the randomness of $\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}$. Since $\hat{r_1}$ and r_2 are uniformly and independently distributed in $\mathbb{Z}_n$, $\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}*}$ has the same distribution as that of the actual key distribution.

To generate $\mathrm{Pvk}_\mathrm{r}^{\mathrm{ID}^*}$, $\mathcal{B}$ choose $s_1, s_2, t_1, t_2 \in \mathbb{Z}_n$. Since no elements in $\mathrm{Pvk}_\mathrm{r}^{\mathrm{ID}^*}$ associate with master key w, $\mathcal{B}$ can compute $\mathrm{Pvk}_\mathrm{r}^{\mathrm{ID}^*}$ using s_1, s_2, t_1, t_2 as its randomness. Since the random integers used in $\mathrm{Pvk}_\mathrm{r}^{\mathrm{ID}^*}$ have to satisfy equation $s_1 \cdot t_2 - s_2 \cdot t_1 \not\equiv 0 \mod p$ and $\not\equiv 0 \mod q$, the simulator has to check equation $g_p^{s_1 \cdot t_2 - s_2 \cdot t_1} \not\equiv 1$ and $g_q^{s_1 \cdot t_2 - s_2 \cdot t_1} \not\equiv 1$. If the random integers used in $\mathrm{Pvk}_\mathrm{r}^{\mathrm{ID}^*}$ do not satisfy the equation, then the simulator chooses other random integers s_1, s_2, t_1 and t_2 and repeats the same procedure until the equation does hold. Since the equation holds without probability $\frac{p+q-1}{n}$, this iteration will finish immediately. Therefore $\mathrm{Pvk}^{\mathrm{ID}^*}$ has the same distribution as that of the actual key distribution.

Challenge. $\mathcal{A}$ sends a message Msg $\in \mathbb{G}$ to $\mathcal{B}$. Then, $\mathcal{B}$ picks random elements Z_1, Z_2 and Z_3 from $\mathbb{G}_q$ and outputs a challenge ciphertext

$$\mathrm{CT} = [\mathrm{Msg} \cdot T' \cdot e(A_1, h^{\gamma}), \quad h \cdot Z_1, \quad h^z \cdot Z_2, \quad h^{y + \Sigma_{i=1}^{L} \mathrm{I_i} x_i} \cdot Z_3],$$

where h and T' are given from challenge pair $(\mathbf{Z}, T')$ We consider h as g_p^c for some unknown $c \in \mathbb{Z}_n$.

If $T' = T$, then CT is equal to

$$[\mathrm{Msg} \cdot e(g_p, g_p^c)^{a^{L+1}} e(g_p^a, g_p^{c\gamma}), g_p^c Z_1, (g_p^z)^c Z_2, (\prod_{i=1}^{L} (g_p^{x_i}/A_{L-i+1})^{\mathrm{I_i}} g_p^y \prod_{i=1}^{L} A_{L-i+1}^{\mathrm{I_i}})^c Z_3]$$

$$= [\mathrm{Msg} \cdot e(A_1, A_L g_p^{\gamma})^c, \quad G^c Z_1', \quad F^c Z_2', \quad (V \prod_{i=1}^{L} H_i^{\mathrm{I_i}})^c \ Z_3']$$

$$= [\mathrm{Msg} \cdot e(A_1, A_L g_p^{\gamma})^c, \quad G^c Z_1', \quad F^c Z_2', \quad (V \prod_{i=1}^{m} H_i^{\mathrm{I_i}})^c Z_3'].$$

Therefore, CT is a ciphertext of Game$_1$. Otherwise, T is a uniformly and independently chosen element from $\mathbb{G}_T$. In that case, the first component of ciphertext is random from the adversarial point of view. Therefore, CT is a ciphertext of Game$_2$.

Query Phase2. $\mathcal{A}$ adaptively queries $\mathcal{B}$ with the same constraints as in Query Phase 1. $\mathcal{B}$ sends corresponding private keys as before.

Guess. $\mathcal{B}$ outputs the same bit as $\mathcal{A}$; i.e., if $\mathcal{A}$ outputs 1 (Game$_1$), then $\mathcal{B}$ also outputs 1 ($T' = T$). Since $\mathcal{B}$ played Game$_1$ with $T' = T$ and played Game$_2$ with T' as a random element from $\mathbb{G}_p$, $\mathcal{B}$'s advantage in the L-wBDHI* game is exactly ϵ, the same as $\mathcal{A}$'s. □

Indistinguishability between Game$_2$ and Game$_3$.

Lemma 2. *If group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ)-BSD assumption, there is no adversary with running time t that distinguishes between* Game$_2$ *and* Game$_3$ *with advantage ϵ.*

Proof. We assume that there exists adversary $\mathcal{A}$ distinguishing between Game$_2$ and Game$_3$ with ϵ advantage. We show that there is a simulator $\mathcal{B}$ using $\mathcal{A}$ to solve the BSD problem with advantage ϵ.

The challenger makes a challenge pair ($\mathbf{Z}$, T') of the BSD problem, which is defined in Section 2.3 and give the challenge pair to simulator $\mathcal{B}$.

Initialization. $\mathcal{A}$ sends a challenge identity ID to $\mathcal{B}$.
Setup. $\mathcal{B}$ generates system parameters as an actual setup algorithm. $\mathcal{B}$ can choose all random elements from $\mathbb{G}_p$ and $\mathbb{G}_p$ by using g_p and g_q.

Query Phase1. $\mathcal{A}$ queries $\mathcal{B}$ and $\mathcal{B}$ responds to queries as the actual key generation center.

Challenge. $\mathcal{A}$ sends message Msg, to $\mathcal{B}$. $\mathcal{B}$ outputs a normal ciphertext with the exception that the its first component is multiplied by T'.

Query Phase2. $\mathcal{A}$ adaptively queries $\mathcal{B}$ with the same constraints as in Query Phase 1. $\mathcal{B}$ sends corresponding private keys as before.

Guess. $\mathcal{B}$ outputs the same bit as $\mathcal{A}$; i.e., if $\mathcal{A}$ outputs 1 (Game$_2$), then $\mathcal{B}$ also outputs 1 ($T' = T$). Since $\mathcal{B}$ played Game$_2$ with T' is a random element from $\mathbb{G}_p$ and played Game$_3$ with T' as a random element from $\mathbb{G}$, $\mathcal{B}$'s advantage in the BSD game is exactly ϵ, the same as $\mathcal{A}$'s. □

Indistinguishability between Game$_3$ and Game$_4$

Lemma 3. *If group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ)-L-cDH assumption, there is no adversary with running time t that makes at most q_s key extraction queries and distinguishes between* Game$_3$ *and* Game$_4$ *with advantage $\epsilon/(1 - \frac{p+q-1}{n})^{q_s}$.*

Proof. We assume that there exists an adversary $\mathcal{A}$ distinguishing between Game$_3$ and Game$_4$ with advantage ϵ'. We show that there is a simulator $\mathcal{B}$ using $\mathcal{A}$ to solve the L-cDH problem with advantage $\epsilon' \cdot (1 - \frac{p+q-1}{n})^{q_s}$.

The challenger makes a challenge pair ($\mathbf{Z}$, T') of L-cDH which is defined in Section 2.3 and gives the challenge pair to simulator $\mathcal{B}$. Let $A_i = g_p^{a^i}$, $B = A_{L+1}R'_1$ and $C = A^b_{L+1}R'_2$ where $g_p^{a^i}$, R'_1 and R'_2 are defined in $\mathbf{Z}$ for $0 \leq i \leq L+1$.

Initialization. $\mathcal{A}$ chooses a challenge $\mathrm{ID} = [\mathrm{I_1}, \mathrm{I_2}, \cdots, \mathrm{I_m}]$ and sends it to simulator $\mathcal{B}$.

Setup. $\mathcal{B}$ chooses random integers and random elements

$$x, y, z, x_1, \cdots, x_L \in \mathbb{Z}_n, \qquad w \in \mathbb{G}_p, \qquad R_g, R_f, R_v, R_{h,1}, \cdots, R_{h,l} \in \mathbb{G}_q.$$

$\mathcal{B}$ puts $G = B^x R_g$, $F = g_p^z R_f$, $H_i = A_i^{x_i} R_{h,i}$ for $1 \leq i \leq L$, $V = (g_p^y / \prod_{j=1}^m H_j^{I_j})$ R_v, and $E = e(B^x, w)$. Then, $\mathcal{B}$ publishes system parameters as

$$params \leftarrow [g_q, G, F, V, H_1, \cdots, H_L, E]$$

Query Phase1. $\mathcal{A}$ queries the private key for $\mathrm{ID}^* = [\mathrm{I}_1^*, \mathrm{I}_2^*, \cdots, \mathrm{I}_\mathrm{u}^*]$ where $u \leq L$ distinct from ID and all its prefixes. This private query of an adaptively chosen identity is carried out by $\mathcal{A}$. Let k be the smallest integer such that $\mathrm{I_k} \neq \mathrm{I}_\mathrm{k}^*$. Then, first $\mathcal{B}$ generates the private key corresponding to $[\mathrm{I}_1^*, \mathrm{I}_2^*, \cdots, \mathrm{I}_\mathrm{k}^*]$ and runs *Derive* algorithm to make ID^*. $\mathcal{B}$ first choose random integers $r_1, r_2 \in \mathbb{Z}_n$. We posit $\hat{r_1} = \frac{z}{a^k} r_1 + \frac{z}{a^{k+1}} r_2, \hat{r_2} = -\frac{y}{a^k} r_1 - (\frac{x_k(\mathrm{I}_\mathrm{k}^* - \mathrm{I_k})}{a} + \frac{y}{a^{k+1}}) r_2$. Next, the algorithm generates $\mathrm{Pvk}^{\mathrm{ID}^*} = [\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}, \mathrm{Pvk}_\mathrm{r}^{\mathrm{ID}^*}]$. We observe the first component of $\mathrm{Pvk}_\mathrm{d}^{\mathrm{ID}^*}$, $w(v \prod_{i=1}^k h_i^{\mathrm{I_i}^*})^{\hat{r_1}} f^{\hat{r_2}}$. Since we know that v, h_i, f are the same as elements by removing blinding factors from V, H_i, F, respectively, we can rewrite the above component as follows:

$$w(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}^*})^{\hat{r_1}} f^{\hat{r_2}} = w((g_p^y/\prod_{j=1}^{m} A_j^{x_j I_j})\prod_{i=1}^{k} A_i^{x_i \mathrm{I_i^*}})^{\hat{r_1}} g_p^{z\hat{r_2}}$$

We focus on the exponent of g_p in $((g_p^y/\prod_{j=1}^{m} A_j^{x_j I_j})\prod_{i=1}^{k} A_i^{x_i \mathrm{I_i^*}})^{\hat{r_1}} g_p^{z\hat{r_2}}$. It is

$$(y - \sum_{j=1}^{m} a^j x_j I_j + \sum_{i=1}^{k} a^i x_i I_i^*)\hat{r_1} + z\hat{r_2}$$

$$= (y - \sum_{j=k+1}^{m} a^j x_j I_j + a^k x_k(I_k^* - I_k))\hat{r_1} + z\hat{r_2}$$

$$= (y - \sum_{j=k+1}^{m} a^j x_j I_j + a^k x_k(I_k^* - I_k))(\frac{z}{a^k}r_1 + \frac{z}{a^{k+1}}r_2) + z(-\frac{y}{a^k}r_1 - (\frac{x_k(\mathrm{I_k^*} - \mathrm{I_k})}{a} + \frac{y}{a^{k+1}})r_2)$$

$$= (-\sum_{j=k+1}^{m} a^{j-k} x_j I_j z + x_k(I_k^* - I_k)z)r_1 - \sum_{i=k+1}^{m} a^{i-k-1} x_i I_i z r_2$$

Since the exponent involves $a^1, \cdots, a^{m-k}$, x_j, z, r_1, r_2, ID and ID^*, $\mathcal{B}$ can compute the first component of $\mathrm{Pvk_d^{ID^*}}$. The remaining elements in $\mathrm{Pvk_d^{ID^*}}$ are $g^{\hat{r_1}}$, $g^{\hat{r_2}}$ and $h_i^{\hat{r_1}}$ for $k+1 \le i \le L$. Note that g and h_i are A_{L+1}^x and $A_i^{x_i}$, respectively, which are elements with blinding factors removed from G and H_i, respectively. Since the second component $g^{\hat{r_1}}$ is equal to $A_{L+1}^{x\hat{r_1}} = A_{L+1}^{x(\frac{z}{a^k}r_1 + \frac{z}{a^{k+1}}r_2)} = A_{L-k+1}^{xzr_1} A_{L-k}^{xzr_2}$, $\mathcal{B}$ can compute the second component of $\mathrm{Pvk_d^{ID^*}}$. Similarly, $\mathcal{B}$ can compute all the remaining elements of $\mathrm{Pvk_d^{ID^*}}$. Since $\hat{r_1}$ and $\hat{r_2}$ are uniformly and independently distributed in $\mathbb{Z}_n$, $\mathrm{Pvk_d^{ID*}}$ has the same distribution as an actual key distribution.

Next, $\mathcal{B}$ generates $\mathrm{Pvk_r^{ID^*}}$. Every component in $\mathrm{Pvk_r^{ID^*}}$ is the same as in $\mathrm{Pvk_d^{ID^*}}$ except for w of the first and $(L-k+4)$th components and for using different randomness. Since the procedure for generating $\mathrm{Pvk_d^{ID^*}}$ will work without w, $\mathcal{B}$ can generate $\mathrm{Pvk_r^{ID^*}}$ in a similar manner to generating $\mathrm{Pvk_d^{ID^*}}$. The details of this are highly similar to those of $\mathrm{Pvk_r^{ID^*}}$, so they are omitted. We let $s_1, s_2, t_1, t_2 \in \mathbb{Z}_n$ be random integers used for generating $\mathrm{Pvk_r^{ID^*}}$ and let $\hat{s_1} = \frac{z}{a^k}s_1 + \frac{z}{a^{k+1}}s_2$, $\hat{s_2} = -\frac{y}{a^k}s_1 - (\frac{x_k(\mathrm{I_k^*}-\mathrm{I_k})}{a} + \frac{y}{a^{k+1}})s_2$, $\hat{t_1} = \frac{z}{a^k}t_1 + \frac{z}{a^{k+1}}t_2$, $\hat{t_2} = -\frac{y}{a^k}t_1 - (\frac{x_k(\mathrm{I_k^*}-\mathrm{I_k})}{a} + \frac{y}{a^{k+1}})t_2$. Then $\mathrm{Pvk_r^{ID^*}}$ is distributed as if $\hat{s_1}, \hat{s_2}, \hat{t_1}$, and $\hat{t_2}$ are the randomness of $\mathrm{Pvk_r^{ID^*}}$. Since $\hat{s_1}$, $\hat{s_2}$, $\hat{t_1}$, and $\hat{t_2}$ are uniformly and independently distributed in $\mathbb{Z}_n$, four integers satisfy equations $\hat{s_1}\cdot\hat{t_2} - \hat{s_2}\cdot\hat{t_1} \not\equiv 0 \mod p$ and $\not\equiv 0 \mod q$ with probability $1 - \frac{p+q-1}{n}$. Therefore, the private key $\mathrm{Pvk^{ID^*}}$ generated by the simulator has the same structure and distribution as that of actual private key with probability $1 - \frac{p+q-1}{n}$.

Challenge. $\mathcal{A}$ sends message Msg to $\mathcal{B}$. $\mathcal{B}$ discards Msg and selects random elements $R \in \mathbb{G}$ and $Z_1, Z_2, Z_3 \in \mathbb{G}_q$. $\mathcal{B}$ sends ciphertext CT=$[R, C^x Z_1, T'^z Z_2, T'^y Z_3]$.

If $T' = T$, then CT is equal to

$$[R, (A_{L+1}^{b} R_2')^{x} Z_1, (g_p^{b} R_3')^{z} Z_2, (g_p^{b} R_3')^{y} Z_3] = [R, G^{b} Z_1', F^{b} Z_2', (\prod_{i=1}^{m} H_i^{\mathrm{I_i}} V)^{b} Z_3'].$$

Therefore CT is a ciphertext of Game$_3$. Otherwise, T can be written by $g_p^{r} R_3''$ as an element from $\mathbb{G}$ where r is a random integer chosen from $\mathbb{Z}_n$ and R_3'' is a random element chosen from $\mathbb{G}_q$. Then, CT is equal to

$$[R, (A_{L+1}^{b} R_2')^{x} Z_1, (g_p^{r} R_3'')^{z} Z_2, (g_p^{r} R_3'')^{y} Z_3] = [R, G^{b} Z_1', F^{r} Z_2', (V \prod_{i=1}^{m} H_i^{\mathrm{I_i}})^{r} Z_3'].$$

From an adversarial viewpoint, b and r first appear in ciphertext CT and both are integers uniformly and independently chosen from $\mathbb{Z}_n$. The third and fourth components in CT share the same random r. However, the second component uses random b independent from r. Therefore the second component of CT is a random element from the adversarial viewpoint, and CT is a ciphertext of Game$_4$.

Query Phase2. $\mathcal{A}$ adaptively queries $\mathcal{B}$ with the same constraints in Query Phase 1. $\mathcal{B}$ sends corresponding private keys as before.

Guess. $\mathcal{B}$ outputs the same bit as $\mathcal{A}$, i.e., if $\mathcal{A}$ outputs 1 (Game$_3$), then $\mathcal{B}$ also outputs 1 ($T' = T$). If $\mathcal{A}$ queried q_s times in total in *Query Phase1* and *Query Phase2*, then $\mathcal{B}$ responded with corresponding private keys having the same distribution as that of actual keys with probability $(1 - \frac{p+q-1}{n})^{q_s}$. Therefore $\mathcal{B}$'s advantage in the L-cDH game is $\epsilon' \cdot (1 - \frac{p+q-1}{n})^{q_s}$. □

Indistinguishability between Game$_4$ and Game$_5$

Lemma 4. *If group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ)-L-cDH assumption, there is no adversary with running time t that makes at most q_s key extraction queries and distinguishes between* Game$_4$ *and* Game$_5$ *with advantage $\epsilon / (1 - \frac{p+q-1}{n})^{q_s}$.*

Proof. We assume that there exists adversary $\mathcal{A}$ distinguishing between Game$_4$ and Game$_5$ with non-negligible advantage ϵ'. We show that there is a simulator $\mathcal{B}$ using $\mathcal{A}$ to solve the L-cDH problem with advantage $\epsilon' \cdot (1 - \frac{p+q-1}{n})^{q_s}$.

The challenger makes a challenge pair ($\mathbf{Z}$, T') of L-cDH which is defined in Section 2.3 and give the challenge pair to simulator $\mathcal{B}$. Let $A_i = g_p^{a^i}$, $B = A_{L+1} R_1'$ and $C = A_{L+1}^{b} R_2'$ where $g_p^{a^i}$, R_1' and R_2' are defined in $\mathbf{Z}$ for $0 \le i \le L+1$.

Initialization. $\mathcal{A}$ chooses a challenge ID $= [\mathrm{I_1}, \mathrm{I_2}, \cdots, \mathrm{I_m}]$ and sends it to simulator $\mathcal{B}$.

Setup. $\mathcal{B}$ chooses random integers and random elements

$$x, y, z, x_1, \cdots, x_L \in \mathbb{Z}_n, \qquad w \in \mathbb{G}_p, R_g, R_f, R_v, R_{h,1}, \cdots, R_{h,l} \in \mathbb{G}_q.$$

$\mathcal{B}$ puts $G = g_p^x R_g$, $F = B^z R_f$, $V = (g_p^y / \prod_{j=1}^m H_j^{I_j}) R_v$, $H_i = A_{L+1-i}^{x_i} R_{h,i}$ for $1 \le i \le L$, and $E = e(g_p^x, w)$. Then, $\mathcal{B}$ publishes system parameters as

$$\mathit{params} \leftarrow [g_q, G, F, V, H_1, \cdots, H_L, E].$$

Query Phase1. $\mathcal{A}$ queries the private key for $\mathrm{ID}^* = [\mathrm{I}_1^*, \mathrm{I}_2^*, \cdots, \mathrm{I}_\mathrm{u}^*]$ where $u < L$ is distinct from ID and all its prefixes. This private query is carried out on an identity adaptively chosen by $\mathcal{A}$. Let k be the smallest integer such that $\mathrm{I_k} \neq \mathrm{I_k^*}$. Then, first $\mathcal{B}$ generates the private key corresponding to $[\mathrm{I}_1^*, \mathrm{I}_2^*, \cdots, \mathrm{I}_\mathrm{k}^*]$ and runs the *Derive* algorithm to make ID^*. $\mathcal{B}$ first chooses random integers $r_1, r_2 \in \mathbb{Z}_n$. We posit $\hat{r_1} = a^k z r_1 + r_2, \hat{r_2} = -x_k(\mathrm{I_k^*} - \mathrm{I_k}) r_1$. Next, the algorithm generates $\mathrm{Pvk}^{\mathrm{ID}^*} = [\mathrm{Pvk_d^{ID^*}}, \mathrm{Pvk_r^{ID^*}}]$. We observe the first component in $\mathrm{Pvk_d^{ID^*}}$, $w(v \prod_{i=1}^k h_i^{\mathrm{I_i}^*})^{\hat{r_1}} f^{\hat{r_2}}$. Since we know that v, h_i, f are same as elements by removing blinding factor from V, H_i, F, respectively, we can rewrite above component as follows:

$$w(v\prod_{i=1}^{k} h_i^{\mathrm{I_i}^*})^{\hat{r_1}} f^{\hat{r_2}} = w((g_p^y / \prod_{j=1}^{m} A_{L+1-j}^{x_j I_j}) \prod_{i=1}^{k} A_{L+1-i}^{x_i \mathrm{I_i^*}})^{\hat{r_1}} A_{L+1}^{z\hat{r_2}}$$

We focus on the exponent of g_p in $((g_p^y / \prod_{j=1}^m A_{L+1-j}^{x_j I_j}) \prod_{i=1}^k A_{L+1-i}^{x_i \mathrm{I_i^*}})^{\hat{r_1}} A_{L+1}^{z\hat{r_2}}$. It is

$$(y - \sum_{j=1}^{m} a^{L+1-j} x_j I_j + \sum_{i=1}^{k} a^{L+1-i} x_i I_i^*)\hat{r_1} + a^{L+1} z\hat{r_2}$$

$$= (y - \sum_{j=k+1}^{m} a^{L+1-j} x_j I_j + a^{L+1-k} x_k (I_k^* - I_k))\hat{r_1} + a^{L+1} z\hat{r_2}$$

$$= (y - \sum_{j=k+1}^{m} a^{L+1-j} x_j I_j + a^{L+1-k} x_k (I_k^* - I_k))(a^k z r_1 + r_2) + a^{L+1} z(-x_k(\mathrm{I_k^*} - \mathrm{I_k}) r_1)$$

$$= (y a^k z - \sum_{j=k+1}^{m} a^{L+k+1-j} x_j I_j z) r_1 + (y - \sum_{j=k+1}^{m} a^{L+1-j} x_j I_j + a^{L+1-k} x_k (I_k^* - I_k)) r_2$$

Since the exponent involves $a^1, \cdots, a^L$, x_j, z, r_1, r_2 and identities, $\mathcal{B}$ can compute the first component of $\mathrm{Pvk_d^{ID^*}}$. The remaining elements in $\mathrm{Pvk_d^{ID^*}}$ are $g^{\hat{r_1}}$, $g^{\hat{r_2}}$ and $h_i^{\hat{r_1}}$ for $k+1 \le i \le L$. Note that g and h_i are g_p^x and $A_{L+1-i}^{x_i}$, respectively, which are elements with blinding factors removed from G and H_i, respectively. Since the second component $g^{\hat{r_1}}$ is equal to $g_p^{x\hat{r_1}} = A_k^{xzr_1} g_p^{xr_2}$, $\mathcal{B}$ can compute the second component of $\mathrm{Pvk_d^{ID^*}}$. Similarly, $\mathcal{B}$ can compute all the remaining elements of $\mathrm{Pvk_d^{ID^*}}$. Since $\hat{r_1}$ and $\hat{r_2}$ are uniformly and independently distributed in $\mathbb{Z}_n$, $\mathrm{Pvk_d^{ID*}}$ has the same distribution as an actual key distribution.

Next, $\mathcal{B}$ generates $\mathrm{Pvk_r^{ID^*}}$. Every component in $\mathrm{Pvk_r^{ID^*}}$ is the same as in $\mathrm{Pvk_d^{ID^*}}$ except for w of the first and $(L-k+4)$th components and for using different randomness. Since generating procedure of $\mathrm{Pvk_d^{ID^*}}$ will work without

w, $\mathcal{B}$ can generate $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}^*}$ is a similar manner to generating $\mathrm{Pvk}_{\mathrm{d}}^{\mathrm{ID}^*}$. The details of this are highly similar to those of $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}^*}$, so they are omitted. We let $s_1, s_2, t_1, t_2 \in \mathbb{Z}_n$ be random integers used for generating $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}^*}$ and let

$$\hat{s_1} = a^k z s_1 + s_2,\ \ \hat{s_2} = -x_k(\mathrm{I}_{\mathrm{k}}^* - \mathrm{I}_{\mathrm{k}})s_1,\ \ \hat{t_1} = a^k z t_1 + t_2, \text{ and } \hat{t_2} = -x_k(\mathrm{I}_{\mathrm{k}}^* - \mathrm{I}_{\mathrm{k}})t_1.$$

Then $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}^*}$ is distributed as if $\hat{s_1}, \hat{s_2}, \hat{t_1}$, and $\hat{t_2}$ is the randomness of $\mathrm{Pvk}_{\mathrm{r}}^{\mathrm{ID}^*}$. Since $\hat{s_1}$, $\hat{s_2}$, $\hat{t_1}$, and $\hat{t_2}$ are uniformly and independently distributed in $\mathbb{Z}_n$, four integers satisfy equations $\hat{s_1} \cdot \hat{t_2} - \hat{s_2} \cdot \hat{t_1} \not\equiv 0 \mod p$ and $\not\equiv 0 \mod q$ with probability $1 - \frac{p+q-1}{n}$. Therefore, the private key $\mathrm{Pvk}^{\mathrm{ID}^*}$ generated by the simulator has the same structure and distribution as that of actual private key with probability $1 - \frac{p+q-1}{n}$.

Challenge. $\mathcal{A}$ sends message Msg to $\mathcal{B}$. $\mathcal{B}$ discards Msg and selects random elements $R, R' \in \mathbb{G}$, $Z_1, Z_2 \in \mathbb{G}_q$. $\mathcal{B}$ sends ciphertext $\mathrm{CT} = [R, R', C^z Z_1, T'^y Z_2]$.

If $T' = T$, then

$$\mathrm{CT} = [R, R', (A_{L+1}^b R_2')^z Z_1, (g_p^b R_3')^y Z_2] = [R, R', F^b Z_1', (V \prod_{i=1}^{m} H_i^{\mathrm{I}_{\mathrm{i}}})^b Z_2'].$$

Therefore, CT is a ciphertext of Game_4. Otherwise, T can be written by $g_p^r R_3''$ as an element from $\mathbb{G}$, where r is a random integer chosen from $\mathbb{Z}_n$, and R_3'' is a random element chosen from $\mathbb{G}_q$. Then,

$$\mathrm{CT} = [R, R', (A_{L+1}^b R_2')^z Z_1, (g_p^r R_3'')^y Z_2] = [R, R', F^b Z_1', (V \prod_{i=1}^{m} H_i^{\mathrm{I}_{\mathrm{i}}})^r Z_2'].$$

From an adversarial viewpoint, b and r first appear in ciphertext CT and both are integers uniformly and independently chosen from $\mathbb{Z}_n$. Therefore, the third and fourth components of CT are independent random elements from the adversarial viewpoint, and CT is a ciphertext of Game_5.

Query Phase2. $\mathcal{A}$ adaptively queries $\mathcal{B}$ with the same constraints as in Query Phase 1. $\mathcal{B}$ sends corresponding private keys as before.

Guess. $\mathcal{B}$ outputs the same bit as $\mathcal{A}$, i.e., if $\mathcal{A}$ outputs 1 (Game_4), then $\mathcal{B}$ also outputs 1 ($T' = T$). If $\mathcal{A}$ queried q_s times in total in *Query Phase1* and *Query Phase2*, then $\mathcal{B}$ responded with corresponding private keys having the same distribution as that of actual keys with probability $(1 - \frac{p+q-1}{n})^{q_s}$. Therefore $\mathcal{B}$'s advantage in the L-cDH game is $\epsilon' \cdot (1 - \frac{p+q-1}{n})^{q_s}$. □

Proof of Theorem 1. If group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ_1)-Decision L-wBDHI* assumption and the (t, ϵ_2)-L-cDH assumption, then Lemma 1 and 2 show that there is no adversary with running time $\Theta(t)$ that makes at most q_s key extraction queries and distinguishes Game_1 and Game_3 with advantage $\epsilon_1 + \epsilon_2$. CT_3 does not leak any information about the message since there is no element involve in the message in CT_3. Therefore, if group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ_1)-Decision L-wBDHI* assumption and the (t, ϵ_2)-L-cDH assumption, our proposed HIBE scheme is $(q_s, \hat{t_1}, \hat{\epsilon_1})$-IND-sID-CPA secure with $\hat{t_1} = \Theta(t)$, $\hat{\epsilon_1} = \epsilon_1 + \epsilon_2$.

If group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ_2)-L-cDH assumption, then Lemma 3 and 4 show that there is no adversary with running time $\Theta(t)$ that makes at most q_s key extraction queries and distinguishes Game$_3$ and Game$_5$ with advantage $2\epsilon_2/(1 - \frac{p+q-1}{n})^{q_s}$.

CT$_5$ does not leak any information about the identity since all components of CT$_5$ are random group elements. Therefore, if group generator algorithm $\mathcal{G}$ satisfies the (t, ϵ_1)-Decision L-wBDHI* assumption and the (t, ϵ_2)-L-cDH assumption, then our proposed HIBE scheme is $(q_s, \hat{t_2}, \hat{\epsilon_2})$-ANON-sID-CPA secure with $\hat{t_2} = \Theta(t)$ and $\hat{\epsilon_2} = \epsilon_1 + \epsilon_2 + 2\epsilon_2/(1 - \frac{p+q-1}{n})^{q_s}$. This completes the proof. □

4 Comparison

The parameters of previous HIBE schemes, anonymous HIBE schemes and our proposed scheme are compared in table 1.

Table 1. HIBE schemes

	anonymity	# of group elements in public parameter	# of group elements in private key	# of group elements in ciphertext	# of pairing in decryption	security
GS-HIBE [17]	Non	2	k	$k+1$	k	w RO, f-ID
BB-HIBE [2]	Non	$L+3$	$k+1$	$k+2$	$k+1$	w/o RO, s-ID
BBG-HIBE [3]	Non	$L+3$	$L-k+2$	3	2	w/o RO, s-ID
BW-HIBE [12]	Ano	L^2+5L+7	$3L^2+(14-k)L$ $-3k+15$	$2L+6$	$2(L+2)$	w/o RO, s-ID
SW-dHVE [22]	Ano	$2L+6$	$(L-k)(k+5)$ $+k+3$	$L+4$	k	w/o RO, s-ID composit
This paper	Ano	$L+4$	$3(L-k+3)$	4	3	w/o RO, s-ID composit

L : the maximum depth of hierarchy, k : a depth of a corresponding identity,
Non: non-anonymous ID, Ano: anonymous ID
w/ RO : with random oracle, w/o RO : without random oracle
f-ID : full-ID, s-ID : selective-ID

Our scheme is the first reported constant size ciphertext anonymous HIBE scheme. Moreover, the computational cost is achieved the cheapest computational cost because during the decryption phase the computational cost is constant. The security proof can be shown without random oracles and our scheme achieves selective-ID CPA security.

5 Conclusion

We proposed an efficient anonymous Hierarchical Identity-Based Encryption scheme. The ciphertext of our scheme is only four group elements without depending on the depth of the hierarchy. Moreover, the computational cost of decryption is also efficient, just three bilinear pairings without depending on the hierarchy depth. The number of group elements in the public parameter as well as the private key of our anonymous HIBE are also the smallest among existing anonymous HIBE schemes.

The security of our scheme for a hierarchy depth L is selective-ID secure against a CPA adversary that was shown under the Decision L-wBDHI* assumption and the new L-composite Diffie-Hellman assumption without using random oracles. CCA2 security can be achieved by using techniques that are method of transforming from CPA-secure HIBE to CCA-secure HIBE, for example [14,8,9].

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Boneh, D., Boyen, X.: Efficient selective-ID identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertexts. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertexts. Cryptology ePrint Archive: Report 2005/015 (2005), http://eprint.iacr.org/2005/015
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public-key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
8. Boneh, D., Katz, J.: Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
9. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM Conference on Computer and Communications Security-CCS 2005. ACM Press, New York (2005)
10. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
11. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
12. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
13. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)

14. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
15. Chatterjee, S., Sarkar, P.: HIBE withe short public parameters without random oracle. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 145–160. Springer, Heidelberg (2006)
16. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
17. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
18. Horwitz, J., Lynn, B.: Towards hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
19. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
20. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. Cryptology ePrint Archive: Report 2007/404 (2007), http://eprint.iacr.org/2007/404
21. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Transactions Fundamentals E85-A(2), 481–484 (2002)
22. Shi, E., Waters, B.: Delegating capabilities in predicate encryption systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
23. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Towards Black-Box Accountable Authority IBE with Short Ciphertexts and Private Keys

Benoît Libert[1] and Damien Vergnaud[2,⋆]

[1] Université Catholique de Louvain, Microelectronics Laboratory
Place du Levant, 3 – 1348 Louvain-la-Neuve – Belgium
[2] Ecole Normale Supérieure – C.N.R.S. – I.N.R.I.A.
45, Rue d'Ulm – 75230 Paris CEDEX 05 – France

Abstract. At Crypto'07, Goyal introduced the concept of *Accountable Authority Identity-Based Encryption* as a convenient tool to reduce the amount of trust in authorities in Identity-Based Encryption. In this model, if the Private Key Generator (PKG) maliciously re-distributes users' decryption keys, it runs the risk of being caught and prosecuted. Goyal proposed two constructions: the first one is efficient but can only trace well-formed decryption keys to their source; the second one allows tracing obfuscated decryption boxes in a model (called *weak black-box* model) where cheating authorities have no decryption oracle. The latter scheme is unfortunately far less efficient in terms of decryption cost and ciphertext size. In this work, we propose a new construction that combines the efficiency of Goyal's first proposal with a very simple weak black-box tracing mechanism. Our scheme is described in the selective-ID model but readily extends to meet all security properties in the adaptive-ID sense, which is not known to be true for prior black-box schemes.

Keywords: Identity-based encryption, traceability, efficiency.

1 Introduction

Identity-based cryptography, first proposed by Shamir [36], alleviates the need for digital certificates used in traditional public-key infrastructures. In such systems, users' public keys are public identifiers (*e.g.* email addresses) and the matching private keys are derived by a trusted party called Private Key Generator (PKG). The first practical construction for *Identity-Based Encryption* (IBE) was put forth by Boneh and Franklin [7] – despite the bandwidth-demanding proposal by Cocks [15] – and, since then, a large body of work has been devoted to the design of schemes with additional properties or relying on different algorithmic assumptions [4,5,6,8,11,19,20,32,38].

[⋆] The first author acknowledges the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.) for their financial support and the BCRYPT Interuniversity Attraction Pole. The second author is supported by the European Commission through the IST Program under Contract ICT-2007-216646 ECRYPT II and by the French *Agence Nationale de la Recherche* through the PACE project.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 235–255, 2009.

In spite of its appealing advantages, identity-based encryption has not undergone rapid adoption as a standard. The main reason is arguably the fact that it requires unconditional trust in the PKG: the latter can indeed decrypt any ciphertext or, even worse, re-distribute users' private keys. The key escrow problem can be mitigated as suggested in [7] by sharing the master secret among multiple PKGs, but this inevitably entails extra communication and infrastructure. Related paradigms [2,18] strived to remove the key escrow problem but only did so at the expense of losing the benefit of human-memorizable public keys: these models get rid of escrow authorities but both involve traditional (though not explicitly certified) public keys that are usually less convenient to work with than easy-to-remember public identifiers.

In 2007, Goyal [21] explored a new approach to deter rogue actions from authorities. With the *Accountable Authority Identity-Based Encryption* (A-IBE) primitive, if the PKG discloses a decryption key associated with some identity over the Internet, it runs the risk of being caught and sued by the user. A-IBE schemes achieve this goal by means of an interactive private key generation protocol between the user and the PKG. For each identity, there are exponentially-many families of possible decryption keys. The key generation protocol provides the user with a single decryption key while concealing to the PKG the family that this key belongs to. From this private key, the user is computationally unable to find one from a different family. Hence, for a given identity, a pair of private keys from distinct families serves as evidence of a fraudulent PKG. The latter remains able to passively eavesdrop communications but is discouraged to reveal users' private keys. Also, users cannot falsely accuse an honest PKG since they are unable to compute a new key from a different family using a given key.

Prior Works. Two constructions were given in [21]. The first one (that we call *Goyal-1* hereafter) builds on Gentry's IBE [19] and, while efficient, only allows tracing well-formed decryption keys. This white-box model seems unlikely to suffice in practice since malicious parties can rather release an imperfect and/or obfuscated program that only decrypts with small but noticeable probability. The second scheme of [21] (let us call it *Goyal-2*), constructed on the Sahai-Waters fuzzy IBE [32], can be extended so as to provide weak black-box traceability: even an imperfect pirate decryption box can be traced (based on its input/output behavior) back to its source although traceability is only guaranteed against dishonest PKGs that have no decryption oracle in the attack game. However, *Goyal-2* is somewhat inefficient as decryption requires a number of pairing calculations that is linear in the security parameter. For the usually required security level, ciphertexts contain more than 160 group elements and decryption calculates a product of about 160 pairings.

Subsequently, Au *et al.* [3] described another A-IBE scheme providing retrievability (*i.e.*, a property that prevents the PKG from revealing more than one key for a given identity without exposing its master key) but remained in the white-box model. More recently, Goyal *et al.* [22] modified the *Goyal-2* system using attribute-based encryption techniques [23,32] to achieve full black-box traceability: unlike *Goyal-2*, the scheme of [22] preserves security against dishonest PKGs that have access to a decryption oracle in the model. While definitely desirable, this property is currently only achievable at the expense of the same significant penalty as in *Goyal-2* [21] in decryption cost and ciphertext size.

Our Contributions. We present a very efficient and conceptually simple scheme with weak black-box traceability. We prove its security (in the standard model) under the same assumption as *Goyal-2*. Decryption keys and ciphertexts consist of a constant number of group elements and their length is thus linear in the security parameter λ (instead of quadratic as in *Goyal-2*). Encryption and decryption take $O(\lambda^3)$-time (w.r.t. $O(\lambda^4)$ in *Goyal-2*) with only two pairing computations as for the latter (against more than 160 in *Goyal-2*).

While presented in the selective-ID security model (where adversaries must choose the identity that will be their prey at the outset of the game) for simplicity, our scheme is easily adaptable to the adaptive-ID model of [7]. In contrast, one of the security properties (*i.e.*, the infeasibility for users to frame innocent PKGs) was only established in the selective-ID setting for known schemes in the black-box model (*i.e.*, *Goyal-2* and its fully black-box extension [22]). Among such schemes, ours thus appears to be the first one that can be tweaked so as to achieve adaptive-ID security against dishonest users.

Our scheme performs almost as well as *Goyal-1* (the main overhead being a long master public key *à la* Waters [38] to obtain the adaptive-ID security). In comparison with the latter, that was only analyzed in a white-box model of traceability, our system provides several other advantages:

- Its security relies on a weaker assumption. So far, the only fully practical A-IBE scheme was resting on assumptions whose strength grows with the number of adversarial queries, which can be as large as 2^{30} as commonly assumed in the literature. Such assumptions are subject to a limited attack [14] that requires a careful adjustment of group sizes (by as much as 50% additional bits) to guarantee a secure use of schemes.
- It remains secure when many users want to run the key generation protocol in a concurrent fashion. *Goyal-1* has a key generation protocol involving zero-knowledge proofs. As its security reductions require to rewind adversaries at each key generation query, security is only guaranteed when the PKG interacts with users sequentially. In inherently concurrent environments like the Internet, key generation protocols should remain secure when executed by many users willing to register at the same time. By minimizing the number of rewinds in reductions, we ensure that our scheme remains secure in a concurrent setting. In these regards, the key generation protocol of *Goyal-2* makes use of oblivious transfers (OT) in sub-protocols. It thus supports concurrency whenever the underlying OT protocol does. As already mentioned however, our scheme features a much better efficiency than *Goyal-2*.
- In a white-box model of traceability, it can be made secure against dishonest PKGs equipped with a decryption oracle[1]. In the following, we nevertheless focus on the (arguably more interesting) weak black-box traceability aspect.

[1] We believe that the *Goyal-1* system can also be modified so as to obtain this property.

Organization. In the rest of the paper, section 2 recalls the A-IBE security model defined in [21]. We first analyze the white-box version of our scheme in section 3 and then describe a weak black-box tracing mechanism in section 4.

2 Background and Definitions

Syntactic definition and security model. We recall the definition of A-IBE schemes and their security properties as defined in [21].

Definition 1. *An* Accountable Authority Identity-Based Encryption scheme *(A-IBE) is a tuple (***Setup**, **Keygen**, **Encrypt**, **Decrypt**, **Trace**) *of efficient algorithms or protocols such that:*

- **Setup** *takes as input a security parameter and outputs a master public key* mpk *and a matching* master secret key msk.
- $\mathbf{Keygen}^{(\mathrm{PKG},\mathsf{U})}$ *is an interactive protocol between the public parameter generator* PKG *and the user* U*:*
  - · *the common input to* PKG *and* U *are: the master public key* mpk *and an identity* ID *for which the decryption key has to be generated;*
  - · *the private input to* PKG *is the master secret key* msk.

  Both parties may use a sequence of private coin tosses as additional inputs. The protocol ends with U *receiving a decryption key* d_{ID} *as his private output.*
- **Encrypt** *takes as input the master public key* mpk*, an identity* ID *and a message* m *and outputs a* ciphertext*.*
- **Decrypt** *takes as input the master public key* mpk*, a decryption key* d_{ID} *and a ciphertext* C *and outputs a message.*
- **Trace** *given the master public key* mpk*, a decryption key* d_{ID}*, this algorithm outputs a* key family number n_F *or the special symbol* $\perp$ *if* d_{ID} *is ill-formed.*

Correctness requires that, for any outputs $(\mathsf{mpk}, \mathsf{msk})$ *of* **Setup***, any plaintext* m *and any identity* ID*, whenever* $d_{\mathsf{ID}} \leftarrow \mathbf{Keygen}^{(\mathrm{PKG}(\mathsf{msk}),\mathsf{U})}(\mathsf{mpk}, \mathsf{ID})$*, we have*

$$\mathbf{Trace}\big(\mathsf{mpk}, d_{\mathsf{ID}}\big) \neq \perp,$$
$$\mathbf{Decrypt}\big(\mathsf{mpk}, d_{\mathsf{ID}}, \mathbf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, m)\big) = m.$$

The above definition is for the white-box setting. In a black-box model, **Trace** takes as input an identity ID, the corresponding user's well-formed private key d_{ID} and a decryption box $\mathbb{D}$ that successfully opens a non-negligible fraction ε of ciphertexts encrypted under ID. The output of **Trace** is either "PKG" or "User" depending on which party is found guilty for having crafted $\mathbb{D}$.

Goyal formalized three security properties for A-IBE schemes. The first one is the standard notion of privacy [7] for IBE systems. As for the other ones, the FindKey game captures the intractability for the PKG to create a decryption key of the same family as the one obtained by the user during the key generation protocol. Finally, the ComputeNewKey game models the infeasibility for users to generate a key $d_{\mathsf{ID}}^{(2)}$ outside the family of the legally obtained one $d_{\mathsf{ID}}^{(1)}$.

Definition 2. *An A-IBE scheme is deemed secure if all probabilistic polynomial time (PPT) adversaries have negligible advantage in the following games.*

1. **The IND-ID-CCA game.** For any PPT algorithm $\mathcal{A}$, the model considers the following game, where $\lambda \in \mathbb{N}$ is a security parameter:

```
$\boxed{\mathbf{Game}_{\mathcal{A}}^{\text{IND-ID-CCA}}(\lambda)}$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(\lambda)$
$(m_0, m_1, \mathsf{ID}^\star, s) \leftarrow \mathcal{A}^{\mathsf{Dec},\mathsf{KG}}(\mathsf{find}, \mathsf{mpk})$
                  | $\mathsf{Dec} : (C, \mathsf{ID})$
                  |   $\dashrightarrow \mathbf{Decrypt}\big(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}, C\big);$
                  | $\mathsf{KG} : \mathsf{ID} \dashrightarrow \mathbf{Keygen}^{(\mathrm{PKG}(\mathsf{msk}),\mathcal{A})}(\mathsf{mpk}, \mathsf{ID})$
                  |           $//\quad \mathsf{ID} \neq \mathsf{ID}^\star$
$d^\star \overset{\$}{\leftarrow} \{0, 1\}$
$C^\star \leftarrow \mathbf{Encrypt}(\mathsf{mpk}, \mathsf{ID}^\star, m_{d^\star})$
$d \leftarrow \mathcal{A}^{\mathsf{Dec},\mathsf{KG}}(\mathsf{guess}, s, C^\star)$
      | $\mathsf{Dec} : (C, \mathsf{ID}) \dashrightarrow \mathbf{Decrypt}\big(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}, C\big);$
      |               $//\quad (C, \mathsf{ID}) \neq (C^\star, \mathsf{ID}^\star)$
      | $\mathsf{KG} : \mathsf{ID} \dashrightarrow \mathbf{Keygen}^{(\mathrm{PKG}(\mathsf{msk}),\mathcal{A})}(\mathsf{mpk}, \mathsf{ID})$
      |          $//\quad \mathsf{ID} \neq \mathsf{ID}^\star$
return 1 if $d = d^\star$ and 0 otherwise.
```

$\mathcal{A}$'s advantage is measured by $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{CCA}}(\lambda) = |\Pr[\mathbf{Game}_{\mathcal{A}}^{\mathrm{CCA}} = 1] - 1/2|$.

The weaker definition of chosen-plaintext security (IND-ID-CPA) is formalized in the same way in [7] but $\mathcal{A}$ is not granted access to a decryption oracle.

2. **The FindKey game.** Let $\mathcal{A}$ be a PPT algorithm. We consider the following game, where $\lambda \in \mathbb{N}$ is a security parameter:

```
$\boxed{\mathbf{Game}_{\mathcal{A}}^{\mathrm{FindKey}}(\lambda)}$
$(\mathsf{mpk}, \mathsf{ID}, s_1) \leftarrow \mathcal{A}(\mathsf{setup}, \lambda)$
$(d_{\mathsf{ID}}^{(1)}, s_2) \leftarrow \mathbf{Keygen}^{(\mathcal{A}(s_1),\cdot)}(\mathsf{mpk}, \mathsf{ID})$
$d_{\mathsf{ID}}^{(2)} \leftarrow \mathcal{A}(\mathsf{findkey}, s_1, s_2)$
return 1 if $\mathbf{Trace}(\mathsf{mpk}, d_{\mathsf{ID}}^{(1)}) = \mathbf{Trace}(\mathsf{mpk}, d_{\mathsf{ID}}^{(2)})$
         0 otherwise.
```

$\mathcal{A}$'s advantage is now defined as $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{FindKey}}(\lambda) = \Pr[\mathbf{Game}_{\mathcal{A}}^{\mathrm{FindKey}} = 1]$.

Here, the adversary $\mathcal{A}$ acts as a cheating PKG and the challenger emulates the honest user. Both parties engage in a key generation protocol where the challenger obtains a private key for an identity ID chosen by $\mathcal{A}$. The latter aims at producing a private key corresponding to ID and belonging to the same family as the key obtained by the challenger in the key generation protocol. Such a successful dishonest PKG could disclose user keys without being caught.

Note that, at the beginning of the experiment, $\mathcal{A}$ generates mpk without revealing the master key msk and the challenger runs a sanity check on mpk.

As noted in [21], it makes sense to provide $\mathcal{A}$ with a decryption oracle that undoes ciphertexts using $d_{\mathsf{ID}}^{(1)}$ (and could possibly leak information on the latter's family) between steps 2 and 3 of the game. We call this enhanced notion FindKey-CCA (as opposed to the weaker one which we call FindKey-CPA).

Finally, in the black-box model, instead of outputting a new key $d_{\mathsf{ID}}^{(2)}$, the dishonest PKG comes up with a decryption box $\mathbb{D}$ that correctly decrypts ciphertexts intended for ID with non-negligible probability ε and wins if the tracing algorithm returns “User” when run on $d_{\mathsf{ID}}^{(1)}$ and with oracle access to $\mathbb{D}$.

3. **The ComputeNewKey game.** For a PPT algorithm $\mathcal{A}$, the model finally considers the following game:

```
$\boxed{\mathbf{Game}_{\mathcal{A}}^{\mathrm{ComputeNewKey}}(\lambda)}$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(\lambda)$
$(d_{\mathsf{ID}^\star}^{(1)}, d_{\mathsf{ID}^\star}^{(2)}, \mathsf{ID}^\star) \leftarrow \mathcal{A}^{\mathsf{KG}}(\mathsf{mpk})$
                    $|\ \mathsf{KG} : \mathsf{ID} \dashrightarrow \mathbf{Keygen}^{(\mathrm{PKG}(\mathsf{msk}),\mathcal{A})}(\mathsf{mpk}, \mathsf{ID})$
return 1 if $\mathbf{Trace}(\mathsf{mpk}, d_{\mathsf{ID}^\star}^{(1)}) \neq \perp$ and
             $\mathbf{Trace}(\mathsf{mpk}, d_{\mathsf{ID}^\star}^{(2)}) \notin \{\perp, \mathbf{Trace}(\mathsf{mpk}, d_{\mathsf{ID}^\star}^{(1)})\}$
       0 otherwise.
```

$\mathcal{A}$'s advantage is $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{ComputeNewKey}}(\lambda) = \Pr[\mathbf{Game}_{\mathcal{A}}^{\mathrm{ComputeNewKey}} = 1]$.

The ComputeNewKey game involves an adversary interacting with a PKG in executions of the key generation protocol and obtaining private keys associated with *distinct* identities of her choosing. The adversary is declared successful if, for some identity that may have been queried for key generation, she is able to find *two* private keys from *distinct* families. Such a pair would allow her to trick a judge into wrongly believing in a misbehavior of the PKG.

In the black-box scenario, the output of the dishonest user consist of a key $d_{\mathsf{ID}^\star}^{(1)}$ and a pirate decryption box $\mathbb{D}$ that yields the correct answer with probability ε when provided with a ciphertext encrypted for $\mathsf{ID}^\star$. In this case, the adversary wins if the output of $\mathbf{Trace}^{\mathbb{D}}(\mathsf{mpk}, d_{\mathsf{ID}^\star}^{(1)})$ is “PKG”.

In [12], Canetti, Halevi and Katz suggested relaxed notions of IND-ID-CCA and IND-ID-CPA security where the adversary has to choose the target identity $\mathsf{ID}^\star$ ahead of time (even before seeing the master public key mpk). This relaxed model, called “selective-ID” model (or IND-sID-CCA and IND-sID-CPA for short), can be naturally extended to the ComputeNewKey notion.

Bilinear Maps and Complexity Assumptions. We use prime order groups $(\mathbb{G}, \mathbb{G}_T)$ endowed with an efficiently computable map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that:

1. $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

In such *bilinear groups*, we assume the hardness of the (now classical) Decision Bilinear Diffie-Hellman problem that has been widely used in the recent years.

Definition 3. *Let $(\mathbb{G}, \mathbb{G}_T)$ be bilinear groups of prime order p and $g \in \mathbb{G}$. The* **Decision Bilinear Diffie-Hellman Problem** *(DBDH) is to distinguish the distributions of tuples $(g^a, g^b, g^c, e(g,g)^{abc})$ and $(g^a, g^b, g^c, e(g,g)^z)$ for random values $a, b, c, z \xleftarrow{\$} \mathbb{Z}_p^*$. The advantage of a distinguisher $\mathcal{B}$ is measured by*

$$\mathbf{Adv}^{\mathrm{DBDH}}_{\mathbb{G},\mathbb{G}_T}(\lambda) = \big|\Pr[a,b,c \xleftarrow{\$} \mathbb{Z}_p^* : \mathcal{B}(g^a, g^b, g^c, e(g,g)^{abc}) = 1] \\ - \Pr[a,b,c,z \xleftarrow{\$} \mathbb{Z}_p^* : \mathcal{B}(g^a, g^b, g^c, e(g,g)^{z}) = 1]\big|.$$

For convenience, we use an equivalent formulation – called *modified* DBDH – of the problem which is to distinguish $e(g,g)^{ab/c}$ from random given (g^a, g^b, g^c).

3 The Basic Scheme

The scheme mixes ideas from the "commutative-blinding" [4] and "exponent-inversion" [33] frameworks. Private keys have the same shape as in commutative-blinding-based schemes [4,5,11,38]. At the same time, their first element is a product of two terms, the first one of which is inspired from Gentry's IBE [19].

According to a technique applied in [21], private keys contain a family number t that cannot be tampered with while remaining hidden from the PKG. This family number t is determined by combining two random values t_0 and t_1 respectively chosen by the user and the PKG in the key generation protocol. The latter begins with the user sending a commitment R to t_0. Upon receiving R, the PKG turns it into a commitment to $t_0 + t_1$ and uses the modified commitment to generate a "blinded" private key d'_{ID}. The user obtains his final key d_{ID} by "unblinding" d'_{ID} thanks to the randomness that was used to compute R.

A difference with *Goyal-1* is that the key family number is perfectly hidden to the PKG and the FindKey-CPA security is unconditional. In the key generation protocol, the user's first message is a perfectly hiding commitment that comes along with a witness-indistinguishable (WI) proof of knowledge of its opening. In *Goyal-1*, users rather send a deterministic (and thus non-statistically hiding) commitment and knowledge of the underlying value must be proven in zero-knowledge because a proof of knowledge of a discrete logarithm must be simulated (by rewinding the adversary) in the proof of FindKey-CPA security. In the present scheme, the latter does not rely on a specific assumption and we do not need to simulate knowing the solution of a particular problem instance. Therefore, we can dispense with perfectly ZK proofs and settle for a more efficient 3-move WI proof (such as Okamoto's variant [30] of Schnorr [35]) whereas 4 rounds are needed[2] using zero-knowledge proofs of knowledge.

[2] A similar modification can be brought to the key generation protocol of *Goyal-1* to statistically hide the key family number to the PKG and avoid the need for 4-round ZK proofs.

3.1 Description

Setup: given $\lambda \in \mathbb{N}$, the PKG selects bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with a random generator $g \xleftarrow{\$} \mathbb{G}$. It chooses $h, Y, Z \xleftarrow{\$} \mathbb{G}$ and $x \xleftarrow{\$} \mathbb{Z}_p^*$ at random. It defines its master key as $\mathsf{msk} := x$ and the master public key is chosen as $\mathsf{mpk} := (X = g^x, Y, Z, h)$.

Keygen$^{(\mathsf{PKG},\mathsf{U})}$: to obtain a private key for his identity ID, a user U interacts with the PKG in the following key generation protocol.

1. The user U draws $t_0, \theta \xleftarrow{\$} \mathbb{Z}_p^*$, provides the PKG with a commitment $R = h^{t_0} \cdot X^\theta$ and also gives an interactive witness indistinguishable proof of knowledge of the pair (t_0, θ), which he retains for later use.
2. The PKG outputs $\perp$ if the proof of knowledge fails to verify. Otherwise, it picks $r', t_1 \xleftarrow{\$} \mathbb{Z}_p^*$ and returns
$$d'_{\mathsf{ID}} = (d'_1, d'_2, d'_3) = \Big((Y \cdot R \cdot h^{t_1})^{1/x} \cdot (g^{\mathsf{ID}} \cdot Z)^{r'},\ X^{r'},\ t_1 \Big). \qquad (1)$$
3. U picks $r'' \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $d_{\mathsf{ID}} = (d'_1/g^\theta \cdot (g^{\mathsf{ID}} \cdot Z)^{r''},\ d'_2 \cdot X^{r''},\ d'_3 + t_0)$ which should equal
$$d_{\mathsf{ID}} = (d_1, d_2, d_3) = \Big((Y \cdot h^{t_0+t_1})^{1/x} \cdot (g^{\mathsf{ID}} \cdot Z)^{r},\ X^{r},\ t_0 + t_1 \Big) \qquad (2)$$
where $r = r' + r''$. Then, U checks whether d_{ID} satisfies the relation
$$e(d_1, X) = e(Y, g) \cdot e(h, g)^{d_3} \cdot e(g^{\mathsf{ID}} \cdot Z, d_2). \qquad (3)$$
If so, he sets his private key as d_{ID} and the latter belongs to the family of decryption keys identified by $n_F = d_3 = t_0 + t_1$. He outputs $\perp$ otherwise.

Encrypt: to encrypt $m \in \mathbb{G}_T$ given mpk and ID, choose $s \xleftarrow{\$} \mathbb{Z}_p^*$ and compute
$$C = \big(C_1, C_2, C_3, C_4\big) = \Big(X^s,\ (g^{\mathsf{ID}} \cdot Z)^s,\ e(g, h)^s,\ m \cdot e(g, Y)^s \Big).$$

Decrypt: given $C = \big(C_1, C_2, C_3, C_4\big)$ and $d_{\mathsf{ID}} = (d_1, d_2, d_3)$, compute
$$m = C_4 \cdot \Big(\frac{e(C_1, d_1)}{e(C_2, d_2) \cdot C_3^{d_3}} \Big)^{-1} \qquad (4)$$

Trace: given a purported private key $d_{\mathsf{ID}} = (d_1, d_2, d_3)$ and an identity ID, check the validity of d_{ID} w.r.t. ID using relation (3). If valid, d_{ID} is declared as a member of the family identified by $n_F = d_3$.

The correctness of the scheme follows from the fact that well-formed private keys always satisfy relation (3). By raising both members of (3) to the power $s \in \mathbb{Z}_p^*$, we see that the quotient of pairings in (4) actually equals $e(g, Y)^s$.

The scheme features about the same efficiency as classical IBE schemes derived from the commutative-blinding framework [4]. Encryption demands no pairing calculation since $e(g, h)$ and $e(g, Y)$ can both be cached as part of the system parameters. Decryption requires to compute a quotient of two pairings which is significantly faster than two independent pairing evaluations when optimized in the same way as modular multi-exponentiations.

In comparison with the most efficient standard model scheme based on the same assumption (which is currently the first scheme of [4]), the only overhead is a slightly longer ciphertext and an extra exponentiation in $\mathbb{G}_T$ at both ends.

3.2 Security

Selective-ID Security. We first prove the IND-sID-CPA security under the modified DBDH assumption (mDBDH).

Theorem 1. *The scheme is IND-sID-CPA under the mDBDH assumption.*

Proof. We show how a simulator $\mathcal{B}$ can interact with a selective-ID adversary $\mathcal{A}$ to solve a mDBDH instance $(T_a = g^a, T_b = g^b, T_c = g^c, T \overset{?}{=} e(g,g)^{ab/c})$. At the outset of the game, $\mathcal{A}$ announces the target identity $\mathsf{ID}^\star$. To prepare mpk, $\mathcal{B}$ chooses $\alpha, \gamma, t^* \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and sets $X = T_c = g^c$, $h = T_b = g^b$, $Y = X^\gamma \cdot h^{-t^*}$, and $Z = g^{-\mathsf{ID}^\star} \cdot X^\alpha$. The adversary's view is simulated as follows.

Queries: at any time, $\mathcal{A}$ may trigger an execution of the key generation protocol for an identity $\mathsf{ID} \neq \mathsf{ID}^\star$ of her choosing. She then supplies an element $R = h^{t_0} \cdot X^\theta$ along with a WI proof of knowledge of (t_0, θ). The simulator $\mathcal{B}$ verifies the proof but does not need to rewind the adversary as it can answer the query without knowing (t_0, θ). To do so, it picks $t_1 \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ at random and defines $W = Y \cdot R \cdot h^{t_1}$, $d_3' = t_1$. Elements d_1' and d_2' are generated as

$$(d_1', d_2') = \Big((g^{\mathsf{ID}} \cdot Z)^{r'} \cdot W^{-\frac{\alpha}{\mathsf{ID}-\mathsf{ID}^\star}},\ X^{r'} \cdot W^{-\frac{1}{\mathsf{ID}-\mathsf{ID}^\star}} \Big) \tag{5}$$

using a random $r' \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. If we set $\tilde{r}' = r' - \frac{w}{c(\mathsf{ID}-\mathsf{ID}^\star)}$, where $w = \log_g(W)$, we observe that (d_1', d_2') has the correct distribution since

$$\begin{aligned} W^{1/c} \cdot (g^{\mathsf{ID}} \cdot Z)^{\tilde{r}'} &= W^{1/c} \cdot (g^{\mathsf{ID}-\mathsf{ID}^\star} \cdot X^\alpha)^{\tilde{r}'} \\ &= W^{1/c} \cdot (g^{\mathsf{ID}-\mathsf{ID}^\star} \cdot X^\alpha)^{r'} \cdot (g^{\mathsf{ID}-\mathsf{ID}^\star})^{-\frac{w}{c(\mathsf{ID}-\mathsf{ID}^\star)}} \cdot X^{-\frac{w\alpha}{c(\mathsf{ID}-\mathsf{ID}^\star)}} \\ &= (g^{\mathsf{ID}} \cdot Z)^{r'} \cdot W^{-\frac{\alpha}{\mathsf{ID}-\mathsf{ID}^\star}} \end{aligned}$$

and $X^{\tilde{r}'} = X^{r'} \cdot (g^c)^{-\frac{w}{c(\mathsf{ID}-\mathsf{ID}^\star)}} = X^{r'} \cdot W^{-\frac{1}{\mathsf{ID}-\mathsf{ID}^\star}}$. Finally, the "partial private key" (d_1', d_2', d_3') is returned to $\mathcal{A}$. Note that the above calculation can be carried out without knowing $w = \log_g(W)$ or the representation (t_0, θ) of R w.r.t. to (h, X) and $\mathcal{B}$ does *not* need to rewind $\mathcal{A}$.

Challenge: when the first stage is over, $\mathcal{A}$ outputs $m_0, m_1 \in \mathbb{G}_T$. At this point, $\mathcal{B}$ picks $r^\star \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and defines a private key $(d_1, d_2, d_3) = (g^\gamma \cdot X^{\alpha r^\star}, X^{r^\star}, t^*)$ for the identity ID^*. It flips a fair coin $d^\star \overset{\$}{\leftarrow} \{0,1\}$ and encrypts $m_{d^\star}$ as

$$C_1^\star = T_a = g^a \qquad C_2^\star = T_a^\alpha \qquad C_3^\star = T \qquad C_4^\star = m_{d^\star} \cdot \frac{e(C_1^\star, d_1)}{e(C_2^\star, d_2) \cdot C_3^{\star\, d_3}}.$$

We see that (d_1, d_2, d_3) is a valid key for $\mathsf{ID}^\star$. Since $g^{\mathsf{ID}^\star} \cdot Z = X^\alpha = T_c^\alpha$ and $h = g^b$, $C^\star = (C_1^\star, C_2^\star, C_3^\star, C_4^\star)$ is a valid encryption of $m_{d^\star}$ (with the exponent $s = a/c$) if $T = e(g,g)^{ab/c}$. If T is random, we have $T = e(g,h)^{s'}$ for some random $s' \in \mathbb{Z}_p^*$ and thus $C_4^\star = m_{d^\star} \cdot e(Y,g)^s \cdot e(g,h)^{(s-s')t^*}$, which means that $m_{d^\star}$ is perfectly hidden since $t^\star$ is independent of $\mathcal{A}$'s view.

As usual, $\mathcal{B}$ outputs 1 (meaning that $T = e(g,g)^{ab/c}$) if $\mathcal{A}$ successfully guesses $d' = d^\star$ and 0 otherwise. □

In the above proof, the simulator does not rewind the adversary at any time. The scheme thus remains IND-sID-CPA in concurrent environments, where a batch of users may want to simultaneously run the key generation protocol.

Also, the simulator knows a valid private key for each identity. This allows using Cramer-Shoup-like techniques [16,17] as in [19,26] to secure the scheme against chosen-ciphertext attacks. The advantage of this approach, as we show in appendix A, is to provide FindKey-CCA security in a white-box setting.

Unlike the *Goyal-1* scheme, the basic system provides unconditional FindKey-CPA security: after an execution of the key generation protocol, even an all powerful PKG does not have any information on the component d_3 that is eventually part of the private key obtained by the new user.

Theorem 2. *In the information theoretic sense, no adversary has an advantage in the* FindKey-CPA *game.*

Proof. The proof directly follows from the perfect hiding property of Pedersen's commitment [31] and the perfect witness indistinguishability of the protocol [30] for proving knowledge of a discrete logarithm representation. Since the commitment $R = h^{t_0} \cdot X^\theta$ and the proof of knowledge of (t_0, θ) perfectly hide t_0 to the PKG, all elements of $\mathbb{Z}_p^*$ are equally likely values of $d_3 = t_0 + t_1$ as for the last part of the user's eventual private key. □

Appendix A describes a hybrid variant of the scheme that provides white-box FindKey-CCA security using authenticated symmetric encryption in the fashion of [27,37,24] so as to reject all invalid ciphertexts with high probability.

Theorem 3. *In the selective-ID* ComputeNewKey *game, any PPT adversary has negligible advantage assuming that the Diffie-Hellman assumption holds.*

Proof. For simplicity, we prove the result using an equivalent formulation of the Diffie-Hellman problem which is to find $h^{1/x}$ given $(g, h, X = g^x)$.

At the outset of the game, $\mathcal{A}$ declares the identity $\mathsf{ID}^\star$ for which she aims at finding two private keys $d_{\mathsf{ID}^\star}^{(1)}$, $d_{\mathsf{ID}^\star}^{(2)}$ comprising distinct values of $d_3 = t$. Then, the simulator $\mathcal{B}$ prepares the PKG's public key as follows. Elements h and X are taken from the modified Diffie-Hellman instance (g, h, X). As in the proof of theorem 1, $\mathcal{B}$ defines $Z = g^{-\mathsf{ID}^\star} \cdot X^\alpha$ for a randomly chosen $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$. To define Y, it chooses random values $\gamma, t_1' \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $Y = X^\gamma \cdot h^{-t_1'}$.

Queries: in this game, $\mathcal{A}$ is allowed to query executions of the key generation protocol w.r.t. any identity, including $\mathsf{ID}^\star$. The only requirement is that queried identities be distinct.

- For an identity $\mathsf{ID} \neq \mathsf{ID}^\star$, $\mathcal{B}$ can proceed *exactly* as suggested by relation (5) in the proof of theorem 1 and does not need to rewind $\mathcal{A}$.
- When $\mathsf{ID} = \mathsf{ID}^\star$, $\mathcal{B}$ conducts the following steps. When $\mathcal{A}$ supplies a group element $R = h^{t_0} \cdot X^\theta$ along with a WI proof of knowledge of (t_0, θ), $\mathcal{B}$ uses the knowledge extractor of the proof of knowledge that allows extracting a representation (t_0, θ) of R by rewinding $\mathcal{A}$. Next, $\mathcal{B}$ computes $t_1 = t_1' - t_0$ picks $r \xleftarrow{\$} \mathbb{Z}_p^*$ and returns
$$(d_1', d_2', d_3') = \big(g^{\gamma+\theta} \cdot (g^{\mathsf{ID}^\star} \cdot Z)^r,\ X^r,\ t_1\big). \qquad (6)$$
To see that the above tuple has the appropriate shape, we note that
$$(Y \cdot R \cdot h^{t_1})^{1/x} = (Y \cdot h^{t_0+t_1} \cdot X^\theta)^{1/x} = (Y \cdot h^{t_1'} \cdot X^\theta)^{1/x} = g^{\gamma+\theta}.$$

Output: upon its termination, $\mathcal{A}$ is expected to come up with distinct valid private keys $d_{\mathsf{ID}^\star}^{(1)} = (d_1^{(1)}, d_2^{(1)}, d_3^{(1)})$ and $d_{\mathsf{ID}^\star}^{(2)} = (d_1^{(2)}, d_2^{(2)}, d_3^{(2)})$, such that $t = d_3^{(1)} \neq d_3^{(2)} = t'$, for the identity $\mathsf{ID}^\star$. Given that we must have
$$d_1^{(1)} = (Y \cdot h^t)^{1/x} \cdot X^{\alpha r} \qquad d_2^{(1)} = X^r$$
$$d_1^{(2)} = (Y \cdot h^{t'})^{1/x} \cdot X^{\alpha r'} \qquad d_2^{(2)} = X^{r'}$$
for some values $r, r' \in \mathbb{Z}_p$, $\mathcal{B}$ can extract $h^{1/x} = \left(\frac{d_1^{(1)}/d_2^{(1)\alpha}}{d_1^{(2)}/d_2^{(2)\alpha}}\right)^{\frac{1}{t-t'}}$. □

We note that, in the above proof, the simulator does not have to rewind all executions of the key generation protocol but only one, when the adversary asks for a private key corresponding to the target identity $\mathsf{ID}^\star$ (recall that all queries involve distinct identities). Given that the number of rewinds is constant, the proof still goes through when the simulator is presented with many concurrent key generation queries. If other executions of the protocol (that necessarily involve identities $\mathsf{ID} \neq \mathsf{ID}^\star$) are nested within the one being rewinded when dealing with $\mathsf{ID}^\star$, the simulator simply runs them as an honest verifier would in the proof of knowledge and calculates the PKG's output as per relation (5) in the proof of theorem 1. Thus, the initial rewind does not trigger any other one and the simulation still takes polynomial time in a concurrent setting.

Adaptive-ID Security. The scheme can obviously be made IND-ID-CPA if Waters' "hash function" $F(\mathsf{ID}) = u' \prod_{j=1}^n u_i^{i_j}$ – where $\mathsf{ID} = i_1 \dots i_n \in \{0,1\}^n$ and $(u', u_1, \dots, u_n) \in \mathbb{G}^{n+1}$ is part of mpk – supersedes the Boneh-Boyen identity hashing $F(\mathsf{ID}) = g^{\mathsf{ID}} \cdot Z$. The function F is chosen so as to equal $F(\mathsf{ID}) = g^{J_1(\mathsf{ID})} \cdot X^{J_2(\mathsf{ID})}$ for integer-valued functions J_1, J_2 that are computable by the simulator. The security proof relies on the fact that J_1 is small in absolute value and cancels with non-negligible probability proportional to $1/q(n+1)$, where q is the number of key generation queries.

When extending the proof of theorem 3 to the adaptive setting, an adversary with advantage ε allows solving CDH with probability $O(\varepsilon/q^2(n+1))$. The reason is that the simulator has to guess beforehand which key generation query will involve the target identity $\mathsf{ID}^\star$. If $\mathsf{ID}^\star$ is expected to appear in the j^{th} query, when the latter is made, $\mathcal{B}$ rewinds $\mathcal{A}$ to extract (t_0, θ) and uses the special value t_1' to answer the query as per (6). With probability $1/q$, $\mathcal{B}$ is fortunate when choosing $j \stackrel{\$}{\leftarrow} \{1, \ldots, q\}$ at the beginning and, again, $J_1(\mathsf{ID}^\star)$ happens to cancel with probability $O(1/q(n+1))$ for the target identity.

4 Weak Black-Box Traceability

Theorem 3 showed the infeasibility for users to compute another key from a different family given their private key. In these regards, a decryption key implements a "1-copyrighted function" – in the terminology of [25,29] – for the matching identity. Using this property and the perfect white-box FindKey-CPA security, we describe a black-box tracing mechanism that protects the user from a dishonest PKG as long as the latter is withheld access to a decryption oracle.

The tracing strategy is the one used by Kiayias and Yung [25] in 2-user traitor tracing schemes, where the tracer determines which one out of two subscribers produced a pirate decoder. In our setting, one rather has to decide whether an ε-useful decryption device stems from the PKG or the user himself.

Trace$^{\mathbb{D}}(\mathsf{mpk}, d_{\mathsf{ID}}, \varepsilon)$**:** given a well-formed private key $d_{\mathsf{ID}} = (d_1, d_2, d_3)$ belonging to a user of identity ID and oracle access to a decoder $\mathbb{D}$ that decrypts ciphertexts encrypted for ID with probability ε, conduct the following steps.

a. Initialize a counter *ctr* $\leftarrow 0$ and repeat the next steps $L = 16\lambda/\varepsilon$ times:
   1. Choose distinct exponents $s, s' \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ at random, compute $C_1 = X^s$, $C_2 = (g^{\mathsf{ID}} \cdot Z)^s$ and $C_3 = e(g, h)^{s'}$.
   2. Calculate $C_4 = m \cdot e(C_1, d_1)/\big(e(C_2, d_2) \cdot C_3^{d_3}\big)$ for a randomly chosen message $m \in \mathbb{G}_T$.
   3. Feed the decryption device $\mathbb{D}$ with (C_1, C_2, C_3, C_4). If $\mathbb{D}$ outputs $m' \in \mathbb{G}_T$ such that $m' = m$, increment *ctr*.

b. If $ctr < 4\lambda$, incriminate the PKG. Otherwise, incriminate the user.

The soundness of this algorithm is proved using a similar technique to [1]. To ensure the independence of iterations, we assume (as in [1]) that pirate devices are stateless, or resettable, and do not retain information from prior queries: each query is answered as if it were the first one.

Theorem 4. *Under the mDBDH assumption, dishonest users have negligible chance to produce a decryption device $\mathbb{D}$ that makes the tracing algorithm incriminate the PKG in the selective-ID ComputeNewKey game.*

Proof. The tracing algorithm points to the PKG if it ends up with a too small value of *ctr*. The latter can be seen as the sum of $L = 16\lambda/\varepsilon$ independent random variables $X_i \in \{0, 1\}$ having the same expected value p_1. We have $\mu = \mathbf{E}[ctr] = Lp_1$. The Chernoff bound tells us that, for any real number ω such that $0 \leq \omega \leq 1$, $\Pr[ctr < (1-\omega)\mu] < \exp(-\mu\omega^2/2)$. Under the mDBDH assumption, we certainly have $\mathbf{Adv}^{\mathrm{mDBDH}}(\lambda) \leq \varepsilon/2$ (since $\varepsilon/2$ is presumably non-negligible). Lemma 1 shows that $p_1 \geq \varepsilon - \mathbf{Adv}^{\mathrm{mDBDH}}(\lambda)$, which implies

$$\mu = Lp_1 \geq L(\varepsilon - \mathbf{Adv}^{\mathrm{mDBDH}}(\lambda)) \geq \frac{L\varepsilon}{2} = 8\lambda. \tag{7}$$

With $\omega = 1/2$, the Chernoff bound guarantees that

$$\Pr[ctr < 4\lambda] = \Pr[ctr < \mu/2] < \exp(-\mu/8) = \exp(-\lambda). \qquad \square$$

Lemma 1. *In the selective-ID ComputeNewKey game, if $\mathbb{D}$ correctly opens well-formed ciphertexts with probability ε, the probability that an iteration of the tracing algorithm increases ctr is at least $p_1 \geq \varepsilon - \mathbf{Adv}^{\mathsf{mDBDH}}(\lambda)$.*

Proof. We consider two games called Game$_0$ and Game$_1$ where the adversary $\mathcal{A}$ is faced with a ComputeNewKey challenger $\mathcal{B}$ and produces a decryption device $\mathbb{D}$ which is provided with ciphertexts during a tracing stage. In Game$_0$, $\mathbb{D}$ is given a properly formed encryption of some plaintext m whereas it is given a ciphertext C where C_3 has been changed in Game$_1$. In either case, we call p_i (with $i \in \{0, 1\}$) the probability that $\mathbb{D}$ returns the plaintext m chosen by $\mathcal{B}$.

In the beginning of Game$_0$, $\mathcal{A}$ chooses a target identity $\mathsf{ID}^\star$ and the challenger $\mathcal{B}$ defines the system parameters as $X = g^c$, $h = g^b$, $Y = X^\gamma \cdot h^{-t^\star}$ and $Z = g^{-\mathsf{ID}^\star} \cdot X^\alpha$ for random $\alpha, \gamma, t^\star \xleftarrow{\$} \mathbb{Z}_p^*$. Then, $\mathcal{A}$ starts making key generation queries that are treated using the same technique as in the proof of theorem 3. Again, $\mathcal{B}$ only has to rewind the WI proof when the query pertains to $\mathsf{ID}^\star$.

At the end of the game, $\mathcal{A}$ outputs a decryption box $\mathbb{D}$ that correctly decrypts a fraction ε of ciphertexts. Then, $\mathcal{B}$ constructs a ciphertext C as

$$C_1 = g^a, \qquad C_2 = (g^a)^\alpha, \qquad C_3 = T, \qquad C_4 = m \cdot \frac{e(C_1, d_1)}{e(C_2, d_2) \cdot C_3^{t^\star}}$$

where $T \in \mathbb{G}_T$. In Game$_0$, $\mathcal{B}$ sets $T = e(g,g)^{ab/c}$ so that we have $C_3 = e(g,h)^{a/c}$ and C is a valid ciphertext (for the encryption exponent $s = a/c$) that $\mathbb{D}$ correctly decrypts with probability ε. In this case, $\mathbb{D}$ thus outputs $m' = m \in \mathbb{G}_T$ with probability $p_0 = \varepsilon$. In Game$_1$, T is chosen as a random element of $\mathbb{G}_T$ and $C = (C_1, C_2, C_3, C_4)$ has the distribution of a ciphertext produced by the tracing stage and $\mathbb{D}$ must output a plaintext $m' = m$ with probability p_1. It is clear that $|p_0 - p_1| \leq \mathbf{Adv}^{\mathrm{mDBDH}}(\lambda)$ and we thus have $p_1 \geq \varepsilon - \mathbf{Adv}^{\mathrm{mDBDH}}(\lambda)$. $\square$

The proofs of theorem 4 and lemma 1 readily extend to the adaptive-ID setting using the same arguments as in the last paragraph of section 3. The system thus turns out to be the first scheme that is amenable for weak black-box traceability against dishonest users in the adaptive-ID sense. Due to their reliance on attribute-based encryption techniques (for which only selective-ID adversaries were dealt with so far), earlier black-box or weakly black-box A-IBE proposals [21,22] are only known to provide selective-ID security against dishonest users.

As for the security against dishonest PKGs, we observed that, in the FindKey-CPA game, the last part $d_3^{(1)} = t$ of the user's private key is perfectly hidden to the malicious PKG after the key generation protocol. Then, a pirate decoder $\mathbb{D}$ made by the PKG has negligible chance of decrypting ciphertexts where C_3 is random in the same way as the user would. When the user comes across $\mathbb{D}$ and takes it to the court, the latter runs the tracing algorithm using $\mathbb{D}$ and the user's well-formed key $d_{\mathsf{ID}}^{(1)} = (d_1^{(1)}, d_2^{(1)}, d_3^{(1)})$ for which $d_3^{(1)}$ is independent of $\mathbb{D}$.

Lemma 2. *In the FindKey-CPA game, one iteration of the tracing algorithm increases ctr with probability at most* $1/p$.

Proof. In an iteration of the tracing stage, $\mathbb{D}$ is given $C = (C_1, C_2, C_3, C_4)$ such that $C_1 = X^s$, $C_2 = (g^{\mathsf{ID}} \cdot Z)^s$, $C_3 = e(g, h)^{s'}$ and $C_4 = m \cdot e(g, Y)^s \cdot e(g, h)^{(s-s')t}$ for distinct $s, s' \xleftarrow{\$} \mathbb{Z}_p^*$. Since $\mathbb{D}$ has no information on $d_3^{(1)} = t$, for any plaintext $m \in \mathbb{G}_T$, there is a value $d_3^{(1)}$ that explains C_4 and it comes that $\mathbb{D}$ returns the one chosen by the tracer with probability $1/p$. □

Theorem 5. *In the black-box FindKey-CPA game, a dishonest PKG has negligible advantage.*

Proof. The dishonest PKG is not detected if it outputs a decryption box for which the tracing ends with a sufficiently large *ctr*. From lemma 2, it easily comes that $\Pr[ctr \geq 4\lambda] \leq \Pr[ctr \geq 1] \leq L/p = 16\lambda/(\varepsilon p) \leq 16\lambda/(2^\lambda \varepsilon)$. □

To secure the scheme against chosen-ciphertext attacks and preserve the weak black-box property, we can use the Canetti-Halevi-Katz [13] technique or its optimizations [9,10] that do not affect the tracing algorithm.

5 Conclusion

We described the first A-IBE system allowing for weak black-box traceability while retaining short ciphertexts and private keys. We also suggested a white-box variant that dwells secure against dishonest PKGs equipped with a decryption oracle. In the black-box setting, it remains an open problem to achieve the latter property without significantly degrading the efficiency.

Acknowledgements

We thank Duong Hieu Phan and the anonymous referees for their comments.

References

1. Abdalla, M., Dent, A., Malone-Lee, J., Neven, G., Phan, D.-H., Smart, N.: Identity-Based Traitor Tracing. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 361–376. Springer, Heidelberg (2007)
2. Al-Riyami, S., Paterson, K.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)

3. Au, M.-H., Huang, Q., Liu, J.-K., Susilo, W., Wong, D.-S., Yang, G.: Traceable and Retrievable Identity-Based Encryption. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 94–110. Springer, Heidelberg (2008)
4. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Boyen, X.: Secure Identity-Based Encryption Without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity-Based encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
7. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. SIAM 32(3), 586–615 (2003); earlier version in Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
8. Boneh, D., Gentry, C., Hamburg, M.: Space-Efficient Identity-Based Encryption Without Pairings. In: FOCS 2007, pp. 647–657 (2007)
9. Boneh, D., Katz, J.: Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
10. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security from Identity-Based Techniques. In: ACM CCS 2005, pp. 320–329 (2005)
11. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
12. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 254–271. Springer, Heidelberg (2003)
13. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
14. Cheon, J.H.: Security Analysis of the Strong Diffie-Hellman Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
15. Cocks, C.: An Identity-Based Encryption Scheme Based on Quadratic Residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
16. Cramer, R., Shoup, V.: A Practical Public-Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
17. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
18. Gentry, C.: Certificate-Based Encryption and the Certificate Revocation Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
19. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)

20. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
21. Goyal, V.: Reducing Trust in the PKG in Identity-Based Cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg (2007)
22. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-Box Accountable Authority Identity Based Encryption. In: ACM-CCS 2008 (2008)
23. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98 (2006)
24. Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
25. Kiayias, A., Yung, M.: Traitor Tracing with Constant Transmission Rate. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 450–465. Springer, Heidelberg (2002)
26. Kiltz, E., Vahlis, Y.: CCA2 Secure IBE: Standard Model Efficiency through Authenticated Symmetric Encryption. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 221–238. Springer, Heidelberg (2008)
27. Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 445–456. Springer, Heidelberg (2004)
28. Libert, B., Vergnaud, D.: Towards Black-Box Accountable Authority IBE with Short Ciphertexts and Private Keys. Computing Research Repository, `http://arxiv.org/abs/0807.1775`
29. Naccache, D., Shamir, A., Stern, J.-P.: How to Copyright a Function. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 188–196. Springer, Heidelberg (1999)
30. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
31. Pedersen, T.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
32. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
33. Sakai, R., Kasahara, M.: ID-based Cryptosystems with Pairing on Elliptic Curve. In: SCIS 2003 (2003), `http://eprint.iacr.org/2003/054`
34. Sarkar, P., Chatterjee, S.: Construction of a Hybrid HIBE Protocol Secure Against Adaptive Attacks. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 51–67. Springer, Heidelberg (2007)
35. Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
36. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
37. Shoup, V., Gennaro, R.: A Note on An Encryption Scheme of Kurosawa and Desmedt. Cryptology ePrint Archive: Report 2004/194 (2004)
38. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A A Variant with White-Box FindKey-CCA Security

To achieve IND-sID-CCA2 security, we can hybridize the scheme using an authenticated symmetric encryption scheme (as defined in appendix B) as previously considered in [34,26] in the context of identity-based encryption. The obtained variant is reminiscent of a version of Gentry's IBE described in [26].

Setup: is the same as in section 3 except that the PKG now chooses two elements $Y_A, Y_B \xleftarrow{\$} \mathbb{G}$ instead of a single one Y. An authenticated symmetric encryption scheme (E, D) of keylength $\ell \in \mathbb{N}$, a secure key derivation function $KDF : \mathbb{G}_T \rightarrow \{0,1\}^{\ell}$ and a target collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ are also needed. The master key is set as $\mathsf{msk} := x$ and the global public key is $\mathsf{mpk} := (X = g^x, h, Y_A, Y_B, Z, H, KDF, (\mathsf{E}, \mathsf{D}))$.

Keygen$^{(\mathsf{PKG},\mathsf{U})}$**:** to obtain a private key for his identity ID, a user U interacts with the PKG as follows.

1. U sends $R = h^{t_0} \cdot X^{\theta}$ to the PKG and proves his knowledge of the underlying pair $(t_0, \theta) \xleftarrow{\$} (\mathbb{Z}_p^*)^2$ in a witness indistinguishable fashion.
2. The PKG outputs $\perp$ if the proof is incorrect. Otherwise, it picks random values $r'_A, t_{A,1}, r'_B, t_B \xleftarrow{\$} \mathbb{Z}_p^*$ and returns

$$d'_{\mathsf{ID},A} = (d'_{A,1}, d'_{A,2}, d'_{A,3}) = \Big((Y \cdot R \cdot h^{t_{A,1}})^{1/x} \cdot (g^{\mathsf{ID}} \cdot Z)^{r'_A},\ X^{r'_A},\ t_{A,1}\Big)$$
$$d'_{\mathsf{ID},B} = (d'_{B,1}, d'_{B,2}, d'_{B,3}) = \Big((Y_B \cdot h^{t_B})^{1/x} \cdot (g^{\mathsf{ID}} \cdot Z)^{r'_B},\ X^{r'_B},\ t_B\Big)$$

3. U computes $d_{\mathsf{ID},A} = (d'_{A,1}/g^{\theta} \cdot (g^{\mathsf{ID}} \cdot Z)^{r''_A},\ d'_{A,2} \cdot X^{r''_A},\ d'_{A,3} + t_0)$ as well as $d_{\mathsf{ID},B} = (d'_{B,1} \cdot (g^{\mathsf{ID}} \cdot Z)^{r''_B},\ d'_{B,2} \cdot X^{r''_B},\ d_{B,3})$, for randomly chosen $r''_A, r''_B \xleftarrow{\$} \mathbb{Z}_p^*$ so that

$$d_{\mathsf{ID},A} = (d_{A,1}, d_{A,2}, d_{A,3}) = \Big((Y_A \cdot h^{t_A})^{1/x} \cdot (g^{\mathsf{ID}} \cdot Z)^{r_A},\ X^{r_A},\ t_A\Big) \quad (8)$$
$$d_{\mathsf{ID},B} = (d_{B,1}, d_{B,2}, d_{B,3}) = \Big((Y_B \cdot h^{t_B})^{1/x} \cdot (g^{\mathsf{ID}} \cdot Z)^{r_B},\ X^{r_B},\ t_B\Big) \quad (9)$$

where $t_A = t_0 + t_{A,1}$, $r_A = r'_A + r''_A$ and $r_B = r'_B + r''_B$. He checks whether $d_{\mathsf{ID},A}$ and $d_{\mathsf{ID},B}$ respectively satisfy

$$e(d_{A,1}, X) = e(Y_A, g) \cdot e(h, g)^{d_{A,3}} \cdot e(g^{\mathsf{ID}} \cdot Z, d_{A,2}) \quad (10)$$
$$e(d_{B,1}, X) = e(Y_B, g) \cdot e(h, g)^{d_{B,3}} \cdot e(g^{\mathsf{ID}} \cdot Z, d_{B,2}). \quad (11)$$

If so, he sets his private key as $(d_{\mathsf{ID},A}, d_{\mathsf{ID},B})$ and the latter belongs to the family of decryption key identified by $n_F = d_{A,3} = t_A$.

Encrypt: to encrypt m given mpk and ID, choose $s \xleftarrow{\$} \mathbb{Z}_p^*$ and compute

$$C = \big(C_1, C_2, C_3, C_4\big) = \Big(X^s,\ (g^{\mathsf{ID}} \cdot Z)^s,\ e(g,h)^s,\ \mathsf{E}_K(m)\Big)$$

where $K = KDF(e(g, Y_A)^s \cdot e(g, Y_B)^{\kappa s})$ and $\kappa = H(C_1, C_2, C_3)$.

Decrypt: given $C = \big(C_1, C_2, C_3, C_4\big)$ and $d_{\mathsf{ID}} = (d_{\mathsf{ID},A}, d_{\mathsf{ID},B})$, compute the plaintext $m = \mathsf{D}_K(C_4)$ (which may just be $\perp$ if C_4 is not a valid authenticated encryption) using the key

$$K = KDF\Big(\frac{e(C_1, d_{A,1} \cdot d_{B,1}^{\kappa})}{e(C_2, d_{A,2} \cdot d_{B,2}^{\kappa}) \cdot C_3^{d_{A,3}+\kappa d_{B,3}}}\Big) \tag{12}$$

with $\kappa = H(C_1, C_2, C_3)$.

Trace: given an alleged private key $(d_{\mathsf{ID},A}, d_{\mathsf{ID},B})$, with $d_{\mathsf{ID},A} = (d_{A,1}, d_{A,2}, d_{A,3})$, for an identity ID, check the validity of d_{ID} w.r.t. ID using relations (10)-(11). If valid, the key is declared as a member of the family $n_F = d_{3,A} = t_A$.

To prove the IND-sID-CCA security, we can apply the technique of [26], which in turn borrows ideas from [27,37,24].

In the chosen-ciphertext scenario, the white-box FindKey security is no longer unconditional but relies on the ciphertext integrity of the symmetric encryption scheme.

Theorem 6. *The scheme is IND-sID-CCA secure in the standard model if the modified DBDH assumption holds, if the symmetric scheme is a secure authenticated encryption scheme, if H is target collision-resistant and if the key derivation function is secure. More precisely, we have*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathrm{CCA}}(\lambda, \ell) \leq \frac{q_d + 2q_d^2}{p} + \mathbf{Adv}^{\mathsf{TCR}}(\lambda) + \mathbf{Adv}^{\mathsf{mDBDH}}(\lambda) + 3q_d \cdot \mathbf{Adv}^{\mathsf{CT\text{-}INT}}(\ell) \\ + (2q_d + 1) \cdot \mathbf{Adv}^{\mathsf{KDF}}(\lambda, \ell) + \mathbf{Adv}^{\mathsf{IND\text{-}SYM}}(\ell)\end{aligned}$$

where q_d denotes the number of decryption queries allowed to the adversary $\mathcal{A}$ and the advantage functions against (E, D) *are defined in appendix B.*

Proof. Given in the full version of the paper [28]. □

Theorem 7. *The scheme is* FindKey-CCA *secure assuming the security of the key derivation function and the (weak) ciphertext integrity of the symmetric encryption scheme. The advantage of an adversary $\mathcal{A}$ making at most q_d decryption queries is bounded by*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathrm{FindKey\text{-}CCA}}(\lambda, \ell) \leq 2 \cdot q_d \cdot \mathbf{Adv}^{\mathsf{CT\text{-}INT}}(\ell) \\ + 2 \cdot q_d \cdot \mathbf{Adv}^{\mathsf{KDF}}(\lambda, \ell) + \frac{2q_d^2 + q_d + 1}{p}.\end{aligned}$$

Proof. Given in appendix C. □

B Authenticated Symmetric Encryption

A symmetric encryption scheme is specified by a pair (E, D), where E is the encryption algorithm and D is the decryption procedure, and a key space $\mathcal{K}(\ell)$

where $\ell \in \mathbb{N}$ is a security parameter. The security of authenticated symmetric encryption is defined by means of two games that capture the ciphertext indistinguishability and ciphertext (one-time) integrity properties.

Definition 4. *An symmetric encryption scheme is secure in the sense of authenticated encryption if any PPT adversary has negligible advantage in the following games.*

1. **The IND-SYM game.** For any PPT algorithm $\mathcal{A}$, the model considers the following game, where $\ell \in \mathbb{N}$ is a security parameter:

```
$\boxed{\mathbf{Game}_{\mathcal{A}}^{\mathsf{IND\text{-}SYM}}(\ell)}$
$K \xleftarrow{\$} \mathcal{K}(\ell)$
$(m_0, m_1, s) \leftarrow \mathcal{A}(\mathsf{find}, \ell)$
$d^\star \xleftarrow{\$} \{0, 1\}$
$c^\star \leftarrow \mathsf{E}_K(m_{d^\star})$
$d \leftarrow \mathcal{A}(\mathsf{guess}, s, c^\star)$
return 1 if $d = d^\star$ and 0 otherwise.
```

$\mathcal{A}$'s advantage is $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SYM}}(\ell) = |\Pr[\mathbf{Game}_{\mathcal{A}}^{\mathsf{IND\text{-}SYM}} = 1] - 1/2|$.

2. **The CT-INT game.** Let $\mathcal{A}$ be a PPT algorithm. We consider the following game, where $\ell \in \mathbb{N}$ is a security parameter:

```
$\boxed{\mathbf{Game}_{\mathcal{A}}^{\mathsf{CT\text{-}INT}}(\ell)}$
$K \xleftarrow{\$} \mathcal{K}(\ell)$
$(m, s) \leftarrow \mathcal{A}(\mathsf{find}, \ell)$
$c \leftarrow \mathsf{E}_K(m)$
$c' \leftarrow \mathcal{A}(\mathsf{create}, \ell, c)$
return 1 if $c' \neq c$ and $\mathsf{D}_K(c') \neq \perp$
         0 otherwise.
```

$\mathcal{A}$'s advantage is now defined as $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{CT\text{-}INT}}(\ell) = \Pr[\mathbf{Game}_{\mathcal{A}}^{\mathsf{CT\text{-}INT}} = 1]$.

The notion of weak ciphertext integrity is defined in the same way but the adversary is not allowed to see an encryption c under the challenge key K.

C Proof of Theorem 7

The proof proceeds again with a sequence of games, in all of which S_i denotes the event that the adversary $\mathcal{A}$ wins.

Game$_0$: is the FindKey-CCA experiment. The dishonest PKG $\mathcal{A}$ generates the master public key, chooses an identity ID that she wishes to be challenged upon. She interacts with the challenger in a key generation protocol, upon completion of which the challenger $\mathcal{B}$ obtains a decryption key consisting of two triples $d^{(1)}_{\mathsf{ID},A} = ({d_{A,1}}^{(1)}, {d_{A,2}}^{(1)}, {d_{A,3}}^{(1)})$, $d^{(1)}_{\mathsf{ID},B} = ({d_{B,1}}^{(1)}, {d_{B,2}}^{(1)}, {d_{B,3}}^{(1)})$ that should pass the key sanity check (otherwise, $\mathcal{B}$ aborts). At this stage, $\mathcal{A}$ knows $t^{(1)}_B = d^{(1)}_{B,3}$ but has no information on $d^{(1)}_{A,3} = t^{(1)}_A$ or on the values $r_A = \log_X(d^{(1)}_{A,2})$ and $r_B = \log_X(d^{(1)}_{B,2})$ (by the construction of the key generation protocol). In the next phase, $\mathcal{A}$ starts making a number of decryption queries that the challenger handles using $(d^{(1)}_{\mathsf{ID},A}, d^{(1)}_{\mathsf{ID},B})$. Namely, when queried on a ciphertext $C = (C_1, C_2, C_3, C_4)$, $\mathcal{B}$ calculates

$$\psi = \frac{e\big(C_1, d^{(1)}_{A,1} \cdot {d^{(1)}_{B,1}}^{\kappa}\big)}{e\big(C_2, d^{(1)}_{A,2} \cdot {d^{(1)}_{B,2}}^{\kappa}\big) \cdot C_3^{d^{(1)}_{A,3} + \kappa {d_{B,3}}^{(1)}}},$$

where $\kappa = H(C_1, C_2, C_3)$, $K = KDF(\psi)$ and $m = \mathsf{D}_K(C_4)$ which is returned to $\mathcal{A}$ (and may be $\perp$ if C is declared invalid).

At the end of the game, $\mathcal{A}$ outputs a key $(d^{(2)}_{\mathsf{ID},A}, d^{(2)}_{\mathsf{ID},B})$ and wins if $d^{(2)}_{\mathsf{ID},A}$ parses into $({d_{A,1}}^{(2)}, {d_{A,2}}^{(2)}, {d_{A,3}}^{(2)})$ such that ${d_{A,3}}^{(1)} = t^{(1)}_A = t^{(2)}_A = {d_{A,3}}^{(2)}$.

We note that decryption queries on well-formed ciphertexts do not reveal any information to $\mathcal{A}$ (since all well-formed keys yield the same result). We will show that, provided all ill-formed ciphertexts are rejected by $\mathcal{B}$, $\mathcal{A}$ still has negligible information on $t^{(1)}_A$ in the end of the game. For convenience, we distinguish two types of invalid ciphertexts: type I ciphertexts (C_1, C_2, C_3, C_4) are such that $\log_X(C_1) \neq \log_{F(\mathsf{ID})}(C_2)$ (and can be told apart from valid ones by checking if $e(C_1, F(\mathsf{ID})) \neq e(X, C_2)$), where $F(\mathsf{ID}) = g^{\mathsf{ID}} \cdot Z$, whereas type II ciphertexts are those for which $\log_X(C_1) = \log_{F(\mathsf{ID})}(C_2) \neq \log_{e(g,h)}(C_3)$.

Game$_1$: is as Game$_0$ but $\mathcal{B}$ rejects all type I invalid ciphertexts (that are publicly recognizable). Such a malformed ciphertext comprises elements $C_1 = X^{s_1}$, $C_2 = F(\mathsf{ID})^{s_1 - s_1'}$ and $C_3 = e(g,h)^{s_1 - s_1''}$ where $s_1' > 0$ and $s_1'' \geq 0$. Hence, the symmetric key K that $\mathcal{B}$ calculates is derived from

$$\psi = e(g, Y_A^{s_1} \cdot Y_B^{\kappa s_1}) \cdot e(F(\mathsf{ID}), X)^{s_1'(r_A + \kappa r_B)} \cdot e(g,h)^{s_1''(t^{(1)}_A + \kappa t^{(1)}_B)} \qquad (13)$$

where $\kappa = H(C_1, C_2, C_3)$. Upon termination of the key generation protocol, $\mathcal{A}$ has no information on r_A, r_B (as $\mathcal{B}$ re-randomizes its key). Even if κ was the same in all decryption queries (which may happen if these queries all involve identical (C_1, C_2, C_3)), the second term of the product (13) remains almost uniformly random to $\mathcal{A}$ at each new query. Indeed, for each failed one, $\mathcal{A}$ learns at most one value that is not $r_A + \kappa r_B$. After i attempts, $p - i$ candidates are left and the distance between the uniform distribution on $\mathbb{G}_T$ and that of $e(F(\mathsf{ID}), X)^{s_1'(r_A + \kappa r_B)}$ becomes at most $i/p \leq q_d/p$. Then, the only way for $\mathcal{A}$ to cause the new rejection rule to apply is to forge a symmetric authenticated encryption for an essentially random key K. A standard argument shows that, throughout all queries, the probability of $\mathcal{B}$ not rejecting a type I ciphertext is smaller than $q_d \cdot (\mathbf{Adv}^{\mathsf{CT\text{-}INT}}(\ell) + \mathbf{Adv}^{\mathsf{KDF}}(\lambda, \ell) + q_d/p)$. It easily comes that $|\Pr[S_1] - \Pr[S_0]| \leq q_d \cdot (\mathbf{Adv}^{\mathsf{CT\text{-}INT}}(\lambda) + \mathbf{Adv}^{\mathsf{KDF}}(\lambda, \ell) + q_d/p)$.

We now consider type II invalid queries. While $\mathcal{A}$ knows $t^{(1)}_B$, she has initially no information on $t^{(1)}_A$ and the last term of the product (13) is unpredictable to her at the first type II query. Each such rejected query allows $\mathcal{A}$ to rule out at most one candidate as for the value $t_A^{(1)}$. After $i \leq q_d$ unsuccessful type II queries, she is left with at least $p - i$ candidates at the next type II query, where the distance between the uniform distribution on $\mathbb{G}_T$ and that of ψ (calculated as per (13)) becomes smaller than $i/p \leq q_d/p$. Again, one can show that, throughout all queries, the probability of $\mathcal{B}$ not rejecting a type II ciphertext is at most $q_d \cdot (\mathbf{Adv}^{\mathsf{CT\text{-}INT}}(\ell) + \mathbf{Adv}^{\mathsf{KDF}}(\lambda, \ell) + q_d/p)$. Let us call `type-2` the latter event. If all invalid ciphertexts are rejected, $\mathcal{A}$'s probability of success is given by $\Pr[S_1|\neg\texttt{type-2}] \leq 1/(p - q_d) \leq (q_d + 1)/p$. Since

$$\begin{aligned}\Pr[S_1] &= \Pr[S_1 \wedge \texttt{type-2}] + \Pr[S_1 \wedge \neg\texttt{type-2}] \\ &\leq \Pr[\texttt{type-2}] + \Pr[S_1|\neg\texttt{type-2}]\Pr[\neg\texttt{type-2}] \\ &\leq \Pr[\texttt{type-2}] + \Pr[S_1|\neg\texttt{type-2}] \\ &\leq q_d \cdot \big(\mathbf{Adv}^{\mathsf{CT\text{-}INT}}(\ell) + \mathbf{Adv}^{\mathsf{KDF}}(\lambda, \ell) + \frac{q_d}{p}\big) + \frac{q_d + 1}{p}\end{aligned}$$

and $|\Pr[S_0] - \Pr[S_1]| \leq q_d \cdot (\mathbf{Adv}^{\mathsf{CT\text{-}INT}}(\lambda) + \mathbf{Adv}^{\mathsf{KDF}}(\lambda, \ell) + q_d/p)$, the claimed upper bound follows. □

Removing Escrow from Identity-Based Encryption

New Security Notions and Key Management Techniques

Sherman S.M. Chow[*]

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu

Abstract. Key escrow is inherent in identity-based encryption (IBE). A curious key generation center (KGC) can simply generate the user's private key to decrypt a ciphertext. However, can a KGC still decrypt if it *does not* know the intended recipient of the ciphertext? We answer by formalizing KGC anonymous ciphertext indistinguishability ($\mathcal{ACI}-\mathcal{KGC}$).

We find that all existing pairing-based IBE schemes without random oracles, whether receipt-anonymous or not, do not achieve KGC one-wayness, a weaker notion of $\mathcal{ACI}-\mathcal{KGC}$. In view of this, we first show how to equip an IBE scheme by Gentry with $\mathcal{ACI}-\mathcal{KGC}$. Second, we propose a new system architecture with an anonymous private key generation protocol such that the KGC can issue a private key to an authenticated user without knowing the list of users identities. This also better matches the practice that authentication should be done with the local registration authorities instead of the KGC. Our proposal can be viewed as mitigating the key escrow problem in a different dimension than distributed KGCs approach.

1 Introduction

The feature that differentiates identity-based encryption (IBE) scheme from other public key encryption schemes lies in the way a public and private key pair is set up – every arbitrary string is a valid public key. There is a trusted authority, called the key generation center (KGC), responsible for the generation of private keys after user authentications. Private key generation applies the KGC's master secret key to the users' identities. The major benefit of this approach is to largely reduce the need for processing and storage of public key certificates under traditional public key infrastructure (PKI).

Nevertheless, the advantages come with a major drawback which is known as the *escrow problem*. The KGC could decrypt any message addressed to a user

[*] The author would like to thank Yevgeniy Dodis for the inspiration of this research and many fruitful discussions, Kenneth Paterson for his invaluable assistance and suggestions, and Melissa Chase for her helpful comments.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 256–276, 2009.

by generating that user's private key. To escape from the eye of the KGC, two users may execute an interactive key agreement protocol (e.g. [24]) to establish a session key known only to themselves, or the recipient can setup another key pair and employ certificateless encryption [2,23,25,26], which is a two-factor encryption method involving both IBE and public key encryption. However, one of the main benefits of IBE is lost – it is no longer true that a ciphertext can be prepared without any action by the recipient.

Can anonymity help confidentiality? Current study of IBE only considers anonymity against malicious users' attack, except a recent and independent work [34] which considers the application of KGC-anonymous IBE in password-authenticated key exchange but without any application in the context of IBE itself. We try to use anonymity against a malicious KGC to fight against the escrow problem. If the KGC *does not* know the intended recipient of the ciphertext, is it still possible for it to decrypt on behalf of the user? We answer this question by introducing the notions of KGC one-wayness ($\mathcal{OW}-\mathcal{KGC}$) and KGC anonymous ciphertext indistinguishability ($\mathcal{ACI}-\mathcal{KGC}$).

We find that (to the best of our knowledge) no existing pairing-based IBE schemes without random oracles can achieve the weakest notion of confidentiality $\mathcal{OW}-\mathcal{KGC}$, no matter whether it is user-anonymous. In view of this, we show to equip Gentry's IBE scheme [28] with $\mathcal{ACI}-\mathcal{KGC}$ in the standard model.

How can KGC *not* know the users' identities? Our notion of $\mathcal{ACI}-\mathcal{KGC}$ minimizes the damage of master secret key exposure, providing protection against adversaries who hold the master secret key but not the list of user identities. However, it is natural for the KGC to have this list. By generating all possible user private keys, the KGC can decrypt all ciphertexts. In view of this, we propose a new system architecture to prevent the KGC from knowing it.

We acknowledge that the KGC can always try to derive all possible user private keys according to a certain "dictionary". It seems that there is not much we can do to protect ourselves against a strong adversary like the KGC in this situation. Nevertheless our notion is useful when there is some min-entropy from the identities (e.g. biometric identity [43]). On the other hand, nothing can be gained if one always stores the identity with the ciphertext.[1]

New Key Management Techniques. We separate the tasks of authentication and key issuing, hence our system architecture employs two parties, namely, an identity-certifying authority (or ICA in short) and a KGC. This setting is different from a typical ID-based cryptosystem, but actually better matches the practice that authentication should be done with the local registration authorities, especially when the KGC is not globally available to authenticate users.

The master secret is still solely owned by the KGC. In particular, it is not spilt across two authorities, in contrast with the distributed KGCs approach. The ICA is responsible for issuing some kind of certificates, but it does not need to store any of them, and only the KGC is required to verify the certificate.

[1] Don't write your address on a tag with your key to guide the thief who picked it up.

After obtaining the private key, users do not require any further interaction with these authorities for decryption. Last but not least, the certificate is not used anywhere else in the system, i.e. the encryption itself is still purely ID-based.

Under this model, we show that one can put anonymous ciphertext indistinguishability in practice. We give a design of the anonymous private key issuing protocol, and present a concrete protocol construction for Gentry-IBE.

1.1 Review of Identity-Based Encryption

The concept of IBE was formulated by Shamir in 1984 [47]. Satisfactory proposals for IBE did not exist until nearly two decades afterward, when Boneh and Franklin [12] and Sakai *et al.* [45] presented two IBE solutions based on pairing and full-domain hash to elliptic curve points (referred to as FDH-IBE).

Reduction Improvement. Since Boneh-Franklin's work (BF-IBE), there has been a flurry of variants. For improving the security reduction in the random oracle model, Attrapadung *et al.* [3] worked out an FDH-IBE having two public keys for an identity, an idea which was used to improve the security reduction of FDH signature. Galindo [27] gave a variant of BF-IBE using another transformation technique (different from the one in [12]) to get adaptive chosen-ciphertext security (CCA2). Modifying BF-IBE, Libert and Quisquater [39] gave an IBE without redundancy [42]. All these schemes share a similar $\mathcal{ACI} - \mathcal{KGC}$ analysis.

Multi-Recipient and Hierarchical ID-Based Encryption (HIBE). In HIBE, the workload of private key generation of a single root KGC is delegated to many lower-level KGCs. Gentry and Silverberg proposed the first full-blown (compared with [33]) HIBE (GS-HIBE) [29]. For encrypting to multiple recipients more efficiently than in the straightforward approach, multi-recipient IBE was proposed by Baek *et al.* (BSS-MIBE) [4]. An extension of [4] with shorter ciphertext was proposed in [39]. These schemes bear similarities to GS-HIBE.

Exponent-Inversion IBE. Sakai and Kasahara [44] proposed another IBE (SK-IBE) with a private key derivation algorithm based on exponent-inversion, which is different from FDH-IBE. The CCA2-security of SK-IBE is proven in another work [22], albeit in the random oracle model.

The first exponent-inversion IBE in the standard model was proposed by Boneh and Boyen [8] (hereinafter referred to as BB-EIIBE), which offers selective-ID security. Using the chameleon hashing technique due to Waters [49], an extension of [8] with adaptive-ID security was proposed in [36]. Since only the way of hashing the identity is changed, they share the same $\mathcal{ACI} - \mathcal{KGC}$ analysis.

Standard Model (Commutative-Blinding). Boneh and Boyen proposed selective-ID IBE and HIBE schemes in [8] (hereinafter referred to as BB-(H)IBE). Shortly afterward, they gave an adaptive-ID version [9]. Waters simplified [9] in [49], and gave a fuzzy version with Sahai in [43]. Extending from [49], Kiltz and Galindo [37] gave a CCA2 ID-based key encapsulation without using any transformation, and Kiltz and Vahlis [38] gave an efficient CCA2 ID-based key encapsulation scheme using authenticated symmetric encryption. Extending from [43], Boldyreva *et al.* [7] gave an IBE with efficient revocation.

Regarding HIBE, [9] and [49] suggested HIBE extensions similar to the approach in [8]. An HIBE with constant-size ciphertext was proposed in [10], which was later made adaptive-ID secure in [20]. Generalizations of the selective-ID model for HIBE, with two HIBE constructions, were proposed in [17]. HIBE with short public parameters was proposed in [18]. A multi-recipient IBE and a parallel key-insulated IBE in standard model were proposed in [19] and [50].

Despite their apparent versatility (e.g. different ways of generating public keys from identities), all these schemes use a similar implicit key encapsulation method. As a result, they share a similar $\mathcal{ACI}-\mathcal{KGC}$ analysis. Finally, [21,41] studied the tradeoff between key size and security reduction for [49].

Standard Model (with User Anonymity). Boyen and Waters [15] proposed an anonymous IBE scheme (BW-IBE) and the first anonymous HIBE (AHIBE). It has been suggested in [15] that AHIBE can obtain adaptive security by the hashing technique of Waters [49]. Similar to the extension of [8] in [36], it does not affect the $\mathcal{ACI}-\mathcal{KGC}$ analysis. Recently, [10] has been made anonymous in [46]. Although these schemes are anonymous, they can be shown to be not $\mathcal{OW}-\mathcal{KGC}$-secure in a similar way to BB-(H)IBE.

Gentry's scheme also provides anonymity in the standard model [28]. It has been extended by Kiltz and Vahlis using authenticated symmetric encryption for better efficiency (KV-IBE) [38], and by Libert and Vergnaud for more efficient weak black-box accountable IBE (LV-IBE) [40]. We will show that Gentry-IBE can be made $\mathcal{ACI}-\mathcal{KGC}$ secure, but interestingly, its extensions [38,40] are not. Actually, LV-IBE mixes commutative-blinding and exponent-inversion – its $\mathcal{OW}-\mathcal{KGC}$ can be broken similar to breaking BB-EIIBE or BB-(H)IBE.

Generalizations of IBE. Recently, there have been many generalizations of IBE, such as hidden-vector encryption [14], predicate encryption [35] and spatial encryption [13]. However, it can be shown that they are not $\mathcal{OW}-\mathcal{KGC}$-secure.

1.2 Attempts in Reducing Trust in the KGC

Accountable IBE. In accountable IBE [30] (AIBE), the trust in the KGC is reduced in another dimension, such that the KGC is discouraged from leaking or selling any user secret key. Consider an IBE scheme with an exponential number of user secret keys for any given identity, such that deriving any other secret key from any one of them (without the knowledge of the master secret key) is intractable; if the key issuing protocol ensures that the user can obtain a user private key without letting the KGC know which one it is, we can conclude that the KGC must be the one who leaks the user private key if a user can show the existence of two private keys for the same identity. Goyal [30] showed that Gentry-IBE satisfies the aforementioned properties, and proposed the corresponding key issuing protocol, which also works with our modified Gentry-IBE. Another AIBE scheme that is based on Waters IBE [49] and Sahai-Waters fuzzy

IBE [43] was also proposed in [30]. Goyal *et al.* [31] later proposed a black-box accountable IBE (BBAIBE). However, these schemes are not $\mathcal{OW}-\mathcal{KGC}$-secure.

KGC-Anonymous ID-based KEM. Independent of our work, anonymity against an honest but curious KGC attack was considered by Izabachène and Pointcheval [34]. Their notion of key anonymity with respective to authority (KwrtA), given in the context of identity-based KEM (IB-KEM), requires the adversary to guess between the two possibilities of recipient identity, with the master secret key and the challenge ciphertext, but *without* the ephemeral session key. In the context of IBE, the ciphertext always contain a component which encrypts the message by this session key. Taking it away means that the challenge is "incomplete" since partial knowledge of it can be seen in the ciphertext produced by IBE. Hence, the real-world impact on IBE given by their security notion may be unclear. Nevertheless, they showed that an IB-KEM with this KwrtA-anonymity and ID-based non-malleability (another new notion in [34]) is a useful tool for constructing password-authenticated key exchange protocols. Relationships between our notion and theirs will be given in §5.4.

Distributed KGCs. A standard method to avoid the inherent key escrow is to split the master secret key to multiple KGCs. The user private key generation is then done in a threshold manner, where each KGC uses a share of the master secret key to generate a private key component for a user. In our approach, the master secret key is not distributed. It is always possible to have this key distribution on top of our idea if an extra layer of protection is desirable.

2 Definitions

2.1 Notations and Complexity Assumptions

We use $x \in_R S$ to denote the operation of picking an element x at random and uniformly from a finite set S. For a probabilistic algorithm $\mathcal{A}$, $x \stackrel{\$}{\leftarrow} \mathcal{A}$ assigns the output of $\mathcal{A}$ to the variable x. If x is a string, $|x|$ denotes its length. If $\lambda \in \mathbb{N}$, 1^λ denotes a string of λ ones. A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible ($\mathsf{negl}(k)$) if for every constant $c \geq 0$ there exists k_c such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

Definition 1 (Bilinear Map). *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of prime order p. A bilinear map $e(\cdot,\cdot) : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies:*

1. *Bilinearity: For all $u, v \in \mathbb{G}$, $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.*
2. *Non-degeneracy: $e(g, g) \neq 1$ where g is a generator of $\mathbb{G}$.*

Definition 2. *(Decisional) Bilinear Diffie-Hellman Problem (DBDHP): Given $g, g^a, g^b, g^c \in \mathbb{G}$, and $\hat{t} \in \mathbb{G}_T$, output* 'yes' *if $\hat{t} = e(g, g)^{abc}$ and* 'no' *otherwise.*

We introduce two problems whose names are inspired by the decisional linear problem [11]. An oracle for solving the first one makes solving DBDHP easily.

Definition 3. *Decisional Bilinear Problem (DBP): Given two $\mathbb{G}$ elements g and g^a, two $\mathbb{G}_T$ elements $e(g,g)^b$ and $\hat{t}$, output* 'yes' *if $\hat{t} = e(g,g)^{ab}$ and* 'no' *otherwise. We name $(g, g^a, e(g,g)^b, e(g,g)^{ab})$ as a decisional bilinear tuple.*

Definition 4. *Modified Decisional Bilinear Problem (MDBP): Given $g, g^a, g^{b^{-1}} \in \mathbb{G}$, and $e(g,g)^b, \hat{t} \in \mathbb{G}_T$, output* 'yes' *if $\hat{t} = e(g,g)^{ab}$ and* 'no' *otherwise.*

Lemma 1. *DBDH assumption implies Decisional Bilinear assumption.*

Proof. Given $(g, g^a, g^b, g^c, \hat{t})$, computes $e(g,g)^{b'} = e(g^b, g^c)$ where $b' = bc$, feeds $(g, g^a, e(g,g)^{b'}, \hat{t})$ to the DBP oracle and outputs its answer. □

Definition 5. *(Decisional) q-Bilinear Diffie-Hellman Exponent Problem (q-BDHEP): Given $(q+2)$ $\mathbb{G}$ elements $(g', g, g^{\alpha}, \cdots, g^{\alpha^q})$, and one $\mathbb{G}_T$ element $\hat{t}$, output* 'yes' *if $\hat{t} = e(g^{\alpha^{q+1}}, g')$ and* 'no' *otherwise.*

A stronger version of q-BDHEP is assumed difficult for the security of Gentry-IBE. We remark that the hard problem considered in [28] is augmented with $g'^{\alpha^{q+2}}$ and q equals to the number of users compromised by the adversary.

Lemma 2. *Decisional 2-Bilinear Diffie-Hellman Exponent assumption implies Modified Decisional Bilinear assumption.*

Proof. Given $(g', g, g^{\alpha}, g^{\alpha^2}, \hat{t})$, set $\theta_1 = g^{\alpha}$, $\theta_2 = g'$, $\theta_3 = g$, $\hat{\theta} = e(g^{\alpha}, g^{\alpha^2})$ and feed $(\theta_1, \theta_2, \theta_3, \hat{\theta}, \hat{t})$ to the MDBP oracle. The input is valid since $\theta_3 = (\theta_1)^{\alpha^{-1}}$ and $\hat{\theta} = e(\theta_1, \theta_1)^{\alpha}$. Let $\theta_2 = \theta_1^{\gamma}$ where $\gamma \in \mathbb{Z}_p$, the MDBP oracle outputs 'yes' if and only if $\hat{t} = e(\theta_1, \theta_1)^{\gamma\alpha}$, since $e(\theta_1, \theta_1)^{\gamma\alpha} = e(g^{\alpha}, g^{\alpha})^{\gamma\alpha} = e(g^{\alpha^3}, g')$. □

2.2 Identity Based Encryption

Under the standard definition, an IBE scheme consists of four algorithms:

1. via $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}(1^{\lambda})$ the randomized key generation algorithm outputs the system parameters mpk and the master secret key msk;
2. via $usk[\mathsf{ID}] \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathsf{ID})$ the KGC outputs[2] a secret key for user ID;
3. via $\mathfrak{C} \xleftarrow{\$} \mathsf{Enc}(mpk, \mathsf{ID}, m)$ anyone can encrypt a message m to user ID in $\mathfrak{C}$;
4. via $m \leftarrow \mathsf{Dec}(mpk, usk[\mathsf{ID}], \mathfrak{C})$ user ID uses secret key usk to get m from $\mathfrak{C}$.

Consistency requires that for all $\lambda \in \mathbb{N}$, all identities ID, all messages $m \in \mathsf{MsgSp}$ (defined by mpk) and all $\mathfrak{C} \xleftarrow{\$} \mathsf{Enc}(mpk, \mathsf{ID}, m)$, $\Pr[\mathsf{Dec}(\mathsf{KeyDer}(msk, \mathsf{ID}), \mathfrak{C}) = m] = 1$, where the probability is taken over the coins of all the above algorithms.

In our definition, we separate the master key generation from the Setup.

Definition 6. *An IBE scheme consists of the following five PPT algorithms:*

[2] This algorithm is deterministic for most schemes stemmed from FDH-IBE.

1. *via* $param \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ *the setup algorithm outputs the system parameters* $param$ *for security parameter* $\lambda \in \mathbb{N}$, *with message space* $\mathsf{MsgSp}(\lambda)$ *included.*
2. *via* $(mpk, msk) \xleftarrow{\$} \mathsf{MKeyGen}(param)$ *the key generation algorithm outputs the master public/secret key* (mpk, msk) *conforming to* $param$*;*
3. KeyDer, Enc *and* Dec *are defined as in the standard definition.*

We can view Setup as a trusted initializer for choosing the system parameters (for examples, the choice of elliptic cure) which are implicitly included in the input of KeyDer, Enc and Dec. The KGC generates a master public/private key pair only via $\mathsf{MKeyGen}$. We assume it is efficient to check if a message m is in $\mathsf{MsgSp}(\lambda)$ or if mpk comes from a group that matches with what is specified in $param$. We denote the latter check by (an abused notation) $mpk \in param$.

3 Anonymity and Indistinguishability against the KGC

3.1 Anonymity against User Attack

User-anonymity is defined by the game below [1]. The adversarial goal is to distinguish the intended recipient of a ciphertext between two chosen identities[3].

```
Experiment $\mathbf{Exp}^{\text{ano-cpa}}_{\mathcal{IBE},\mathcal{A}}(\lambda)$
    $\mathsf{IDset} \leftarrow \emptyset$; $(param) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$; $(mpk, msk) \xleftarrow{\$} \mathsf{MKeyGen}(param)$;
    $(\mathsf{ID}_0, \mathsf{ID}_1, m^*, st) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDerO}(\cdot)}(\text{‘}\mathsf{find}\text{’}, param, mpk)$;
    If $m^* \notin \mathsf{MsgSp}(\lambda)$ then return 0;
    $b \xleftarrow{\$} \{0,1\}$; $\mathfrak{C} \xleftarrow{\$} \mathsf{Enc}(mpk, \mathsf{ID}_b, m^*)$; $b' \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDerO}(\cdot)}(\text{‘}\mathsf{guess}\text{’}, \mathfrak{C}, st)$;
    If $b \neq b'$ or $(\{\mathsf{ID}_0, \mathsf{ID}_1\} \cap \mathsf{IDset} \neq \emptyset)$ then return 0 else return 1;
```

where the private key derivation oracle $\textsc{KeyDerO}(\mathsf{ID})$ is defined as:

$$\mathsf{IDset} \leftarrow \mathsf{IDset} \cup \{\mathsf{ID}\}; usk[\mathsf{ID}] \leftarrow \mathsf{KeyDer}(msk, \mathsf{ID});\ \text{return } usk[\mathsf{ID}]$$

and st denotes the state information maintained by the adversary $\mathcal{A}$.

3.2 Anonymous Ciphertext Indistinguishability

We use the term "anonymous ciphertext" to refer a ciphertext that the KGC holds without the knowledge of who is the intended recipient. We do not model the case where the KGC maliciously generates the system parameters (e.g. the choice of elliptic curve), but we provide a new "embedded-identity encryption" oracle, which lets the adversary adaptively get many ciphertexts designated to the same person, without knowing the real identity. The absence of such an oracle gives the adversary no way to see more than one ciphertext for the unknown recipient. For the ease of discussion, we suppose an identity is of n-bit length.

[3] IBE's ciphertext does not mean to reveal the recipient's identity. We omit anonymity revocation oracle which is present in some cryptographic schemes (e.g. [11]).

Definition 7. *An IBE scheme is (t, q_E, ϵ) $\mathcal{ACI} - \mathcal{KGC}$ secure if all t-time adversaries making at most q_E embedded-identity encryption oracle queries have advantage at most ϵ in winning the game below (i.e. the experiment returns 1).*

$$
\begin{array}{l}
\textbf{Experiment } \mathbf{Exp}^{\mathrm{aci-kgc}}_{\mathcal{IBE},\mathcal{A}}(\lambda) \\
(param) \xleftarrow{\$} \mathsf{Setup}(1^\lambda);\ \mathsf{ID}^* \xleftarrow{\$} \{0,1\}^n; \\
(mpk, st) \xleftarrow{\$} \mathcal{A}(\text{`}\mathsf{gen}\text{'}, param);\ \text{If } mpk \notin param \text{ then return } 0; \\
(m_0^*, m_1^*, st) \xleftarrow{\$} \mathcal{A}^{\mathrm{E{\scriptsize NC}O}_{(mpk,\mathsf{ID}^*)}(\cdot)}(\text{`}\mathsf{find}\text{'}, mpk, st); \\
\text{If } \{m_0^*, m_1^*\} \not\subseteq \mathsf{MsgSp}(\lambda) \text{ or } |m_0^*| \neq |m_1^*| \text{ then return } 0; \\
b \xleftarrow{\$} \{0,1\}; \mathfrak{C} \xleftarrow{\$} \mathsf{Enc}(mpk, \mathsf{ID}^*, m_b^*);\ b' \xleftarrow{\$} \mathcal{A}^{\mathrm{E{\scriptsize NC}O}_{(mpk,\mathsf{ID}^*)}(\cdot)}(\text{`}\mathsf{guess}\text{'}, \mathfrak{C}, st); \\
\text{If } b \neq b' \text{ then return 0 else return 1;}
\end{array}
$$

where the embedded-identity oracle $\mathrm{E{\scriptsize NC}O}_{(mpk,\mathsf{ID}^*)}(m)$ returns $\mathsf{Enc}(mpk, \mathsf{ID}^*, m)$ and the advantage of $\mathcal{A}$ is defined as $|\Pr[\mathbf{Exp}^{\mathrm{aci-kgc}}_{\mathcal{IBE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$.

One may define semantic security of the hidden identity in a similar way; but we omitted it to keep our focus on whether the KGC can decrypt the ciphertext.

Embedded-Identity Decryption. The above game just considers chosen-plaintext attack (CPA). One may consider giving the adversary adaptive access to a decryption oracle, or even an "embedded-identity decryption oracle". We consider this stronger notion from both the theory and practice perspectives.

Our security notion is actually quite strong in the sense that the adversary is not required to reveal the master secret key to the challenger. We start our discussion with a weakened definition such that the adversary is instead required to do so. While it is possible that the decryption oracle could help the adversary to deduce information about the challenge ciphertext, this happens when a maliciously formed ciphertext is presented to the decryption oracle. If we are able to put some validity tag in the ciphertext such that the challenger, with the master secret key, can do a sanity check before the actual decryption; only "invalid" will be returned for any malformed ciphertext or those not encrypted for the challenge identity, i.e. CCA2-security against user also helps in here.

If the challenger does not know the master secret, it may sound impossible to simulate the decryption oracle. Nevertheless, our definition assumes trusted parameter generation, which possibly allows us to solve the problem with approaches similar to simulating the strong decryption oracle in certificateless encryption [2,23,25,26], such as a knowledge-extractor with the help of the random oracle, or a non-interactive zero-knowledge proof system setup according to the trusted parameters. Due to the lack of space, we do not delve into details.

In practice, while it makes sense to trick a user into encrypting some predefined messages (as modeled by the embedded-identity encryption oracle); it may not make much sense to consider the case that the KGC gained accesses to an embedded-identity decryption oracle – which possibly means the KGC has identified this user already. Due to these complications, we keep our focus on the CPA notion. Nevertheless, this does not preclude the possibility of achieving $\mathcal{ACI} - \mathcal{KGC}$-security and CCA2-security against user attack simultaneously.

3.3 Comparison of User Anonymity and KGC One-Wayness

A KGC is a powerful adversary. We consider KGC one-wayness ($\mathcal{OW}-\mathcal{KGC}$), a notion strictly weaker than $\mathcal{ACI}-\mathcal{KGC}$, to better reflect the security of IBE against KGC attacks. We also present two separation results.

Definition 8. *An IBE is* $\mathcal{OW}-\mathcal{KGC}$ *secure if* $\Pr[\mathbf{Exp}^{\text{ow}-\text{kgc}}_{\mathcal{IBE},\mathcal{A}}(\lambda)=1] < \mathsf{negl}(\lambda)$.

```
**Experiment** $\mathbf{Exp}^{\text{ow}-\text{kgc}}_{\mathcal{IBE},\mathcal{A}}(\lambda)$
$(param) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, $\mathsf{ID}^* \xleftarrow{\$} \{0,1\}^n$;
$(mpk, st) \xleftarrow{\$} \mathcal{A}(\text{‘}\mathsf{gen}\text{’}, param)$; If $mpk \notin param$ then return 0;
$m^* \xleftarrow{\$} \mathsf{MsgSp}(\lambda)$; $\mathfrak{C} \xleftarrow{\$} \mathsf{Enc}(mpk, \mathsf{ID}^*, m^*)$; $m' \xleftarrow{\$} \mathcal{A}(\text{‘}\mathsf{guess}\text{’}, \mathfrak{C}, st)$;
If $m^* \neq m'$ then return 0 else return 1;
```

Theorem 1. *User anonymity does not imply* $\mathcal{OW}-\mathcal{KGC}$.

Proof. Given any user-anonymous IBE scheme with encryption algorithm Enc, define a new IBE with encryption algorithm $\mathsf{Enc}'(mpk, \mathsf{ID}, m) = (\mathsf{Enc}(mpk, \mathsf{ID}, m), \mathsf{Enc}(mpk, \text{“0”}, \mathsf{ID}))$, where “0” is a dummy identity and the corresponding user secret key is never released by the KGC. If the IBE scheme is semantically secure, the ciphertext produced by Enc' is still user-anonymous. But it is not $\mathcal{OW}-\mathcal{KGC}$ since the KGC can just generate the user secret key for “0”, decrypt the second component of the ciphertext and then decrypt the first component.

Theorem 2. $\mathcal{ACI}-\mathcal{KGC}$ *does not imply user anonymity.*

Proof. Given any $\mathcal{ACI}-\mathcal{KGC}$ with encryption algorithm Enc, define a new IBE with encryption algorithm which appends the first bit of identity to the ciphertext. Any adversary can just choose two identities which differ at the first bit to break the user-anonymity. On the other hand, the notion of $\mathcal{ACI}-\mathcal{KGC}$ depends on the number of random bits in the identity; essentially only one bit of security is lost and $\mathcal{ACI}-\mathcal{KGC}$ is still preserved.

In next section, we will see they are also orthogonal to each other in practice.

4 Analysis

Table 1 gives a concise and unified review of existing IBE schemes in the context of $\mathcal{ACI}-\mathcal{KGC}$ analysis. Seven (H)IBE schemes representing a large class of IBE schemes in the literature are selected. Note that we made many simplifications and omitted many elegant components of the IBE schemes being analyzed. We do not intend to give a complete review of the constructions of all these schemes (it seems we are reducing these IBE schemes to ID-based key encapsulations or even just public key encryption schemes), but we want to keep our focus on how a KGC can decrypt the message using the master secret key. Thus, we only show the essential components in the master public key mpk, the master secret key msk, the ciphertext, and the variable that can be computed (without using any

Table 1. Concise Review of IBE Schemes for $\mathcal{ACI}-\mathcal{KGC}$ Analysis. Elements in $\mathbb{G}, \mathbb{Z}_p, \mathbb{G}_T$: capital letters, small letters, small letters with hat respectively. Generators of $\mathbb{G}$ and $\mathbb{G}_T$: P and $\hat{g} = e(P, P)$ resp. Ephemeral randomness employed in encryption: r, r'. $Q_{\mathsf{ID}} = H_0(\mathsf{ID})$, where $H_0(\cdot) : \{0,1\}^n \to \mathbb{G}$. Hierarchical identity: $(\mathsf{ID}_1, \mathsf{ID}_2, \cdots)$.

Schemes	mpk	msk	Ciphertext	$\mathcal{K}$
FDH-IBE [12,45]	P^s	s	P^r	$e({Q_{\mathsf{ID}}}^r, P^s)$
GS-HIBE [29]	P^s	s	$P^r, Q^r_{\mathsf{ID}_2}, \cdots$	$e({Q_{\mathsf{ID}_1}}^r, P^s)$
BSS-MIBE [4]	P^s, Q	s	P^r	$e(Q, P^s)^r$
BB-EIIBE [8]	$\hat{g}, V = P^s$	s	V^r	$\hat{g}^r$
BB-(H)IBE [8]	$e(P, S)$	S	P^r	$e(P, S)^r$
BW-IBE [15]	$\hat{v} = \hat{g}^{s_1 s_2 s_3}, V_1 = P^{s_1}, V_2 = P^{s_2}$	s_1, s_2, s_3	$V_1^{r-r'}, V_2^{r'}$	$\hat{v}^r$
KV-IBE [38]	$\hat{g}, \hat{v}_1 = \hat{g}^{s_1}, \hat{v}_2 = \hat{g}^{s_2}$	s_1, s_2	$\hat{g}^r, t$	$(\hat{v}_1^t \hat{v}_2)^r$

secret key) from the ciphertext (t in KV-IBE), which are sufficient for the KGC to do the decryption. We use $\mathcal{K}$ to denote the random session key created by the implicit KEM, which is a crucial piece of data to decrypt the ciphertext.

4.1 Schemes That Are Not $\mathcal{OW}-\mathcal{KGC}$-Secure

The session key $\mathcal{K}$ in BSS-MIBE can be computed by $e(Q, P^r)^s$. For BB-EIIBE, $\mathcal{K}$ can be computed by $e(P, V^r)^{1/s}$. For BB-(H)IBE, $e(P^r, S) = \mathcal{K}$. For BW-IBE, it can be computed by $e((V_2^{r'})^{1/s_2}(V_1^{r-r'})^{1/s_1}, P)^{s_1 s_2 s_3} = e(P^{r'} P^{r-r'}, P^{s_1 s_2 s_3}) = \hat{v}^r$. For KV-IBE, $(\hat{g}^r)^{s_1 t + s_2} = \mathcal{K}$. Hence, they are not $\mathcal{OW}-\mathcal{KGC}$-secure. BBAIBE [31] is not exactly covered by the above analysis, however, it can be easily shown that it is not $\mathcal{OW}-\mathcal{KGC}$-secure. Note that all of the above computations use the master secret key as-is, instead of exploiting the knowledge of any discrete logarithm between some group elements in the system parameters.

4.2 Schemes That Are $\mathcal{ACI}-\mathcal{KGC}$-Secure

We consider FDH-IBE [12,45] – when $\mathcal{K} = e({Q_{\mathsf{ID}}}^r, P^s)$ is used to encrypt the message $m \in \mathbb{G}_T$ by $m\mathcal{K}$, this gives a CPA-secure IBE scheme. To prove its $\mathcal{ACI}-\mathcal{KGC}$-security, we assume the parameters for the hash functions are setup by an honest party, which means the random oracles are not controlled by the adversary in the security proof.

Theorem 3. *If DBP is hard, FDH-IBE is $\mathcal{ACI}-\mathcal{KGC}$ secure.*

Due to the lack of space, we give an informal argument to get some intuition on why is it so. Given any pair of messages (m_0^*, m_1^*) and an encryption of one of them, there is always a pair of identities $(\mathsf{ID}_0, \mathsf{ID}_1)$ such that the decryption of the ciphertext using session key $e({Q_{\mathsf{ID}_0}}^r, P^s)$ gives m_0^* and decryption using $e({Q_{\mathsf{ID}_1}}^r, P^s)$ gives m_1^*. If the challenge identity is chosen from a uniform distribution with high entropy, any adversary simply has no clue to distinguish, and hence the scheme is $\mathcal{ACI}-\mathcal{KGC}$-secure. Note that the above argument remains valid even if the adversary can compute r from P^r.

For the CCA2-secure BF-IBE [12], we can prove it is $\mathcal{ACI}-\mathcal{KGC}$ secure by considering the computational bilinear problem (CBP), the computational variant of DBP (i.e., to compute $e(g,g)^{ab}$ instead of distinguishing it from random). The simulation is similar to that in Theorem 3, but $e(g,g)^{ab}$ will be "trapped" by the random oracle if the adversary has non-negligible in winning the game.

Lemma 3. *If CBP is hard, BF-IBE is $\mathcal{ACI}-\mathcal{KGC}$ secure.*

Thus, we can still enjoy the usual CCA2-security against the user (outsider adversary) with the extra $\mathcal{ACI}-\mathcal{KGC}$ protection. A similar argument applies to Gentry-Silverberg HIBE and Yao *et al.*'s HIBE [51]. Extra elements in the challenge ciphertext only contain more information about r and the identities at the lower level, which cannot help the adversary to determine the first-level identity or distinguish the ciphertext. They can also be easily simulated by manipulating the random oracle. This gives an interesting result that even when the ciphertext is not "strictly" user-anonymous, it is still possible to get $\mathcal{ACI}-\mathcal{KGC}$-security.

5 "Escrow-Free" IBE in the Standard Model

BF-IBE is $\mathcal{ACI}-\mathcal{KGC}$-secure but its CCA2-security is only proven in the random oracle model. Below we review Gentry-IBE [28], an IBE with CCA2-security proven in the standard model, under the original four-algorithm IBE framework.

Setup: The KGC selects g, h_1, h_2, h_3 randomly from $\mathbb{G}$, randomly chooses an exponent $\alpha \in_R \mathbb{Z}_p$, sets $g_1 = g^\alpha \in \mathbb{G}$, and chooses a hash function $H : \{0,1\}^n \to \mathbb{Z}_p$ from a family of universal one-way hash functions. The public parameters and the master secret key are given by $mpk = (g, g_1, h_1, h_2, h_3, H)$, $msk = \alpha$.

KeyDer: To generate a private key for identity $\mathsf{ID} \in \mathbb{Z}_p$, the KGC picks $\tau_{\mathsf{ID},i} \in_R \mathbb{Z}_p$ and computes $h_{\mathsf{ID},i} = \left(h_i g^{-\tau_{\mathsf{ID},i}}\right)^{\frac{1}{\alpha-\mathsf{ID}}}$ for $i \in \{1,2,3\}$, outputs $\{\tau_{\mathsf{ID},i}, h_{\mathsf{ID},i}\}_{i\in\{1,2,3\}}$. The KGC must always use the same random value $\tau_{\mathsf{ID},i}$ for ID. This can be accomplished by using a pseudorandom function (PRF) or an internal log [28].

Enc: To encrypt $m \in \mathbb{G}_T$ for identity $\mathsf{ID} \in \mathbb{Z}_p$, the sender picks $r \in_R \mathbb{Z}_p$, computes $\mathfrak{C} = (u, v, w, y) = \Big((g_1 g^{-\mathsf{ID}})^r,\ e(g,g)^r,\ m/e(g,h_1)^r,\ e(g,h_2)^r e(g,h_3)^{r\cdot H(u,v,w)} \Big)$.

Dec: To decrypt the ciphertext $\mathfrak{C}$ with a private key $\{\tau_{\mathsf{ID},i}, h_{\mathsf{ID},i}\}_{i\in\{1,2,3\}}$, first check $\mathfrak{C}$'s validity by testing if $y = e(u, h_{\mathsf{ID},2} {h_{\mathsf{ID},3}}^{\beta}) v^{\tau_{\mathsf{ID},2}+\tau_{\mathsf{ID},3}\beta}$ where $\beta = H(u, v, w)$. In case of inequality, $\perp$ is outputted. Otherwise, return $m = w \cdot e(u, h_{\mathsf{ID},1}) v^{\tau_{\mathsf{ID},1}}$.

5.1 Modification

To get $\mathcal{ACI}-\mathcal{KGC}$, instead of letting the KGC to select g, h_1, h_2, h_3 randomly from $\mathbb{G}$, we require that the discrete logarithm of one with respect to another be unknown to the KGC, or $\mathcal{OW}-\mathcal{KGC}$ can be easily broken. This requirement was not stated in [28]. In practice, this can be achieved by using a common public seed to generate these parameters with a cryptographic hash function. Specifically, we separate the master key generation from the Setup as follows.

Setup: The trusted initializer chooses the group $\mathbb{G}$ according to the security parameter, and selects g, h_1, h_2, h_3 randomly from $\mathbb{G}$. It also chooses a hash function $H : \{0,1\}^n \to \mathbb{Z}_p$ from a family of universal one-way hash functions. The public parameter *param* is given by (g, h_1, h_2, h_3, H).

MKeyGen: The KGC chooses an exponent $\alpha \in_R \mathbb{Z}_p$. It sets $g_1 = g^\alpha \in \mathbb{G}$. The master public/private key pair is given by $(mpk = g_1,\ msk = \alpha)$.

Note that the above change does not affect the original security guarantees of Gentry-IBE against users attack, i.e. CCA2-security and user anonymity.

5.2 Security

With Lemma 2, the below theorem shows that the above IBE is $\mathcal{ACI} - \mathcal{KGC}$ secure without extra number-theoretic assumptions other than what has been assumed in the original proof for indistinguishability against users' attack [28].

Theorem 4. *If MDBP is hard, the above IBE is* $\mathcal{ACI} - \mathcal{KGC}$ *secure.*

Proof. Let $\mathcal{A}$ be an adversary that breaks $\mathcal{ACI} - \mathcal{KGC}$ of the IBE system described above. We construct an algorithm, $\mathcal{S}$, that solves a MDBP instance $(g, g^r, g^{s^{-1}}, e(g,g)^s, \hat{t})$ as follows.

$\mathcal{S}$ randomly chooses two exponents $\gamma_2, \gamma_3 \in_R \mathbb{Z}_p$ and a hash function $H : \{0,1\}^n \to \mathbb{Z}_p$ from a family of universal one-way hash functions. The system parameter *param* is set as (g, h_1, h_2, h_3, H) where $h_1 = g^r$, $h_2 = g^{\gamma_2}$ and $h_3 = g^{\gamma_3}$. $\mathcal{A}$ then returns $g_1 = g^\alpha \in \mathbb{G}$ as the master public key, $\alpha \in \mathbb{Z}_p$ is not given to $\mathcal{S}$ and $\mathcal{S}$ never uses α in the simulation. $\mathcal{S}$ also picks a random element $c \in_R \mathbb{Z}_p$.

To simulate the embedded-identity encryption oracle with message m_i as input (for $i \in \{1, \cdots, q_E\}$), $\mathcal{S}$ selects a random element $d_i \in_R \mathbb{Z}_p$ and returns

$$(u_i, v_i, w_i, y_i) = \left((g^{s^{-1}})^{cd_i}, e(g,g)^{d_i}, m_i/e(g,h_1)^{d_i}, e(g,g)^{d_i(\gamma_2 + \gamma_3 \cdot H(u_i, v_i, w_i))}\right).$$

Let $\hat{s} = e(g,g)^s$. When $\mathcal{A}$ outputs two equal length messages (m_0^*, m_1^*), $\mathcal{S}$ randomly generates a bit b, the challenge ciphertext is given by $\mathfrak{C} = (u, v, w, y) = \left(g^c, \hat{s}, m_b^*/\hat{t}, \hat{s}^{\gamma_2 + \gamma_3 \cdot \beta}\right)$, where $\beta = H(u, v, w)$. From the structure of the ciphertext, the intended recipient's identity ID^* is implicitly defined by $c = s(\alpha - \mathsf{ID}^*)$.

Since $s^{-1}c = s^{-1}s(\alpha - \mathsf{ID}^*) = \alpha - \mathsf{ID}^*$, the ciphertexts returned by the embedded-identity encryption oracle are valid ciphertexts encrypted for ID^*.

After $\mathcal{A}$ receives $\mathfrak{C}$, it outputs b' with probability ϵ at the end of the guess stage. If $b = b'$, $\mathcal{S}$ outputs 0 (meaning $\hat{t} = e(g,g)^{rs}$); otherwise, it outputs 1.

If $\hat{t} = e(g,g)^{rs}$, (u, v, w, y) is a valid, appropriately-distributed challenge to $\mathcal{A}$. If $\hat{t} \neq e(g,g)^{rs}$, since $\hat{t}$ is uniformly random and independent from $\mathcal{A}$'s view (other than the challenge ciphertext), (u, v, w, y) imparts no information regarding the bit b, so we have the success probability equal to

$$\Pr[\hat{t} = e(g,g)^{rs}] \cdot \Pr[\mathcal{A} \text{ succeeds}] + \Pr[\hat{t} \neq e(g,g)^{rs}] \cdot \Pr[b \neq b']$$
$$= (\frac{1}{2})(\frac{1}{2} + \epsilon) + (\frac{1}{2})(\frac{1}{2}) = \frac{1}{2} + \frac{\epsilon}{2} \qquad \square$$

Using a similar argument, SK-IBE [44] can be proven $\mathcal{ACI} - \mathcal{KGC}$-secure.

5.3 $\mathcal{ACI} - \mathcal{KGC}$-Security without User-Anonymity

Now we modify the scheme presented in §5.1 to give a contrived construction in the standard model. The modification just introduces the term g^{ID} to the ciphertext. An immediate consequence is that the modified scheme no longer provides user-anonymity, To revise the $\mathcal{ACI} - \mathcal{KGC}$ proof, the extra term in the challenge ciphertext (and this term appears in all ciphertexts returned by the embedded-identity encryption oracle as well) can be simulated by $g^{\alpha}/(g^{s^{-1}})^{c}$.

5.4 Comparisons with Accountability, Anonymity with Respect to the KGC, and ID-Based Non-malleability

The above scheme can be made to be accountable [30], but other accountable IBE schemes [31,40] are not $\mathcal{ACI} - \mathcal{KGC}$-secure, which shows that accountability is orthogonal to $\mathcal{ACI} - \mathcal{KGC}$-security. For KwrtA-anonymous IBE, [34] showed that BF-IBE [12] is KwrtA but not ID-based non-malleable, a variant of SK-IBE [44] is both KwrtA and ID-based non-malleable, while BB-IBE [8], AHIBE [15] and Gentry-IBE [28] are *not* KwrtA but are ID-based non-malleable. Together with our analysis in §4, it is clear that the notions of KGC-anonymity, ID-based non-malleability and $\mathcal{ACI} - \mathcal{KGC}$-security are independent of each other.

6 Anonymous Private Key Issuing

In anonymous key issuing (AKI), we need to achieve two somewhat contradictory requirements simultaneously. On one hand, the identity of a user should not be leaked, but a user must be authenticated to obtain the corresponding private key. We propose a new system architecture to realize such an AKI protocol, by employing non-colluding identity-certifying authority (ICA) and KGC.

From a high level, the ICA is responsible for issuing each user a certificate on the purported identity after authentication. This certificate is generated using the master certifying key sk_{cert}. The certificate alone would not enable the user to decrypt. The user should contact the KGC who issues a private key based on the certificate presented, but the KGC never gets to know the identity involved in the certificate. The user private key is still generated with the help of the master secret key, that is owned by the KGC and kept secret from the ICA. Figure 1 depicts the certification and the key issuing process. Since the ICA keeps the identities list of the system's users, we make the trust assumption that the ICA does not collude with the KGC (or the KGC can get the identities list easily).

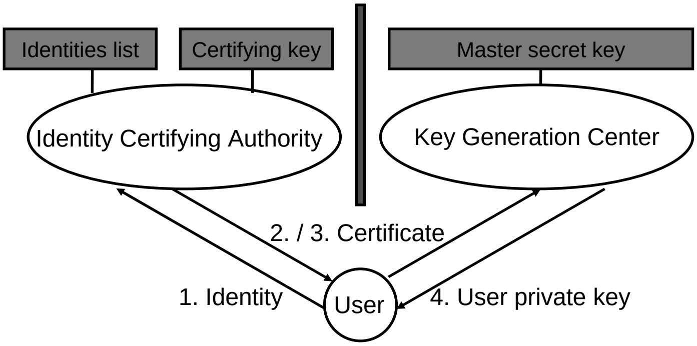


Fig. 1. Our System Architecture

As in PKI, we also assume that the ICA would not impersonate any user. Our solution requires a user to contact two parties before getting a key. Nevertheless, it may be cost-prohibitive to have a globally available KGC to authenticate users and issue keys to users via secure channels in a typical ID-based cryptosystem.

6.1 General Framework

An anonymous key issuing protocol for an IBE scheme consists of four polynomial-time algorithms in additional to the Setup and $\mathsf{MKeyGen}$ algorithms from the IBE. For brevity, the public parameter *param* output by Setup is omitted below.

1. via $(pk_{cert}, sk_{cert}) \overset{\$}{\leftarrow} \mathsf{IKeyGen}()$ the ICA probabilistically outputs the public/private key pair for certification pk_{cert}, sk_{cert};
2. via $(cert, aux) \overset{\$}{\leftarrow} \mathsf{SigCert}(sk_{cert}, \mathsf{ID})$ the ICA probabilistically outputs a certificate for identity ID and some auxiliary information aux;
3. $\mathsf{ObtainKey}(mpk, \mathsf{ID}, cert, aux) \leftrightarrow \mathsf{IssueKey}(sk, cert)$ are two interactive algorithms which execute a user secret key issuing protocol between a user and the KGC. The user takes as input the master public key mpk, an identity ID, and the corresponding certificate $cert$ with auxiliary information aux, and gets a user secret key $usk[\mathsf{ID}]$ as output. The KGC gets the master secret key msk and the certificate $cert$ as input and gets nothing as output.

Here we give a general design framework of such a protocol. We do not claim that any design based on the primitives mentioned here must be secure, but we will analyze the security of our proposed protocol, which is based on the standard argument in anonymous credential literature [5,16].

The first step of our AKI protocol is to get a certificate on an identity from the ICA, which just utilizes a signature scheme. However, the user needs to show this signature to the KGC without leaking the identity (being signed). So the ICA signs on a hiding commitment of the identity instead. This also requires the ability to prove that the contents of a commitment have been signed.

For the KGC side, considering that a user secret key in IBE is essentially a signature on an identity given by the master secret key, obtaining a user secret key without leaking the identity to the KGC boils down to obtaining something similar to a blind signature from the KGC (not to be confused with the signature by the ICA). The blinding step can make a commitment to the identity, the key issuing protocol becomes one for obtaining a signature on a committed value. A crucial difference between our protocol and a blind signature or anonymous credential is manifest at the final stage of our protocol. We require that the user can transform the response from the KGC to a normal signature which directly signs on the value being committed, such that it can be used as the private decryption key of the IBE scheme. In particular, if the final signature just includes a non-interactive proof for proving that the contents of a commitment has been signed, it does not seem to work with any of the existing IBE schemes.

6.2 Security Requirements

One can view $(cert, aux)$ as a signature and $\mathsf{SigCert}$ as the signing algorithm of a signature scheme. For security we require existential unforgeability against adaptive chosen message attack. We omit this standard definition. Our framework assumes $\mathsf{SigCert}$ is used to sign on the (perfectly binding and strongly computationally hiding) commitment of an identity, which is included in *cert*.

Regarding $\mathsf{ObtainKey}$ and $\mathsf{IssueKey}$, we require that malicious users can only get the user private key for the identity "embedded" in the ICA's certificate from the interaction with the KGC, but nothing else. For security protection of the users, we require that the KGC cannot learn anything from the certificate about the real identity of the user. Below is a formalization of the above intuition, which is adopted from some of the security properties of the P-signature [5], a suite of protocols for obtaining signature in a privacy-preserving way.

Definition 9. *An AKI protocol satisfies* issuer privacy *if there exists a simulator* $\mathsf{SimIssue}$ *such that for all PPT adversaries* $(\mathcal{A}_1, \mathcal{A}_2)$,

$$\begin{aligned}
|\Pr\,[\,& param \xleftarrow{\$} \mathsf{Setup}(1^\lambda); (mpk, msk) \xleftarrow{\$} \mathsf{MKeyGen}(param);\\
& (\mathsf{ID}, aux, st) \xleftarrow{\$} \mathcal{A}_1(param, mpk, msk); com \leftarrow \mathsf{Commit}(param, \mathsf{ID}, aux);\\
& b \xleftarrow{\$} \mathcal{A}_2(st) \leftrightarrow \mathsf{IssueKey}(param, msk, com) : b = 1]\\
-\Pr\,[\,& param \xleftarrow{\$} \mathsf{Setup}(1^\lambda); (mpk, msk) \xleftarrow{\$} \mathsf{MKeyGen}(param);\\
& (\mathsf{ID}, aux, st) \xleftarrow{\$} \mathcal{A}_1(param, mpk, msk); com \leftarrow \mathsf{Commit}(param, \mathsf{ID}, aux);\\
& b \xleftarrow{\$} \mathcal{A}_2(st) \leftrightarrow \mathsf{SimIssue}(param, \mathsf{KeyDer}(msk, \mathsf{ID}), com) : b = 1]| < \mathsf{negl}(\lambda).
\end{aligned}$$

Intuitively, this captures the requirement that the protocol itself reveals no information to the adversary (in particular, *msk*) other than a user secret key.

In our definition, both $\mathsf{SimIssue}$ and $\mathsf{IssueKey}$ get an honestly generated commitment, for adversarially chosen identity ID and opening *aux*. Since we assume the commitment is perfectly binding, this automatically guarantees that the identity associated with the commitment is well defined, and only a user secret key corresponding to that particular identity is obtained by the adversary.

For a cleaner definition, $\mathsf{SigCert}$ is not involved. Whether $\mathsf{SimIssue}$ and $\mathsf{IssueKey}$ receives a signature on a commitment of ID or ID itself is just about how their interfaces take ID as the input. We allow $\mathsf{SimIssue}$ to rewind the adversary and it can extract the hidden ID from the commitment.

The above definition assumes the adversary knows msk even its purpose is for the protection of the secrecy of msk. This is adopted from the security definition of secure two-party computation protocols, which models the situation that even the adversary is given some partial information of msk (e.g. through our IBE scheme), it is still unable to distinguish whether it is interacting with a simulator or the real key issuing protocol. Together with the security of the underlying IBE scheme (e.g. CCA2 with access to a user secret key oracle), our definition guarantees that the AKI protocol can be used with the IBE scheme.

Definition 10. *An AKI protocol satisfies* user privacy *if there exists a simulator* $\mathsf{SimObtain}$ *such that for all PPT adversaries* $(\mathcal{A}_1, \mathcal{A}_2)$*,*

$$\begin{aligned}
|\Pr\,[\,& param \xleftarrow{\$} \mathsf{Setup}(1^\lambda), (mpk, \mathsf{ID}, aux, st) \xleftarrow{\$} \mathcal{A}_1(param);\\
& com \leftarrow \mathsf{Commit}(param, \mathsf{ID}, aux);\\
& b \xleftarrow{\$} \mathcal{A}_2(st) \leftrightarrow \mathsf{ObtainKey}(param, mpk, \mathsf{ID}, com, aux) : b = 1]\\
-\Pr\,[\,& param \xleftarrow{\$} \mathsf{Setup}(1^\lambda), (mpk, \mathsf{ID}, aux, st) \xleftarrow{\$} \mathcal{A}_1(param);\\
& com \leftarrow \mathsf{Commit}(param, \mathsf{ID}, aux);\\
& b \xleftarrow{\$} \mathcal{A}_2(st) \leftrightarrow \mathsf{SimObtain}(param, mpk, com) : b = 1]| < \mathsf{negl}(\lambda).
\end{aligned}$$

This models that the protocol reveals no information about the identity ID to the malicious KGC which interacts with the user. Both privacy notions are defined based on a single interaction, but a simple hybrid argument can be used to show that these definitions imply privacy over many sequential instances.

6.3 AKI Protocol for Modified Gentry-IBE

Our protocol extends the interactive protocol for obtaining a signature on a committed value of the first P-signature scheme in [5][4]. We change the signature structure of their scheme so that it fits with the user secret key produced in the modified Gentry-IBE. There are three components sharing the same structure in the key. For brevity, we just show how to build the first component.

Setup: This algorithm executes Setup of modified Gentry-IBE, setups the perfectly binding, strongly computationally hiding commitment and the signature.

$\mathsf{IKeyGen}$: The ICA generates a key pair (pk_{cert}, sk_{cert}) for the signature scheme.

[4] While the signature of the second construction in [5] shares similarity with the user secret key of BB-IBE [8], its second component r cannot be recovered.

$\mathsf{SigCert}$: For $\mathsf{ID} \in \{0,1\}^n$, the ICA creates the certificate $cert = (sig, com, aux)$ by randomly picking[5] aux from the decommitment-string space; and generating a signature sig on $com = \mathsf{Commit}(\mathsf{ID}, aux)$ by running the signing algorithm.

$\mathsf{ObtainKey}(mpk, \mathsf{ID}, cert, aux) \leftrightarrow \mathsf{IssueKey}(msk, cert)$:

1. The user and the KGC engage in a secure two-party computational protocol[6], where the user's private input is (ρ, ID, aux) where $\rho \in_R \mathbb{Z}_p$, and the KGC's private input is α. The KGC then gets a private output which is either $x = (\alpha - \mathsf{ID})\rho$ if $com = \mathsf{Commit}(\mathsf{ID}, aux)$, or $x =\perp$ otherwise.
2. If $x \neq \perp$, the KGC randomly picks[7] $\tau_{\mathsf{ID},1} \in \mathbb{Z}_p$. Then it computes $usk'_{cert} = (usk'_1 = (h_1 g^{-\tau_{\mathsf{ID},1}})^{1/x}, usk'_2 = \tau_{\mathsf{ID},1})$.
3. The user outputs $(usk_1, usk_2) = ((usk'_1)^\rho = (h_1 g^{-\tau_{\mathsf{ID},1}})^{1/(\alpha - \mathsf{ID})}, usk'_2)$.

Analysis. Signer privacy and user privacy follow exactly as in the protocol in [5]. $\mathsf{SimIssue}$ invokes the simulator for the two-party computational (2PC) protocol to extract the adversary's input (ρ, ID, aux), check if $com = \mathsf{Commit}(\mathsf{ID}, aux)$ and sends (usk_1^ρ, usk_2) to the user. $\mathsf{SimObtain}$ also invokes the same simulator to extract the secret key. Then the simulator is given the target output of the computation x, and proceeds to interact with the adversary such that if the adversary completes the protocol, its output is x. In both cases, if the adversary can determine that it is talking with a simulator, it must be the case that the adversary's input to the protocol was incorrect which breaks the security of 2PC.

6.4 Related Work

"Anonymous" private key issuing in ID-based cryptosystems was firstly considered by Sui *et al.* [48], in a system where the duties of authentication and key issuing are separated to local registration authorities (LRAs) and the KGC. Instead of having an LRA to issue a signature, a user supplies a password to the LRA. However, their anonymity guarantee just considers outsider adversaries, and actually an LRA is required to send a list of identities and passwords to the KGC, while our protocol does not require any communication between them.

The "blind" extraction protocols for IBE with leak freeness and selective-failure blindness were proposed in a rigorous manner by Green and Hohenberger [32]. Our notion of issuer privacy is very similar to leak freeness as both are defined in a secure 2PC fashion. A minor difference is that their definition is not coupled with any specific way (e.g. commitment) to hide the identity. Nevertheless, their concrete protocols utilize commitment scheme as well. The motivating

[5] We require that the ICA always use the same aux for a given ID. We can just take aux as the output of a PRF with input ID, for a seed only known to the ICA.

[6] An efficient protocol for securely computing $g'^{1/(sk+m)}$ based on any homomorphic encryption in the standard model [16, §4.3.3] can be used.

[7] If a certificate signing the same commitment is presented later, same $\tau_{\mathsf{ID},1}$ is used.

application in [32] is oblivious transfer, hence the notion of selective-failure blindness considers maliciously generated parameter. Our user privacy is weaker, but it should be fine for our purpose, especially when the KGC is not motivated to induce a selective failure and the user can verify the validity of the key obtained.

As noted in [32], it is non-trivial to come up with an efficient AKI protocol for BF-IBE, another IBE that we showed is $\mathcal{ACI}-\mathcal{KGC}$-secure. However, if one is willing to weaken the security guarantee from 2PC to something like one-more unforgeability of blind signature [6], we conjecture that an efficient AKI protocol for BF-IBE can be constructed similar to the blind signature scheme in [6].

6.5 Applications in Privacy-Preserving Searches on Encrypted Data

Anonymous IBE has attracted attention for the privacy benefits, and as a leverage to construct public key encryption with keyword search [1] as follows. Identity strings are used to represent the keywords. The private key for a particular identity is the trapdoor for testing whether a ciphertext is tagged with a particular keyword. The role of the KGC is now known as the trapdoor generator. To create an encrypted tag, one encrypts a random message using the keyword as the identity in IBE, and appends the message with the tag. To locate the ciphertexts tagged with a keyword, one tries to use a trapdoor to decrypt the tag, and see if the result matches the accompanying message.

Back to our notion, $\mathcal{ACI}-\mathcal{KGC}$ implies that the compromise of the private key does not leak the keyword from an encrypted tag. Our AKI protocol also finds application in privacy-preserving delegated forensic search with authorization, which the government issues a warrant on a keyword to a law enforcing agent (e.g. the police). This warrant is then presented to the encrypted-data owner to indicate that the agent is authorized to ask for a trapdoor for the certified keyword, without revealing what is of forensic interests or (the extreme way of) asking the data owner to surrender the private key. While the idea of privacy-preserving delegated keyword search has been considered, only blind protocols for non-user-anonymous IBE schemes like BB-IBE and Waters-IBE are proposed [32], and without addressing a realistic concern that the hidden keyword should be certified by some authority. We remark that the government can be responsible for the system parameter generation to ensure keyword privacy.

7 Conclusions

We propose a new notion of anonymous ciphertext indistinguishability against KGC attacks ($\mathcal{ACI}-\mathcal{KGC}$), which is orthogonal to existing notions like user anonymity. We modified Gentry's IBE to get an $\mathcal{ACI}-\mathcal{KGC}$-secure IBE in the standard model. We propose a new system architecture with an anonymous key issuing (AKI) protocol to protect the confidentiality of the users identities. We hope that future IBE proposals will consider $\mathcal{ACI}-\mathcal{KGC}$ as one of the key properties, and IBE with $\mathcal{ACI}-\mathcal{KGC}$ or AKI protocol will find more applications.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. J. Crypt. 21(3), 350–391
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
3. Attrapadung, N., Furukawa, J., Gomi, T., Hanaoka, G., Imai, H., Zhang, R.: Efficient Identity-Based Encryption with Tight Security Reduction. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 19–36. Springer, Heidelberg (2006)
4. Baek, J., Safavi-Naini, R., Susilo, W.: Efficient Multi-receiver Identity-Based Encryption and Its Application to Broadcast Encryption. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 380–397. Springer, Heidelberg (2005)
5. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and Noninteractive Anonymous Credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
6. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures based on the Gap-Diffie-Hellman-Group Signature. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
7. Boldyreva, A., Goyal, V., Kumar, V.: Identity-Based Encryption with Efficient Revocation. In: CCS 2008, pp. 417–426 (2008)
8. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
9. Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
10. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
11. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
12. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
13. Boneh, D., Hamburg, M.: Generalized Identity Based and Broadcast Encryption Schemes. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455–470. springer, Heidelberg (2008)
14. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
15. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
16. Chase, M.: Efficient Non-Interactive Zero-Knowledge Proofs for Privacy Applications. PhD thesis, Brown University (2008)

17. Chatterjee, S., Sarkar, P.: Generalization of the Selective-ID Security Model for HIBE Protocols. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 241–256. Springer, Heidelberg (2006)
18. Chatterjee, S., Sarkar, P.: HIBE With Short Public Parameters Without Random Oracle. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 145–160. Springer, Heidelberg (2006)
19. Chatterjee, S., Sarkar, P.: Multi-receiver Identity-Based Key Encapsulation with Shortened Ciphertext. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 394–408. Springer, Heidelberg (2006)
20. Chatterjee, S., Sarkar, P.: New Constructions of Constant Size Ciphertext HIBE Without Random Oracle. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 310–327. Springer, Heidelberg (2006)
21. Chatterjee, S., Sarkar, P.: Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 424–440. Springer, Heidelberg (2005)
22. Chen, L., Cheng, Z.: Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 442–459. Springer, Heidelberg (2005)
23. Chow, S.S.M.: Certificateless Encryption. In: Joye, M., Neven, G. (eds.) Identity-Based Cryptography. IOS Press, Amsterdam (2008)
24. Chow, S.S.M., Choo, K.-K.R.: Strongly-Secure Identity-Based Key Agreement and Anonymous Extension. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 203–220. Springer, Heidelberg (2007)
25. Chow, S.S.M., Roth, V., Rieffel, E.: General Certificateless Encryption and Timed-Release Encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 126–143. Springer, Heidelberg (2008)
26. Dent, A.W., Libert, B., Paterson, K.G.: Certificateless Encryption Schemes Strongly Secure in the Standard Model. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 344–359. Springer, Heidelberg (2008)
27. Galindo, D.: Boneh-Franklin Identity Based Encryption Revisited. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 791–802. Springer, Heidelberg (2005)
28. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
29. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
30. Goyal, V.: Reducing Trust in the PKG in Identity Based Cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg (2007)
31. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-Box Accountable Authority Identity-Based Encryption. In: CCS 2008, pp. 427–436 (2008)
32. Green, M., Hohenberger, S.: Blind Identity-Based Encryption and Simulatable Oblivious Transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
33. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)

34. Izabachène, M., Pointcheval, D.: New Anonymity Notions for Identity-Based Encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 375–391. Springer, Heidelberg (2008)
35. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. J. Crypt. (to appear)
36. Kiltz, E.: From Selective-ID to Full Security: The Case of the Inversion-Based Boneh-Boyen IBE Scheme. Cryptology ePrint Archive, 07/033
37. Kiltz, E., Galindo, D.: Chosen-Ciphertext Secure Threshold Identity-Based Key Encapsulation Without Random Oracles. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 173–185. Springer, Heidelberg (2006)
38. Kiltz, E., Vahlis, Y.: CCA2 Secure IBE: Standard Model Efficiency through Authenticated Symmetric Encryption. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 221–238. Springer, Heidelberg (2008)
39. Libert, B., Quisquater, J.-J.: Identity Based Encryption Without Redundancy. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 285–300. Springer, Heidelberg (2005)
40. Libert, B., Vergnaud, D.: Towards Black-Box Accountable Authority IBE with Short Ciphertexts and Private Keys. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 235–255. Springer, Heidelberg (2009)
41. Naccache, D.: Secure and practical Identity-based Encryption. Inf. Sec. 1(2), 59–64
42. Phan, D.H., Pointcheval, D.: Chosen-Ciphertext Security without Redundancy. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 1–18. Springer, Heidelberg (2003)
43. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 457–473. Springer, Heidelberg (2003)
44. Sakai, R., Kasahara, M.: ID based Cryptosystems with Pairing on Elliptic Curve. Cryptology ePrint Archive, 03/054
45. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on Pairing over Elliptic Curve (in Japanese). In: SCIS 2001 (2001)
46. Seo, J.H., Kobayashi, T., Ohkubo, M., Suzuki, K.: Anonymous Hierarchical Identity-Based Encryption with Constant Size Ciphertexts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 215–234. Springer, Heidelberg (2009)
47. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
48. Sui, A.F., Chow, S.S.M., Hui, L.C.K., Yiu, S.-M., Chow, K.P., Tsang, W.W., Chong, C.F., Pun, K.K.H., Chan, H.W.: Separable and Anonymous Identity-Based Key Issuing. In: ICPADS 2005, pp. 275–279 (2005)
49. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 114–127. Springer, Heidelberg (2003)
50. Weng, J., Liu, S., Chen, K., Ma, C.: Identity-based Parallel Key-Insulated Encryption without Random Oracles. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 409–423. Springer, Heidelberg (2006)
51. Yao, D., Fazio, N., Dodis, Y., Lysyanskaya, A.: ID-based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption. In: CCS 2004, pp. 354–363 (2004)

On the Theory and Practice of Personal Digital Signatures

Ivan Damgård and Gert Læssøe Mikkelsen*

Depertment of Computer Science, Aarhus University

Abstract. We take a step towards a more realistic modeling of personal digital signatures, where a human user, his mobile equipment, his PC and a server are all considered as independent players in the protocol, and where only the human user is assumed incorruptible. We then propose a protocol for issuing digital signatures on behalf of the user. This protocol is proactively UC-secure assuming at most one player is corrupted in every operational phase. In more practical terms, this means that one can securely sign using terminals (PC's) that are not necessarily trusted, as long as the mobile unit and the PC are not both corrupted at the same time. In other words, our solution cannot be broken by phising or key-logging via the PC. The protocol allows for mobile units with very small computing power by securely outsourcing computation to the PC and also allows usage of any PC that can communicate properly. Finally, we report on the results of a prototype implementation of our solution.

1 Introduction

When cryptographic protocols make use of digital signatures, this is usually described in the cryptographic theory literature by saying something of the following form: "Player P_i signs the message m using his secret key sk_i, and sends m and the signature $\sigma_i = S_{sk_i}(m)$ to player P_j".

While this may be a convenient abstraction in some cases, it hides some details that are often very important in practice: in real life a "player" is usually not really a single entity but consists in fact of a human user as well as one or more computing devices he uses in order to store the key and issue the signature. Each of the devices could be corrupted by an adversary without the user being dishonest. In such a case, the theoretic model above would have to consider the entire "player" to be corrupt and would conclude that we can now no longer protect the secret key. However, in real life, it is of obvious interest to protect the user, even if some of his equipment is corrupt.

A well known example of this is the so called man-in-the-middle (or man-in-the-browser) attack which in the context of signatures takes the form of showing the user a message on screen that he would approve, while at the same time trying to have the secret key applied to a different message. If the user's secret key resides on his PC, protected say, by encryption under a password, a man-in-middle attack is always possible if the PC is corrupt.

* Supported by the Danish Strategic research Council.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 277–296, 2009.

In this paper, we first propose a model that we believe reflects in a more realistic way the issues arising when a private person wants to issue digital signatures using current technology. The main players are: a human user U, a mobile unit M, a terminal T, and a server S. In practice, M could be a PDA, a cell-phone or special-purpose device, T could be a PC (but not necessarily the user's own machine), and S could be some server where the user has an account. S could be the user's own machine, or it could be run by a company handling many users – the only assumption we make is that S is on-line whenever a signature is to be issued. In this model, only the user is assumed incorruptible.

We assume that T can interact with U in a standard way via keyboard and screen. M has a screen where it can show a message to be signed and may receive an OK or a reject from the user.

In practice, we would like our solutions to be mobile, i.e., any machine can in principle be used as T, so we therefore consider only protocols where no user specific key material is stored permanently on T. Another practical issue is that a mobile handheld unit can easily be lost or stolen, and it should be possible to securely replace it without having to generate keys (and issue certificates) again. This requires our solution to be secure against an adversary who first steals M and later breaks into T or S.

This issue makes it natural for us to aim for proactive security in the UC model. In proactive security, first introduced in [17], one divides time into *operational phases*, interleaved with (short) *refreshment phases*. In operational phases, the system provides normal service, while refreshment phases are typically used to update key material using fresh randomness. The adversary is assumed to only corrupt a certain number of players in every operational phase, but the set of corrupted players can be different in different phases, meaning that all players may have been corrupt at some point, and the system must still be secure. Our (adaptive) adversary may in each operational phase actively corrupt at most one of M, T, S. The adversary mentioned above who first steals M and later breaks into T or S can be modeled in the proactive framework as an adversary who corrupts M in one operational phase and T or S in the next. Note also that T models any machine(s) that the user U uses as terminal. So if U in real life first uses an untrusted terminal in some Internet cafe and then returns to an uncorrupted PC, this means in our model that T is first corrupted and then becomes honest again.

We stress that our protocols are secure, even if corruption does not take the form of loss of a device: whenever our system is operational, it is secure if the adversary has corrupted at most one of M, T and S, even if the user is not aware of the corruption. In particular this means we are secure against phising or keylogging since this corresponds to corruption of T. On the the other hand if we *know* that M or S have been corrupted, we can make the system operational again, by replacing e.g., a lost M by a new uncorrupted device and restoring the key from a backup.

We model this by introducing a new player, a database, D. This player is only active in the refreshment phase and is used to restore data held by S that could have been erased or corrupted by an active attack. We allow the adversary to corrupt D passively. In practice, we think of D as run by the same party who runs S, and as such the introduction of D models the assumption that the server's organization is able to ensure a backup that is reliable, but not necessarily secret.

The functionality we aim to implement is basically the standard UC functionality for secure signatures, except that the adversary is allowed to stop a signature from being generated [1], and a message is only signed if the user approves it. We then propose a protocol that is secure in this model. The protocol has the following properties:

- To execute the protocol, a user first needs to form the message m to be signed while interacting with T (one may think of using machine T to buy something on the net, where m is a payment order). To sign, he first authenticates himself towards S (typically by entering his log-in data on T), and second he is shown on M's screen the message m and must in response tell M "OK" or "reject".
- To execute the proactive refreshment phase, a user just has to update his log-in data for S, and if M is a new mobile unit (a replacement for a stolen one), he must enter a special code on M (this can be done without many key strokes, e.g., using the camera in a mobile phone to scan a 2-d bar code)
- The protocol can produce as output standard hash-and-sign RSA signatures, compatible with existing PKI's.
- The protocol allows M to use only very little computing power. We can securely outsource most of the computation to T, so that for each signature, M only has to evaluate one pseudo random function and do one addition of large numbers. This is useful if M is a cheap special-purpose device, or if one wants to run a high-level language implementation on M - this allows to cover many types of mobile phones using pure Java, for instance, but it will typically be much slower than device specific code.
- If desired, the protocol can keep the message to be signed secret from S with no loss of efficiency. Note that this can be desirable for privacy related reasons, but could also be undesirable if one wants S to keep a log of what was signed.

On the technical side, we start by borrowing a standard technique from threshold signatures where we share the secret RSA exponent additively between M and S. We then augment this with a new technique allowing the outsourcing to T mentioned above. We also propose an extension of the proactive security model by introducing two kinds of refreshment protocols: one that is done routinely with no other user intervention than a change of password, and one that is invoked in case an attack has been detected, e.g., M has been lost or stolen or a virus attack was detected in S. In such cases we may have lost the information stored on M or S, which means that the secret key is effectively lost as well. We therefore need to design a way to use back-up information stored off-line

[1] It is easy to see that we cannot avoid this in a scenario where only M and S can store key material on-line and either of them may be corrupted.

to securely reestablish the secret key. Finally, keeping the signed message secret from S can be done using standard blinding techniques [8].

In the final part of the paper, we report on results from a prototype implementation of our protocol. In the prototype M was a mobile phone, running a Java application while communicating with T (a PC) via Bluetooth. T then communicates with S via the Internet and SSL. The results show that our outsourcing technique can give a significant speedup, and provide a far better experience for the user.

1.1 Related Work

We are not aware of any previous work that attempts to model our scenario in the UC framework. However, the idea of using a personal (mobile) device to improve security in practice has been studied in several previous papers. In [18], Parno et al. use a mobile phone to set up secure SSL/TLS connections and in [15], Mannan and Oorschot use a personal device to improve security of password authentication. Both solutions basically aim to do user authentication with improved security, in particular to protect against key-logging and phising. In [20], Weigold et al. use a trusted mobile USB device with a display and two buttons to improve security of online services such as Internet banking. All communication between the used PC and the server is routed through this trusted device, where the user has to accept sensitive transactions. [15,20] also contains a good overview of other existing anti-phising techniques and their properties. Finally, there are many examples of using secure devices for transaction authentication, see [2,13], for instance.

The main difference to our results is that the previous works need to assume that the personal device M is uncorrupted (malware-free), while our solution is secure even if M is corrupt, as long as S, T are honest, due to the sharing of the key between M and S. Also, previous works typically does not consider proactive techniques.

As for existing cryptographic techniques, previous literature considers several types of solutions that protect a secret signature key, even if one or more entities are corrupt. A first such technique is known as *Threshold Signature Schemes*, where the secret key is shared among a set of entities called signature servers. One can then have protocols that guarantee security if a majority of servers remain honest. A large body of literature exists on threshold signatures see, e.g., [1,9,12,19]. In these protocols the signature servers play symmetric roles, i.e., they all execute essentially the same program and each server is assumed to be equally hard to break into (hence the honest majority assumption). Protocols for threshold signatures usually assume that the honest servers already agree on the message to be signed and take it from there without considering how such an agreement would be reached in practice.

Therefore, standard threshold signatures are not immediately applicable in our case where a private citizen wants to use digital signatures: he may store key material in different devices with completely different security properties, for instance, on a mobile unit such as a PDA or a cellphone, or on a server.

Also, the idea that all players should agree on the message to be signed, does not really make sense here: we clearly want that only the user decides, and only messages approved by the user get signed. It is not even clear that we want all players to know the messages signed. If a server handling many users is involved, it may be undesirable for privacy reasons that the server knows what is signed.

A second related class of solutions is known as *Key Insulated* and *Intrusion Resilient* signatures [11,14]. In a nutshell, this model and ours are incomparable, but we believe that our approach is the more realistic of the two.

In more detail, while we try to improve security against key exposures by requiring participation of several entities, KI/IR signatures insist that one entity that they call “the user” can sign on its own. Those schemes then instead try to limit the effect of key exposures, by making the users private key valid only for a certain period. When a period expires, the secret key must be updated using a message from a second entity, called the *key base*. Security properties generally hold assuming that user and key base are not simultaneously compromised, so the two trust models are comparable in that only one entity at a time is assumed corrupt. The properties obtained are different, however: If the assumption on our adversary holds, no signature can be generated other than those the user approves, while compromising the user in KI/IR signatures allows generating false signatures in the current period. On the other hand, KI/IR signatures may retain forward security even if both units are compromised in the same period.

The known KI/IR signatures schemes are specially engineered to get the desired security properties. Therefore, certification authorities and receivers of signatures must be aware of the scheme and its special properties to use it. Our scheme outputs completely standard hash-and-sign RSA signatures and so it can co-exist under the same PKI with any other solutions for storing the secret key. This is a very important feature for such a scheme to be useful in a real-life application. A final comment is that existing work on KI/IR signatures assumes that the “user” holds and uses a secret key, and hence ignore the problems stemming from the fact that in real life the secret key must of course be held and operated by some device that is separate from the human user.

2 Security Model

Traditional formal models of digital signatures, e.g., the one described by Canetti in the description of the UC framework [6], are models of the computer used during signature generation and its security. Intuitively, the security we aim for is different, namely: *Only messages accepted by the human user should be signed.*

Modeling human behavior in the UC framework is not an obvious task. Modeling the human ability to decide whether a message should be accepted or rejected would result in an extremely rough approximation, and make our model unclear. Instead we let the environment $\mathcal{Z}$ decide by sending acceptable messages to the model of the human user U, and a protocol is deemed secure if it only outputs signatures on messages that were given to U by $\mathcal{Z}$. Since human users cannot calculate digital signatures and therefore rely on corruptable computing equipment, in our case the terminal T, the output of our ideal functionality is output through T.

The ideal functionality for our mobile signatures $\mathcal{F}_{\text{M-SIG}}$ (Fig. 1) is an extension of the functionality $\mathcal{F}_{\text{SIG}}$ by Canetti [6, Section 7.2.1]. The signer, player S in $\mathcal{F}_{\text{SIG}}$ has been split into: U, S, M and T in our protocol, while the verifier V is the same in both protocols. $\mathcal{F}_{\text{M-SIG}}$ differs from from $\mathcal{F}_{\text{SIG}}$ in that: 1) The message m to be signed has to be sent to U and 2) The adversary is able to stop the generation of a signature (in the model, he can do so by stopping delivery of output from $\mathcal{F}_{\text{M-SIG}}$). The main idea behind $\mathcal{F}_{\text{M-SIG}}$ is to not specify a particular signature algorithm, but to keep track of messages that have been submitted for signing and accept only these messages as signed. This is the reason for the signature verification part of $\mathcal{F}_{\text{M-SIG}}$, which at first may seem a bit counter-intuitive.

Ideal functionality $\mathcal{F}_{\text{M-SIG}}$

Key Generation: Upon receiving a value $(\texttt{KeyGen}, \mathit{sid})$ from U, verify that $\mathit{sid} = (\mathit{uid}, \mathit{sid}')$ for some valid uid and sid'. If not then ignore the request. Else, hand $(\texttt{KeyGen}, \mathit{sid})$ to the adversary $\mathscr{S}$. Upon receiving $(\texttt{Algorithms}, \mathit{sid}, s, v)$ from $\mathscr{S}$, where s is a description of a PPT ITM, and v is a description of a deterministic polytime ITM, output the message $(\texttt{VerificationAlgorithm}, \mathit{sid}, v)$ to U.

Signature Generation: Upon receiving a value $(\texttt{Sign}, \mathit{sid}, m)$ from U, let $\sigma \leftarrow s(m)$, and verify that $v(m, \sigma) = 1$. If so, send a public delayed output $(\texttt{Signature}, \mathit{sid}, m, \sigma)$ message to T, and record the entry (m, σ) when the signature has been outputted. If $v(m, \sigma) \neq 1$ output $\perp$ to T.

Signature Verification: Upon receiving a value $(\texttt{Verify}, \mathit{sid}, m, \sigma, v')$ from V, do: If $v' = v$, $v(m, \sigma) = 1$, and no entry (m, σ') for any σ' is recorded, then output an error message to U and halt. Else, output $(\texttt{Verified}, \mathit{sid}, m, v'(m, \sigma))$ to V.

Fig. 1. Ideal functionality for *"secure mobile digital signatures"*, based on $\mathcal{F}_{\text{SIG}}$ in [6]

3 Protocol Securely Realizing $\mathcal{F}_{\text{M-SIG}}$

The main player in our protocol is the human user U. The main idea behind the protocol is to protect the user from a corrupt terminal or a corrupt mobile unit by letting him accept the message to be signed on both the terminal and on the mobile device; and to ensure that the signature cannot be generated unless both units received an accept from U. This can be implemented by secret sharing the private exponent d of the user's private key, with a simple additive secret sharing.

Secret sharing: The following additive sharing scheme is used to secret share the user's secret RSA key $sk = \langle d, N \rangle$: The private exponent d is shared to a uniform randomly chosen d_1 and a value d_2 s.t.:

$$d \equiv d_1 + d_2 \bmod \varphi(N) \qquad (1)$$

Hence, for any m, we have:

$$m^d \bmod N = m^{d_1} m^{d_2} \bmod N \tag{2}$$

Note that (2) still holds if the addition in (1) is done over the integers.

By giving d_1 to the mobile device M and d_2 to the server S, a simple protocol realizing $\mathcal{F}_{\text{M-SIG}}$ can be implemented. In this protocol the message, if accepted by the user on the terminal T is sent to M and to S. Then M shows the message on its screen and will sign the message with its exponent share if the user accepts the message. S will sign the message with the other exponent share if the correct password is typed into T and sent to S. Finally, T can assemble the complete signature by multiplying the two "half signatures".

This protocol, however, requires M to do a full scale exponentiation. This is problematic because we want to include cases where M is a special purpose, cheap and small device, or where the software running on M is high-level code only, for portability (such as pure Java on a mobile phone). The Chinese Remainder Theorem (CRT) method[2] often used to speed up RSA exponentiation cannot be used here, since this would reveal the factorization of N to the mobile device. Alternatively, we could make M's exponent share be a number (much) smaller than d. This would speed up M's computation, but would reveal significant information to S about d. We do not know if this is secure, but we strongly suspect it is not. We therefore propose to exploit the fact that T is likely to have much more computing power than M. Doing this requires some changes in the protocol, as explained in the following section.

3.1 Protocol $\pi_{\text{M-SIG}}$, for Computationally Limited M

$\pi_{\text{M-SIG}}$ assumes keys are set up beforehand, this is done by using an ideal functionality $\mathcal{F}_{\text{KeyGen}}$ (Fig. 2) for generation and distribution of keys and password. Because the signatures generated follow existing standards, verification is normal RSA signature verification without any communication involved. This means that the main part of $\pi_{\text{M-SIG}}$ is generation of signatures.

Key Generation. $\mathcal{F}_{\text{KeyGen}}$ Will generate a password *pwd* for the user, generate keys, and share the user's private key. After *pwd* and keys have been generated they are distribute to the respective players.

Signature Generation. When U receives a value $(\texttt{Sign}, sid, m)$, m is forwarded to T together with the password *pwd*, then T starts a signing protocol where U is asked from M if m_M should be signed. U will accept message m_M from M iff $m_M = m$. Before the formal definition of $\pi_{\text{M-SIG}}$ we need to define some components of the protocol.

Definition 1. *$\mathcal{H}(m)$ denotes the hashing and padding applied to the message m before the exponentiation is done in the used RSA signature scheme.*

[2] For a description of the CRT speedup method see [3, 5.2].

Ideal functionality $\mathcal{F}_{\mathbf{KeyGen}}$

Upon receiving a value $(\texttt{KeyGen}, sid, \kappa)$ from player U, if $sid = (uid, sid')$ for some valid uid; then generate a password pwd, and a pair of RSA keys $(sk = \langle d, p, q\rangle, pk = \langle e, N\rangle)$ with security parameter κ, and compute:

$$\begin{aligned} d_M &\in_{\mathrm{R}} [1, \varphi(N) - 1] \\ d_S &\leftarrow d - d_M \mod \varphi(N) \\ k &\in_{\mathrm{R}} \{0,1\}^{\kappa} \end{aligned}$$

Send $\langle d_M, k, N\rangle$ to M, $\langle d_S, k, pwd, e, N\rangle$ to S, pwd to U and $\langle e, N\rangle$ to the adversary and halt.

Fig. 2. Functionality $\mathcal{F}_{\mathrm{KeyGen}}$. Generating keys and a password for the user

Hashing and padding is needed to make RSA signature schemes secure against *chosen plaintext attacks*, and is therefore already used in most standards. Our protocol is secure no matter how $\mathcal{H}$ works, as long as combining $\mathcal{H}$ and RSA results in a secure signature scheme. Hashing can also give some amount of privacy because the server only needs to see $\mathcal{H}(m)$ and not m itself. If desired, blinding can be applied to ensure the users privacy unconditionally, see section 6. If logging is desired on the server, the server can do the hashing and padding and m itself can be sent around in the protocol.

Definition 2. $F_k(\cdot)$ *denotes a* secure pseudo-random function *with $\widetilde{\kappa} + \kappa$-bit output and key k, with $\tilde{\kappa}$ being the length of the RSA keys. More precisely, a polynomial time bounded adversary who gets oracle access to either $F_k()$ or a random function cannot distinguish the two alternatives with an advantage that is non-negligible (in the length of k).*

An overview of the protocol can be found in Fig. 3. In case of errors during execution of the signing protocol, players communicating with the terminal T will send $\perp$ to T and T will then stop the protocol and return $\perp$ to the environment $\mathcal{Z}$. The protocol is executed the following way: First the message m is sent to the user U from $\mathcal{Z}$, and U sends m to T together with pwd. T will now send m to the mobile device M. M sends m_M $(m_M \leftarrow m)$ to U and U returns $(\texttt{accept})$ if $m_M = m$, else U rejects and $\perp$ is returned to M and forwarded to T. If U accepts, M calculates δ_M and sends δ_M to T.

$$\delta_M \leftarrow F_k(\mathcal{H}(m)) + d_M \tag{3}$$

The value δ_M is a blinding of the key share d_M known to M. Because k is unknown to T, T can do the exponentiation without gaining knowledge of d_M (the blinding is later removed by S).

When T receives δ_M, T calculates σ_M and $\mathcal{H}(m)$ and send these values and pwd to S.

$$\sigma_M \leftarrow \mathcal{H}(m)^{\delta_m} \mod N \tag{4}$$

When S receives σ_M, $\mathcal{H}(m)$ and pwd it checks if pwd is correct, if not $\perp$ is sent back to T. Else σ_S (6) and σ (7) are calculated and S checks if σ is a valid signature of $\mathcal{H}(m)$, if this is the case σ is sent back to T, if not $\perp$ is sent back. Note that sending σ_M to S lets S calculate and verify σ, and thus indirectly check that m has been accepted by U through both T and M. The protocol *might* be secure without this check, however, our proof requires it. Furthermore when we later extend the protocol to be proactive, this check make some recoveries simpler and thereby more user friendly.

$$\begin{aligned}
\delta_S &\leftarrow d_S - F_k(\mathcal{H}(m)) &(5)\\
\sigma_S &\leftarrow \mathcal{H}(m)^{\delta_S} \mod N &(6)\\
\sigma &\leftarrow \sigma_M \times \sigma_S \mod N &(7)\\
&= \mathcal{H}(m)^{d_M + F_k(\mathcal{H}(m)) + d_S - F_k(\mathcal{H}(m))} \mod N &(8)\\
&= \mathcal{H}(m)^d \mod N &(9)
\end{aligned}$$

Communication: The model we use assumes communication to be secret from the adversary unless he has corrupted one of the communicating parties. The real-life justification for this differs between the different communication channels used. For key generation the ideal functionality $\mathcal{F}_{\text{KeyGen}}$ is used. Formally, communication with ideal functionalities is done over perfect secure channels and the functionality specifies what to leak to the adversary. $\mathcal{F}_{\text{KeyGen}}$ leaks the public key and the fact that keys have been generated. $\mathcal{F}_{\text{KeyGen}}$ is thought of

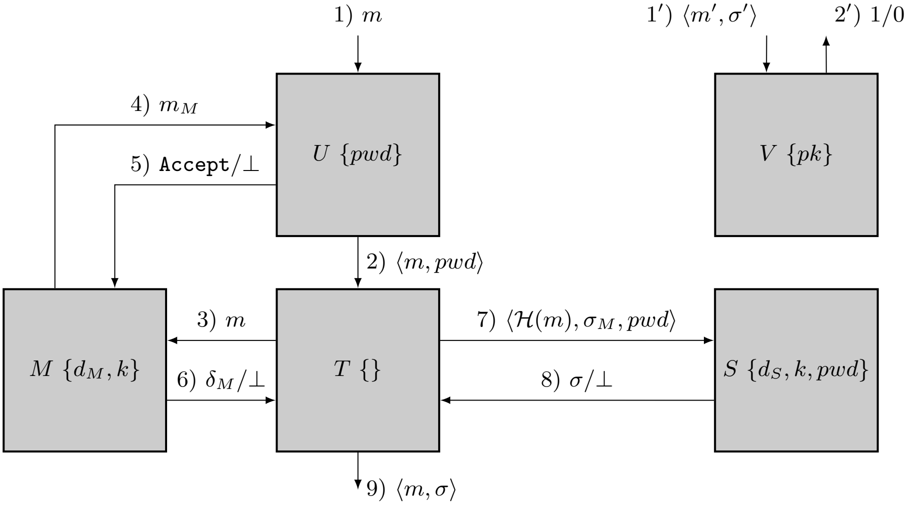


Fig. 3. Overview of the signing phase in $\pi_{\text{M-SIG}}$. For simplicity sid is left out of all messages in this overview. Values sent to the players during key generation are presented after the name of the player.

either: as a trusted third player, in which case secure encrypted communication is a reasonable assumption; or alternatively as a protocol doing secure shared key generation e.g., [4], in which case communication to the protocol is done locally.

During signature generation different communication channels are used, these channels are modeled by an ideal functionality "Secure message transmission" $\mathcal{F}_{\text{SMT}}$ delivering the message n and leaking the length of n to the adversary. For a concrete formal definition of $\mathcal{F}_{\text{SMT}}$ see [6, section 6.3]. Communication that involves U models what the user sees and types on the terminal or mobile device and is therefore assumed secure against adversaries located physically away.

Communication between T and S is done over a cryptographically secured channel (but it only has to be secure if T and S are honest). This can be done using SSL/TLS if an appropriate public-key infrastructure is in place, but a password-based key exchange such as SRP or a password-based cipher suite for TLS [5] is a more natural and secure solution (as this avoids problems like selection of the right certificate to use for obtaining S's public key).

For communication between M and T, there are two possible justifications for assuming it to be secure: 1) Communication happens over a secure connection, this could be via USB cable or a connection where security is based on cryptography; the latter case can be feasible even if M is computationally weak, namely if RSA with a small public exponent is used, or in case we use a secure Bluetooth protocol with pairing. 2) We could also base ourselves on the fact that the communication only has to be secure if M, T are honest and S is corrupt. Since S is typically located physically away from T and M, one may decide that unencrypted communication is good enough if done such that it can only be picked up in physical proximity.

4 Protocol $\pi_{\text{M-SIG}}$ UC-realizes $\mathcal{F}_{\text{M-SIG}}$

In this section we will prove that under appropriate assumptions $\pi_{\text{M-SIG}}$ UC-realizes $\mathcal{F}_{\text{M-SIG}}$. The security of $\pi_{\text{M-SIG}}$ is obviously based on the security of the underlying RSA signature scheme, hence we need:

Assumption 1. *With proper choice of hashing and padding function $\mathcal{H}(\cdot)$, the underlying RSA signature scheme: $Sig(m) = \mathcal{H}(\cdot)^d \mod N$ is secure against* adaptive chosen plaintext attacks.

Our protocol clearly needs that U can securely authenticate himself towards S. For concreteness, we have specified that this happens using a password *pwd*, but in fact any authentication method could be used, as long as it is secure against an adversary that does not corrupt S or T. We have chosen to leave out the details of the authentication and its security by simply assuming that the adversary cannot with significant probability get the password except by corrupting a player who has seen it. Actually, since the adversary has to do online attacks if trying to guess *pwd*, and the server implementation can take this into account, this assumption may be justifiable. Other ways and discussions about modeling password security in the UC framework can be found in [7].

Assumption 2. *If the adversary does not corrupt S or T, he can produce the correct password with only negligible probability.*

Theorem 1 ($\pi_{\textbf{M-SIG}}$ UC-realizes $\mathcal{F}_{\textbf{M-SIG}}$). *Under assumptions 1 and 2 and if F_k is secure, $\pi_{M\text{-}SIG}$ UC-realizes $\mathcal{F}_{M\text{-}SIG}$ with respect to adaptive and active adversaries, corrupting at most one of the players: T, M or S.*

Proof. For any real world adversary $\mathcal{A}$ interacting with $\pi_{\text{M-SIG}}$, and corrupting at most one of the players: T, M or S, we need to show that there exists a simulator $\mathscr{S}$ interacting with $\mathcal{F}_{\text{M-SIG}}$, such that no PPT environment $\mathcal{Z}$ can distinguish $\mathcal{A}$ interacting with $\pi_{\text{M-SIG}}$ from $\mathscr{S}$ interacting with $\mathcal{F}_{\text{M-SIG}}$. The proof of this is done in two steps: First we present an $\mathscr{S}$ capable of simulating $\pi_{\text{M-SIG}}$ for all $\mathcal{A}$, except if $\mathcal{Z}$ is able to produce a forged signature (i.e., a signed message not accepted by the user). Next we present a reduction, which, given a $\mathcal{Z}$ capable of forging signatures, can use $\mathcal{Z}$ to forge "normal" RSA signatures, and thereby breaking assumption 1. Consequently, simulation only fails with negligible probability.

Simulating $\pi_{\textbf{M-SIG}}$. The simulator $\mathscr{S}$ needs to simulate the adversary's view of $\pi_{\text{M-SIG}}$, when the players forwards all input to $\mathcal{F}_{\text{M-SIG}}$ instead of running $\pi_{\text{M-SIG}}$. $\mathscr{S}$ has to simulate the leakage (i.e., the length of the sent data) of communication. If a player is corrupted, $\mathscr{S}$ in addition has to simulate the view of this player. $\mathscr{S}$ will do this by generating keys following the algorithm of $\mathcal{F}_{\text{KeyGen}}$ and sending the expected keyshares to corrupt players. Since it knows all secret keys, it can now simulate $\pi_{\text{M-SIG}}$ by simply running the protocol. It is evident that verification of signatures is the only way for $\mathcal{Z}$ to distinguish simulation from the real protocol. Invalid signatures will be rejected both in the real and the ideal world, genuine valid signatures will be accepted in both worlds. However, forged signatures will only be accepted in the real world since $\mathcal{F}_{\text{M-SIG}}$ enforces unforgeability. Thus $\mathscr{S}$ simulates $\pi_{\text{M-SIG}}$ perfectly except if the environment $\mathcal{Z}$ is able to produce a forged signature.

Reduction. To prove that $\pi_{\text{M-SIG}}$ provides unforgeability, we construct a reduction that - if $\pi_{\text{M-SIG}}$ is insecure - can beak the underlying RSA signatures scheme, and thereby violate assumption 1. The idea is that if there exist a PPT environment $\mathcal{Z}$ that with nonnegligble probability can forge signatures, based on information gained by controlling a corrupted player; we would be able to use $\mathcal{Z}$ to forge an RSA signature in polynomial time. The reduction Red_{RSA} is formally described in Fig. 4. A forge of an RSA signature is modeled by giving Red_{RSA} access to an RSA oracle $\mathcal{O}_{\text{RSA}}$. $\mathcal{O}_{\text{RSA}}$ will return a public RSA key to Red_{RSA} when prompted, and sign messages when Red_{RSA} sends them. We say that Red_{RSA} has forged an RSA signature successfully, if Red_{RSA} can output a signature on a message that has not been signed by $\mathcal{O}_{\text{RSA}}$.

We need to prove that communicating with Red_{RSA} is indistinguishable from $\pi_{\text{M-SIG}}$, so $\mathcal{Z}$ will behave the same way in both cases. It is evident that simulating communication (i.e., leak the length of data sent) can be done.

If $\mathcal{A}$ corrupts M, $\mathcal{Z}$ learns the random variables k and d_M. k has the same distribution in both cases, while d_M is uniform random in $[1, \varphi(N) - 1]$ in $\pi_{\text{M-SIG}}$ and d_M in Red_{RSA} is uniform random in $[1, N]$. The two distributions are, however, statistically close. In both cases $\mathcal{Z}$ also learns all messages m signed so far, and since neither M or T has been corrupted, U has accepted them all. So all input is indistinguishable when corrupting M.

The reduction $Red_{\mathbf{RSA}}$

Red_{RSA} takes the following inputs: an environment $\mathcal{Z}$; the security parameter κ, the length of RSA keys $\widetilde{\kappa}$ and an RSA oracle $\mathcal{O}_{\text{RSA}}$.

Key Generation: Upon receiving $(\texttt{KeyGen}, sid)$ from U, verify that $sid = (uid, sid')$ for some valid uid and sid'. If not then ignore the request. Else, ask $\mathcal{O}_{\text{RSA}}$ for the public RSA key $pk = \langle e, N\rangle$ and output pk as $(\texttt{VerificationAlgorithm}, sid, v(pk))$ public delayed to U, $v(pk)$ being the verification algorithm, with public key pk.

Signature Generation (all honest): Upon receiving $(\texttt{Sign}, sid, m)$ from U. Send $(\texttt{Sign}, sid, m)$ to $\mathcal{O}_{\text{RSA}}$, wait for a signature σ of m from $\mathcal{O}_{\text{RSA}}$, store $\langle m, \sigma\rangle$ and output $(\texttt{Signature}, sid, m, \sigma)$ to T.

Signature Verification: Upon receiving $(\texttt{Verify}, sid, m, \sigma, v')$ from V, do: If $v' = v$, $v(m, \sigma) = 1$, and no entry m is recorded, then output $(\texttt{RSA-Broken}, m, \sigma)$ and halt. Else, output $(\texttt{Verified}, sid, m, v'(m, \sigma))$ to V.

Corruption of M: Pick $d_M \in_{\text{R}} [1, N]$, $k \in_{\text{R}} \{0,1\}^{\widetilde{\kappa}+\kappa}$ and send d_M, k and all stored messages m as simulated input to M. From now on when $\mathcal{Z}$ sends m to U, send m to M. If M thereafter sends $m' \neq m$ to U, return $\perp$ to M, if M on the other hand sends m to U, return $(\texttt{Accept})$ to M. If M sends $\delta_M \equiv d_M + F_k(\mathcal{H}(m)) \mod \varphi(N)$ to T after having received m, output $(\texttt{Signature}, sid, m, \sigma)$ to T, else output $\perp$ to T.

Corruption of T: Pick pwd as $\mathcal{F}_{\text{KeyGen}}$ would, and from all stored pairs $\langle m, \sigma\rangle$ calculate the simulated input (m, pwd, δ_M, σ) of T:

$$\delta_M \in_{\text{R}} \{0,1\}^{\widetilde{\kappa}+\kappa} \tag{10}$$

When $\mathcal{Z}$ sends m to U, send $\langle m, pwd\rangle$ to T. If T now sends $m' \neq m$ to M, return $\perp$. If $m' = m$ return a new random δ_M (10) to T. If T now sends $\langle \mathcal{H}(m), \sigma_M, pwd\rangle$, with $\sigma_M = m^{\delta_M} \mod N$ to S return the signature σ of m. If T at any point sends a correct triple $\langle \mathcal{H}(m'), \sigma'_M, pwd\rangle$ of a previous signed message m', return the signature σ' for m'. If T sends anything else to S return $\perp$ to T.

Corruption of S: Pick $d_S \in_{\text{R}} [1, N]$, $k \in_{\text{R}} \{0,1\}^{\widetilde{\kappa}+\kappa}$ and pwd as $\mathcal{F}_{\text{KeyGen}}$ would. From all stored pairs $\langle m, \sigma\rangle$ calculate the simulated input (d_S, k, $\mathcal{H}(m)$, pwd, σ_M) of S:

$$\sigma_M \leftarrow \sigma \times \left(\mathcal{H}(m)^{(d_S - F_k(\mathcal{H}(m)))}\right)^{-1} \mod N \tag{11}$$

When $\mathcal{Z}$ sends m to U, send $\langle \mathcal{H}(m), \sigma_M, pwd\rangle$ to S, if σ is returned output $(\texttt{Signature}, sid, m, \sigma)$ to T, else output $\perp$ to T.

Fig. 4. Reduction from forgery by interacting with $\pi_{\text{M-SIG}}$ to forgery of normal RSA signatures

Both U in $\pi_{\text{M-SIG}}$ and Red_{RSA} acting as U will accept only a genuine message m; furthermore both in Red_{RSA} and in $\pi_{\text{M-SIG}}$ sending $\delta_M \equiv d_M + F_k(\mathcal{H}(m)) \mod \varphi(N)$, but nothing else, to T will result in a signature. This proves that controlling the output of M does not give $\mathcal{Z}$ the ability to distinguish between $\pi_{\text{M-SIG}}$ and Red_{RSA}.

If $\mathcal{A}$ corrupts T, $\mathcal{Z}$ learns pwd, k, all signed messages m and there signatures σ, the distribution of these are equal in the two cases. In addition $\mathcal{Z}$ learns δ_M. In Red_{RSA} δ_M is uniformly chosen in $\{0,1\}^{\widetilde{\kappa}+\kappa}$, whereas in $\pi_{\text{M-SIG}}$, $\delta_M = d_M + F_k(\mathcal{H}(m))$. By definition (2) F_k is indistinguishable from a uniform chosen $\widetilde{\kappa}+\kappa$ bit value, and since d_M is $\widetilde{\kappa}$ bits long, the two distributions are statistically close.

When T sends a message m to M, M will in both cases return a indistinguishable δ_M if m did originate from U, while $\perp$ is returned to everything else. T sending S a correct triple $\langle \mathcal{H}(m), \sigma_M, pwd\rangle$ will in both cases result in S returning a signature σ on m, the same is the case with a correct triple $\langle \mathcal{H}(m'), \sigma'_M, pwd\rangle$ from an earlier signed message m'. On the contrary in Red_{RSA} T would get $\perp$ back, if a correct triple $\langle \mathcal{H}(\bar{m}), \bar{\sigma_M}, \bar{pwd}\rangle$ of a not yet signed message $\bar{m}$ is send to S, whereas S in $\pi_{\text{M-SIG}}$ would produce a signature $\bar{\sigma}$ of $\bar{m}$. Since we assume that F_k is secure (definition 2), the probability of $\mathcal{A}$ producing a correct δ_M and thereby a correct σ_M is, however, negligible. All other data send from T to S will in both cases result in $\perp$ in return. So corrupting T will not let $\mathcal{Z}$ distinguish.

If $\mathcal{A}$ corrupts S, $\mathcal{Z}$ learns pwd, the hash values $\mathcal{H}(m)$ and the signatures σ of all signed messages, the distributions of these are equal in both cases. $\mathcal{Z}$ also learns σ_M of the signed messages which has been constructed different in the two cases; however, if $\mathcal{H}(m)^{(d_S - F_k(\mathcal{H}(m)))}$ has an inverse[3] $\mod N$ then:

$$\sigma \equiv \sigma_M \times \sigma_S \mod N \tag{12}$$

$$\Rightarrow \sigma_M \equiv \sigma \times \sigma_S^{-1} \equiv \sigma \times \left(\mathcal{H}(m)^{(d_S - F_k(\mathcal{H}(m)))}\right)^{-1} \mod N \tag{13}$$

So $\mathcal{Z}$ cannot distinguish Red_{RSA} from $\pi_{\text{M-SIG}}$ by the input to S. If S sends a correct signature σ this signature is the output of both Red_{RSA} and $\pi_{\text{M-SIG}}$ and anything else results in $\perp$ in both cases.

This proves that under the assumptions 1, 2 and that F_k is secure no PPT environment can forge signatures, and thereby we have proven Theorem 1.

5 Proactive Security

As pointed out in the introduction, proactive secure protocols contain alternating operational and refreshment phases, where the latter are used to refresh the stored key material.

One way to do refreshment is to update the user password and then reshare d by adding a random value to one share and subtract the same value from

[3] If not, we can construct a nontrivial factor of N, and thereby forge RSA signatures of arbitrary messages.

the other share. This solution is the *Refreshment* protocol described below, and this does in fact give us proactive security. This may seem strange since the adversary can do an active attack on M, for instance, and delete M's share of the key. Then we can never issue signatures again, but formally speaking, this is not a problem because our ideal functionality allows the adversary to stop signatures from being generated. However, in the real world, we would want to be able to get the system operational again, particularly in case M (e.g., the users mobile phone) is lost or stolen. This can be thought of as a corruption of M, where in addition it becomes known to the honest players that M has been corrupted. If such an event happens, we can exploit the knowledge that M was corrupted, to replace it by a new uncorrupted device and reestablish the key sharing from a back-up. This is done in the alternative *Refreshment** protocol below.

5.1 Proactive Definition of Security

Technical details on how to model proactive security in the UC framework can be found in [1] and [6]. We give a short summary here: As usual the adversary may corrupt players (adaptively), but when a player is corrupted, the adversary is not given the complete history of that player (contrary to the standard case), only the history dating back to the start of the current operational phase. The adversary may decide to leave a corrupted player when a refreshment begins, and this player now again follows the protocol, starting from some default state. The adversary may then corrupt a new player in the next operational phase, as long as the number of corrupted players stays below the specified threshold. Corruption during a refreshment phase is not allowed or, better said, it counts as if the involved player is also corrupt in both the previous and following operational phases.

It is standard to let the environment decide when a refreshment phase begins by sending *refreshment* as input to all honest players. Motivated by the above discussion, we extend the model by allowing the environment to send either *refreshment*, signaling the start of a routine refreshment, or *refreshment**, signaling the start of a refreshment where a device has to be set up from scratch, but is assumed honest. We assume that the environment only sends *refreshment** if this makes sense, that is when the adversary has left a player, and we therefore can assume the players to be honest during the *refreshment** phase. This models the case where the mobile device was lost and a new, not yet corrupted one is to be set up, or when a virus attack on the server has been detected, but after clean-up and reboot, we believe the server is honest again.

5.2 Protocol $\pi_{\mathcal{P}\text{-M-SIG}}$, a Proactive Version of $\pi_{\text{M-SIG}}$

The ideal functionality we want to implement is $\mathcal{F}_{\text{M-SIG}}$, as in previous sections. The protocol $\pi_{\mathcal{P}\text{-M-SIG}}$ is a proactive version of $\pi_{\text{M-SIG}}$. To be able to do *refreshment** we introduce a new player D in the protocol. D is a database with the property that the entry for user U only is writable during key generation, therefore we only allow passive corruption of D. During key generation a backup share of d is given to the human user, and during the *refreshment** phase the user has to send this share to the mobile device. The length of this share is evidently beyond the capacity of information users can enter on a keyboard. Nevertheless this can be solved by e.g., storing the share as a 2D barcode on paper, and send it to M via a camera, if M is a camera equipped cellphone. Another solution is to store the share on a USB-pen or similar. A third solution is to give a short key to the user and generate the share pseudo-randomly from the key. It is important to understand that the back-up share is erased from M as soon as the refreshment is over, and the method is therefore secure under the assumption that a fresh mobile device will stay honest in the short time it takes the refreshment to complete.

Formal Description of $\boldsymbol{\pi_{\mathcal{P}\text{-M-SIG}}}$. Signature generation and verification is handled in the same way as in $\pi_{\text{M-SIG}}$.

Key Generation. Like in $\pi_{\text{M-SIG}}$ key generation is handled by an ideal functionality, $\mathcal{F}_{\mathcal{P}\text{-KeyGen}}$ is defined as $\mathcal{F}_{\text{KeyGen}}$ (Fig. 2) with the following extensions: First to simplify[4] our security proof we blind the sharing of d with $d_\Delta \in_{\text{R}} [-2^\kappa N, 2^\kappa N]$ s.t. $d_S \leftarrow d_S + d_\Delta$ and $d_M \leftarrow d_M - d_\Delta$, blinding happens during the *refreshment* phase, so the small expansion of the shares is not an issue. Second a backup sharing $\langle \widehat{d_M}, \widehat{d_S} \rangle$ of d is computed. $\widehat{d_M}$ is sent to U in addition to the values sent by $\mathcal{F}_{\text{KeyGen}}$, and $\widehat{d_S}$ is sent to, and stored by D. This backup sharing is required for *refreshment**.

Communication. Communication is done with the same assumptions as in $\pi_{\text{M-SIG}}$, with an extra secure line from S to M, which is used once during each *refreshment** phase. This line should either be thought of as provided by the mobile phone network, if M is a cellphone, or a physical link between S and M, if M is some special purpose device.

Refresh k, resharing of d

1. S does the following calculations ($\mathcal{E}_k^S$ denotes a semantically secure symmetric encryption scheme, with message authentication and key k), and sends $\widetilde{ref}$ to M through T:

$$k_{new} \in_{\text{R}} \{0,1\}^\kappa, \quad d_\Delta \in_{\text{R}} [-2^\kappa N, 2^\kappa N], \quad \widetilde{ref} \leftarrow \mathcal{E}_k^S(k_{new}, d_\Delta) \tag{14}$$

$$k \leftarrow k_{new}, \quad d_S \leftarrow d_S + d_\Delta \tag{15}$$

2. M decodes $\widetilde{ref}$ with k and let $k \leftarrow k_{new}$ and $d_M \leftarrow d_M - d_\Delta$.
3. **Erasure:** M and S erases d_Δ and the old values of d_M, d_S and k.

Fig. 5. Refreshing d_M, d_S and k in the *refreshment* phase of $\pi_{\mathcal{P}\text{-M-SIG}}$

[4] Blinding from start avoids treating the first operational phase as a special case.

Refreshment and Refreshment*. The *Refreshment* protocol first does refresh of k and resharing of d, see Fig. 5, and then pwd is updated using the protocol in in Fig. 7. To check if we are indeed in a valid state, a "test-signature" can be issued afterwards.

The *Refreshment** protocol first creates a new value of k and shares of d from the backup, see Fig. 6. It then does the refreshment of pwd as in Fig. 7 using a blank password as the old value of pwd because the adversary might have changed the password if he had control of S earlier.

A couple of remarks on the protocol for updating passwords: we append a 1-bit to the input of F_k since then no input used in refreshment can be equal to inputs used in signature generation, so a corrupt T does not get F_k-values that can be misused later. Using the string s to hide pwd_{new} is not strictly necessary in our model - since we assume a secure channel from T to S when both are honest, T could just send pwd_{new}. In practice, however, if the channel is set up using password-based key exchange as discussed earlier, it is not secure if the old password has been compromised, say, by an earlier attack on T. Using the extra hiding under s solves this problem.

Restore k and shares of d from backup

1. D sends $\widehat{d_S}$ to S. S chooses $k \in_{\mathrm{R}} \{0,1\}^{\kappa}$ and $d_{\Delta} \in_{\mathrm{R}} [-2^{\kappa}N, 2^{\kappa}N]$, sets $d_S \leftarrow \widehat{d_S} + d_{\Delta}$ and sends $\langle k, d_{\Delta} \rangle$ to M.
2. M sets k to the received value and sends (`send-backup`) to U.
3. U returns $\widehat{d_M}$ to M and M sets $d_M \leftarrow \widehat{d_M} - d_{\Delta}$.
4. The protocol for refreshment of pwd (Fig. 7) is run, with old password pwd being blank.
5. **Erasure:** M and S erases $\widehat{d_M}$, $\widehat{d_S}$, d_{Δ} and the old values of d_M, d_S and k.

Fig. 6. *Refreshment** phase of $\pi_{\mathcal{P}\text{-M-SIG}}$

Refreshment of pwd

1. U starts by sending a request (`refresh`, pwd, pwd_{new}) to T.
2. T computes a challenge $c \leftarrow \mathcal{H}(pwd, pwd_{new}, r)$, and $r \in_{\mathrm{R}} \{0,1\}^{\kappa}$, $s \in_{\mathrm{R}} \{0,1\}^{\ell}$ (ℓ is the length of pwd_{new}) and sends (`ch-pwd`, c), s to M.
3. M sends (`ch-pwd?`) to U, and U returns either $\perp$ or (`OK`) to M.
4. If $\perp$ was returned to M, $\perp$ is forwarded to T, while in case of (`OK`), M calculates a response $\rho \leftarrow F_k(c|1)$, where $(c|1)$ means c concatenated with a 1-bit, and $\alpha = \mathcal{E}_k^S(s)$. It sends ρ, α to T.
5. T sends (`ch-pwd`, $pwd, pwd_{new} \oplus s, \alpha, r, \rho$) to S.
6. S decrypts *alpha* and calculates $pwd_{new} = (pwd_{new} \oplus s) \oplus s$. If S accepts pwd, and if $\rho = F_k(\mathcal{H}(pwd, pwd_{new}, r))$, then S will set $pwd \leftarrow pwd_{new}$.

Fig. 7. Refreshment of the password pwd in $\pi_{\mathcal{P}\text{-M-SIG}}$

5.3 Security of $\pi_{\mathcal{P}\text{-M-SIG}}$

In this section we prove in theorem 2, the security of $\pi_{\mathcal{P}\text{-M-SIG}}$, however, we also comment on, and emphasize some of the security properties of $\pi_{\mathcal{P}\text{-M-SIG}}$, since not all are covered directly by the UC framework and theorem 2. The proof of Theorem 2 is analogous to the one for theorem 1 and can be found in the full version of this paper [10]. The only substantial change is that the reduction showing that the protocol does not allow forgery of signatures, now needs to simulate refreshment phases without knowing the secret key, which turns out to be straightforward. Theorem 2 relies on our earlier assumptions, but also requires a standard assumption of the security of cryptographic hash functions:

Assumption 3. *For any PPT Turing machine A: If A is given $\mathcal{H}$ and $y = \mathcal{H}(x)$, A can only with negligible probability return a x' s.t. $y = \mathcal{H}(x')$.*

Theorem 2 ($\pi_{\mathcal{P}\text{-M-SIG}}$ UC-realizes $\mathcal{F}_{\text{M-SIG}}$). *Under assumptions 1, 2 and 3 and if F_k and $\mathcal{E}^S$ are secure, $\pi_{\mathcal{P}\text{-M-SIG}}$ UC-realizes $\mathcal{F}_{\text{M-SIG}}$ with respect to adaptive adversaries in the proactive model, where the adversary may corrupt at most one of: M, T or S and D. Active corruption is allowed for all players except D, which can be passively corrupted.*

In the model we use here, U stores his backup share of d. It could be argued that this is not realistic and we should extend the model with a separate player D_U modeling whatever back-up storage U is using. This is indeed possible, and our protocol would still be secure in this extended model against an adversary who may do corruptions as usual and in addition may corrupt D_U passively, provided he never corrupts other players in a way that would give him both back-up shares (e.g., corrupting both D and D_U would be forbidden). We have omitted this for simplicity. Note that since we assume U is never corrupted, we could in principle implement the back-up needed for *refreshment** by giving U the complete key d. However, this solution is clearly dubious since it stores the entire secret key in a single location. Indeed, it is clearly insecure if we extend the model with D_U, which is why we prefer $\pi_{\mathcal{P}\text{-M-SIG}}$. Another advantage of $\pi_{\mathcal{P}\text{-M-SIG}}$ is that higher security during *refreshment** can be implemented by using two backup sharings. One sharing when S has been corrupt, but is honest again, and an other sharing when M has been corrupt, but is recovered. This improves security in a case where several adversaries work independently of each other, but each adversary corrupts at most one player. Such a case cannot covered by the standard model, where a single monolithic adversary is always assumed.

Comments on the usage of Passwords. It should be emphasized that the refreshment protocol of *pwd* in Fig. 7 can run without the rest of refreshment. The reason for doing this is if the user suspects a corrupted terminal has been used. To ensure that none other than the current used terminal T can change the password during *refreshment**, where a blank “old” password is used, a list of onetime passwords can be generated during key generation and sent to U and stored by D.

6 Blinding Messages

A well-known technique by Chaum [8] can be used to blind the messages to be signed so that S learns no information whatsoever on what is signed. The idea is that M, when it handles a message m, will choose a random $r \in Z_N^*$ and compute $b = \mathcal{H}(m)r^e \bmod N$, where e is the public RSA exponent. Note that this can be feasible even for a computationally weak M if a small public exponent is used. In the rest of the signature issuing, $\mathcal{H}(m)$ is replaced by b, and the PRF-value needed for outsourcing is computed from b. The blinding factor r is sent to T. Since the signature issuing process is otherwise unchanged, it will eventually allow T to compute $b^d \bmod N = \mathcal{H}(m)^d r \bmod N$, so by dividing out r, T can recover the signature. On the other hand, S has only seen b which reveals nothing about m since r was uniformly chosen.

7 Implementing a Prototype

We have formulated a cryptographic approach to improve the security and mobility of the usage of digital signatures. If these improvements are to reach the actual users in the real world, the cryptographic protocols has to be implemented in a way that makes it attractive for real users to use them. Therefore we are working together with experts in the field of "Human Computer Interaction" on a prototype that implements the described protocols. This is still work in progress, but the signature issuing phase of the protocol has been implemented. The implementation is done in Java using the Bouncy Castle library [16] for cryptographic tasks. We used a Sony Ericsson T850i cellphone to play the role of M while T was a standard PC, speaking to M via Bluetooth, and to S via SSL. We have verified that the signatures produced conform to the standard of the nation-wide PKI OCES[5] already in use in Denmark.

We observed that outsourcing the exponentiation from M to T gives a very significant speedup. With 1024 bit keys, exponentiation on the phone takes around 6 seconds. In a real use case, this makes the system seem heavy and slow and gives the impression that improved security degrades the user friendliness. Running the outsourcing protocol reduces the signature time to a fraction of a second and makes it seem as if the signing happens instantly. Of course, improved speed of exponentiation is possible by using native code instead of Java. But apart from the fact that portability would suffer if we do this, it is not clear that it would be enough: we would probably need to move to 2000 bit RSA in a real system which might cost up to a factor of 8 in performance.

8 Conclusion, Future Work and Acknowledgement

We have proposed a model for personal digital signatures that we believe reflects reality better than previous proposals. We have proposed a protocol for signature

[5] Danish for "Public Certificates for Electronic Services".

generation that remains secure even if the computing equipment used is partially corrupt. Finally we have implemented the essential parts of the protocol and verified that it is practical. The protocol still assumes key generation as a trusted service, and in ongoing work we investigate methods for generating keys using a secure distributed computation, that takes into account the computational weakness of the mobile device.

We thank the anonymous PKC referees and Michael Steiner for comments that helped us improve the paper substantially.

References

1. Almansa, J.F., Damgård, I.B., Nielsen, J.B.: Simplified Threshold RSA with Adaptive and Proactive Security. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 593–611. Springer, Heidelberg (2006)
2. Asokan, N., Baum-Waidner, B., Pedersen, T.P., Pfitzmann, B., Schunter, M., Steiner, M., Waidner, M.: In: Lacoste, G., Pfitzmann, B., Steiner, M., Waidner, M. (eds.) SEMPER 2000. LNCS, vol. 1854, pp. 45–64. Springer, Heidelberg (2000) ISBN 3-540-67825-5
3. Boneh, D.: Twenty years of attacks on the RSA cryptosystem. Notices of the American Mathematical Society (AMS) 46(2) (1999)
4. Boneh, D., Franklin, M.K.: Efficient generation of shared RSA keys. J. ACM 48(4), 702–722 (2001)
5. Buhler, P., Eirich, T., Waidner, M., Steiner, M.: Secure password-based cipher suite for tls. In: NDSS. The Internet Society (2000)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000 (2005 version)
7. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
8. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO, pp. 199–203 (1982)
9. Damgård, I., Koprowski, M.: Practical threshold RSA signatures without a trusted dealer. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 152–165. Springer, Heidelberg (2001)
10. Damgård, I., Mikkelsen, G.: On the theory and practice of personal digital signatures. In: Eprint Archive (2008)
11. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
12. Gennaro, R., Rabin, T., Jarecki, S., Krawczyk, H.: Robust and efficient sharing of RSA functions. J. Cryptology 13(2), 273–300 (2000)
13. Herzberg, A.: Payments and banking with mobile personal devices. CACM 46(5), 53–58 (2003)
14. Itkis, G., Reyzin, L.: Sibir: Signer-base intrusion-resilient signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 499–514. Springer, Heidelberg (2002)
15. Mannan, M.S., van Oorschot, P.C.: Using a personal device to strengthen password authentication from an untrusted computer. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 88–103. Springer, Heidelberg (2007)

16. Legion of the Bouncy Castle. Bouncy castle crypto APIs, `http://www.bouncycastle.org`
17. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks (extended abstract). In: PODC, pp. 51–59 (1991)
18. Parno, B., Kuo, C., Perrig, A.: Phoolproof phishing prevention. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 1–19. Springer, Heidelberg (2006)
19. Rabin, T.: A simplified approach to threshold and proactive RSA. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 89–104. Springer, Heidelberg (1998)
20. Weigold, T., Kramp, T., Hermann, R., Höring, F., Buhler, P., Baentsch, M.: The zurich trusted information channel – an efficient defence against man-in-the-middle and malicious software attacks. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 75–91. Springer, Heidelberg (2008)

Security of Blind Signatures under Aborts

Marc Fischlin and Dominique Schröder

Darmstadt University of Technology, Germany
marc.fischlin@gmail.com, schroeder@me.com,
www.minicrypt.de

Abstract. We explore the security of blind signatures under aborts where the user or the signer may stop the interactive signature issue protocol prematurely. Several works on blind signatures discuss security only in regard of completed executions and usually do not impose strong security requirements in case of aborts. One of the exceptions is the paper of Camenisch, Neven and shelat (Eurocrypt 2007) where the notion of selective-failure blindness has been introduced. Roughly speaking, selective-failure blindness says that blindness should also hold in case the signer is able to learn that some executions have aborted.

Here we augment the work of Camenisch et al. by showing how to turn every secure blind signature scheme into a selective-failure blind signature scheme. Our transformation only requires an additional computation of a commitment and therefore adds only a negligible overhead. We also study the case of multiple executions and notions of selective-failure blindness in this setting. We then discuss the case of user aborts and unforgeability under such aborts. We show that every three-move blind signature scheme remains unforgeable under such user aborts. Together with our transformation for selective-failure blindness we thus obtain an easy solution to ensure security under aborts of either party and which is applicable for example to the schemes of Pointcheval and Stern (Journal of Cryptology, 2000).

We finally revisit the construction of Camenisch et al. for simulatable adaptive oblivious transfer protocols, starting from selective-failure blind signatures where each message only has one valid signature (uniqueness). While our transformation to achieve selective-failure blindness does not preserve uniqueness, it can still be combined with a modified version of their protocol. Hence, we can derive such oblivious transfer protocols based on unique blind signature schemes only (in the random oracle model), without necessarily requiring selective-failure blindness from scratch.

1 Introduction

Blind signatures, proposed by Chaum [5], allow a signer to interactively sign messages for users such that the messages are hidden from the signer. Since their introduction many blind signatures schemes have been proposed [1,3,5,6,9,16,17, 18,21,22], and they typically share two basic security properties: *blindness* says that a malicious signer cannot decide upon the order in which two messages have

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 297–316, 2009.

been signed in two executions with an honest user, and *unforgeability* demands that no adversarial user can create more signatures than interactions with the honest signer took place.

The security requirements for blind signatures have been formalized by Juels et al. [16] and by Pointcheval and Stern [22]. Although these widely used definitions give basic security guarantees, blindness only holds in a restricted sense when it comes to aborted executions. That is, prior work does not guarantee blindness in case the signer is able to learn which of two executions aborted (even if one execution aborts only after the protocol has concluded)". However, in e-cash scenarios an honest user, unable to eventually derive a valid coin, will most likely complain to the malicious bank afterwards.

Recently, Camenisch et al. [7] consider a stronger kind of aborts where a cheating signer may be able to make the user algorithm fail depending on the message being signed,[1] and where the malicious signer is informed afterwards which execution has failed (if any). Considering for example a voting protocol based on blind signatures [7,10], a malicious administrator can potentially deduce information about votes (possibly also for non-aborted executions) by causing some voters to abort and consulting the subsequent complaints.

As for user aborts and unforgeability, albeit the definitions [16] and [22] are identical in spirit, the "one-more" notion in [22] leaves two possible interpretations: either the adversarial user is deemed to generate one more signature than executions with the signer have been *initiated* (i.e., even counting executions in which the user aborts), or the malicious user needs to output one more signature than executions have been *completed* (i.e., allowing user aborts). In fact, this ambiguity re-appears in many works about blind signatures, some explicitly counting initiated executions [3,9,15], some emphatically referring to completed executions [6,16,18,21] and some remaining vague, too [1,7,14].

For both cases, user and signer aborts, the stronger notions are desirable of course. For a blind signature scheme used to sign coins in an e-cash system, for instance, a malicious signer may otherwise abort executions deliberately and, by this, may be able to revoke unlinkability of coins. Vice versa, if unforgeability says that no adversarial user is able to create more signatures than interactions with the signer have been *initiated*, and no requirement about aborted sessions is imposed, then an adversarial user could potentially derive more signatures from such aborted executions. The signing bank could generally charge users for executions, which have stopped early. Yet, if the connection in the signing process breaks down accidently, the honest user is most likely unable to derive the coin and would hence be reluctant to pay for the transaction The bank may then gracefully waive the fee for such aborted executions, but still needs to handle forgery attempts.

Related Work. As mentioned before, Camenisch et al. [7] have already considered the limitations of the standard blindness notion. They have introduced an extension called *selective-failure blindness* in which the a malicious signer

[1] Ultimately, since the malicious signer causes the abort, this can be seen as a more general case of signer aborts.

should not be able to force an honest user to abort the signature issue protocol because of a certain property of the user's message, which would disclose some information about the message to the signer. They present a construction of a simulatable oblivious transfer protocols from so-called unique selective-failure blind signature schemes (in the random oracle model) for which the signature is uniquely determined by the message. Since the main result of the work [7] is the construction of oblivious transfer protocols, the authors note that Chaum's scheme [5] and Boldyreva's protocol [3] are examples of such selective-failure blind schemes, but do not fully explore the relationship to (regular) blindness.

Hazay et al. [15] present a concurrently-secure blind signature scheme and, as part of this, they also introduce a notion called a-posteriori blindness. This notion considers blindness of multiple executions between the signer and the user (as opposed to two sessions as in the basic case), and addresses the question how to deal with executions in which the user cannot derive a signature. However, the definition of a-posteriori blindness is neither known to be implied by ordinary blindness, nor implies it ordinary blindness (as sketched in [15]). Thus, selective-failure blindness does not follow from this notion.

Aborts of players have also been studied under the notion of fairness in two-party and multi-party computations, especially for the exchange of signatures, e.g. [2, 11, 13]. Fairness should guarantee that one party obtains the output of the joint computation if and only if the other party receives it. Note, however, that in case of blind signatures the protocol only provides a one-sided output to the user (namely, the signature). In addition, solutions providing fairness usually require extra assumptions like a trusted third party in case of disputes, or they add a significant overhead to the underlying protocol.

Our Results. We pick up the idea of selective-failure blindness to deal with signer aborts and expand the work of Camenisch et al. [7] towards its relationship to blindness and further constructions of such schemes. We first show that selective-failure blindness is indeed a strictly stronger notion than regular blindness. We also extend the notion of selective-failure blindness to multiple executions, particularly addressing aborts of a subset of executions. We give two possible definitions for the multi-execution case and prove them to be equivalent. We then show that blindness in the basic case of two executions suffices to guarantee security in the case of many sessions and discuss the relation to a-posteriori blindness [15].

Next we present a general transformation which turns every secure blind signature scheme into a selective-failure blind scheme. Our transformation only requires an additional commitment of the message, which the user computes before the actual protocol starts and which the user then uses in the original protocol instead of the message itself.[2] Since the commitment is non-interactive, our transformation inherits important characteristics of the underlying protocol like the number of moves and concurrent security.

[2] This idea has been conjectured by Hazay et al. [15] to also work for a-posteriori blindness. We are not aware of any formal claim or proof in the literature that using a commitment indeed provides security against aborts.

It should be noted, though, that the transformation destroys uniqueness (i.e., that each message has only one valid signature per key pair), as as required by [7] to derive oblivious transfer from such blind signatures. However, we show that our transformation is still applicable if we modify the oblivious transfer protocol of [7] slightly. Hence, we can now easily obtain an adaptive oblivious transfer from any unique blind signature scheme such that the protocol is simulatable in presence of failures. Put differently, we show that selective-failure blindness is not necessary to obtain such oblivious transfer protocols, but uniqueness is sufficient. We note that like the original protocol in [7] this result is in the random oracle model.

We finally study the case of user aborts and show that every three-move blind signature scheme is unforgeable under user aborts. While this is clear for two-move schemes like Chaum's protocol [5] our result shows that this remains true for other schemes like the ones by Pointcheval and Stern [22]. We show that, in general, this does not hold for schemes with four or more moves, assuming the existence of a secure two-move blind signature scheme. It remains open if there is a non-trivial and efficient transformation to take care of user aborts for schemes with more than three moves.[3]

In summary, our transformation to achieve selective-failure blindness, together with the result about user aborts, shows that any scheme with two or three moves can be efficiently turned into one, which is secure under aborts (of either party).

2 Blind Signatures

To define blind signatures formally we introduce the following notation for interactive executions between algorithms $\mathcal{X}$ and $\mathcal{Y}$. By $(a, b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$ we denote the joint execution of $\mathcal{X}$ and $\mathcal{Y}$, where x is the private input of $\mathcal{X}$ and y defines the private input of $\mathcal{Y}$. The private output of $\mathcal{X}$ equals a and the private output of $\mathcal{Y}$ is b. We write $\mathcal{Y}^{\langle \mathcal{X}(x), \cdot \rangle^{\infty}}(y)$ if $\mathcal{Y}$ can invoke an unbounded number of executions of the interactive protocol with $\mathcal{X}$ in arbitrarily interleaved order. Accordingly, $\mathcal{X}^{\langle \cdot, \mathcal{Y}(y_0) \rangle^1, \langle \cdot, \mathcal{Y}(y_1) \rangle^1}(x)$ can invoke arbitrarily ordered executions with $\mathcal{Y}(y_0)$ and $\mathcal{Y}(y_1)$, but interact with each algorithm only once.

Definition 1 (Blind Signature Scheme). *A blind signature scheme consists of a tuple of efficient algorithms* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *where*

Key Generation. $\mathsf{KG}_{\mathsf{BS}}(1^n)$ *for parameter* n *generates a key pair* $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}})$.

Signature Issuing. *The joint execution of algorithm* $\mathcal{S}(sk_{\mathsf{BS}})$ *and algorithm* $\mathcal{U}(pk_{\mathsf{BS}}, m)$ *for message* $m \in \{0,1\}^n$ *generates an output* σ *of the user (and some possibly empty output* λ *for the signer),* $(\lambda, \sigma) \leftarrow \langle \mathcal{S}(sk_{\mathsf{BS}}), \mathcal{U}(pk_{\mathsf{BS}}, m) \rangle$.

Verification. $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m, \sigma)$ *outputs a bit.*

[3] By trivial transformations we refer for instance to a solution which ignores the underlying scheme and simply runs, say, Chaum's protocol.

It is assumed that the scheme is complete, *i.e., for any* $n \in \mathbb{N}$, *any* $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$, *any message* $m \in \{0,1\}^n$ *and any* σ *output by* $\mathcal{U}$ *in the joint execution of* $\mathcal{S}(sk_{\mathsf{BS}})$ *and* $\mathcal{U}(pk_{\mathsf{BS}}, m)$ *we have* $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m, \sigma) = 1$.

Security of blind signature schemes is defined by unforgeability and blindness [16, 22]. An adversary $\mathcal{U}^*$ against unforgeability tries to generate $k+1$ valid message-signatures pairs after at most k completed interactions with the honest signer, where the number of executions is adaptively determined by $\mathcal{U}^*$ during the attack. To identify completed sessions we assume that the honest signer returns a special symbol ok when having sent the final protocol message in order to indicate a completed execution (from its point of view). We remark that this output is "atomically" connected to the final transmission to the user.

The blindness condition says that it should be infeasible for a malicious signer $\mathcal{S}^*$ to decide which of two messages m_0 and m_1 has been signed first in two executions with an honest user $\mathcal{U}$. If one of these executions has returned $\perp$ then the signer is not informed about the other signature either.

Definition 2 (Secure Blind Signature Scheme). *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *is called* secure *if the following holds:*

Unforgeability. *For any efficient algorithm* $\mathcal{U}^*$ *the probability that experiment* $\mathsf{Unforge}_{\mathcal{U}^*}^{\mathsf{BS}}(n)$ *evaluates to* 1 *is negligible (as a function of* n*) where*

```
***Experiment*** $\mathsf{Unforge}_{\mathcal{U}^*}^{\mathsf{BS}}(n)$
    $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$
    $((m_1, \sigma_1), \ldots, (m_{k+1}, \sigma_{k+1})) \leftarrow \mathcal{U}^{*\langle \mathcal{S}(sk_{\mathsf{BS}}), \cdot \rangle^{\infty}}(pk_{\mathsf{BS}})$
    Return 1 iff
        $m_i \neq m_j$ for $1 \leq i < j \leq k+1$, and
        $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m_i, \sigma_i) = 1$ for all $i = 1, 2, \ldots, k+1$, and
        $\mathcal{S}$ has returned $\mathsf{ok}$ in at most $k$ interactions.
```

Blindness. *For any efficient algorithm* $\mathcal{S}^*$ *(working in modes* find, issue *and* guess*) the probability that the following experiment* $\mathsf{Blind}_{\mathcal{S}^*}^{\mathsf{BS}}(n)$ *evaluates to* 1 *is negligibly close to* 1/2, *where*

```
***Experiment*** $\mathsf{Blind}_{\mathcal{S}^*}^{\mathsf{BS}}(n)$
    $(pk_{\mathsf{BS}}, m_0, m_1, \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{B}^*(\mathsf{find}, 1^n)$
    $b \leftarrow \{0,1\}$
    $\mathsf{st}_{\mathsf{issue}} \leftarrow \mathcal{B}^{*\langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_b) \rangle^1, \langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{1-b}) \rangle^1}(\mathsf{issue}, \mathsf{st}_{\mathsf{find}})$
        and let $\sigma_b, \sigma_{1-b}$ denote the (possibly undefined) local outputs
        of $\mathcal{U}(pk_{\mathsf{BS}}, m_b)$ resp. $\mathcal{U}(pk_{\mathsf{BS}}, m_{1-b})$.
    set $(\sigma_0, \sigma_1) = (\perp, \perp)$ if $\sigma_0 = \perp$ or $\sigma_1 = \perp$
    $b^* \leftarrow \mathcal{B}^*(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{st}_{\mathsf{issue}})$
    return 1 iff $b = b^*$.
```

3 Selective-Failure Blindness

In this section we review the definition of selective-failure blindness and show that selective-failure blindness is a strictly stronger requirement than the basic blindness property. Second, we discuss how to extend selective-failure blindness to multiple executions.

3.1 Definition

Camenisch et al. [7] put forward the notion of *selective-failure blindness*, which says that a malicious signer $\mathcal{S}^*$ cannot force the user algorithm $\mathcal{U}$ to abort based on the specific message. This is formalized by informing $\mathcal{S}^*$ which instance has aborted (i.e., if the left, the right, or both user instances have failed):

Definition 3. *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S},\mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *is called* selective-failure blind *if it is unforgeable (as in Definition 2) and the following holds:*

Selective-Failure Blindness. *For any efficient algorithm* $\mathcal{S}^*$ *(which works in modes find, issue and guess) the probability that experiment* $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$ *evaluates to* 1 *is negligibly close to* 1/2 *where*

```
***Experiment*** $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$
    $(pk_{\mathsf{BS}}, m_0, m_1, \beta_{\mathit{find}}) \leftarrow \mathcal{S}^*(\mathit{find}, 1^n)$
    $b \leftarrow \{0,1\}$
    $\beta_{\mathit{issue}} \leftarrow \mathcal{S}^{*\langle \cdot,\mathcal{U}(pk_{\mathsf{BS}},m_b)\rangle^1, \langle \cdot,\mathcal{U}(pk_{\mathsf{BS}},m_{1-b})\rangle^1}(\mathit{issue}, \beta_{\mathit{find}})$
        and let $\sigma_b, \sigma_{1-b}$ denote the (possibly undefined) local outputs
        of $\mathcal{U}(pk_{\mathsf{BS}}, m_b)$ resp. $\mathcal{U}(pk_{\mathsf{BS}}, m_{1-b})$.
    define answer as: left  if only the first execution has failed,
                      right if only the second execution has failed,
                      both  if both executions have failed,
                      and $(\sigma_b, \sigma_{1-b})$ otherwise.
      $b^* \leftarrow \mathcal{S}^*(\mathit{guess}, \mathsf{answer}, \beta_{\mathit{issue}})$
    Return 1 iff $b = b^*$.
```

3.2 Relation to Regular Blindness

We first prove formally the fact that selective-failure blindness implies regular blindness. Then we separate the notion by turning a secure blind signature scheme into a one which is still secure but provably not selective-failure blind.

Proposition 1. *Every selective-failure blind signature scheme* $\mathsf{BS}_{\mathsf{SF}}$ *is also a secure blind signature scheme.*

The claim follows easily and the formal proof is given in the full version.

Proposition 2. *If there exists a secure blind signature scheme* BS*, then there exists a secure blind signature scheme* $\mathsf{BS}_{\overline{\mathsf{SF}}}$ *which is* not *selective-failure blind.*

Proof. We modify BS slightly into a scheme $\mathsf{BS}_{\overline{\mathsf{SF}}}$ which is identical to BS, except that we modify the key generation algorithm and add a break condition into the user algorithm. More precisely, let $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S},\mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ be a secure blind signature scheme. We define the new blind signature scheme $\mathsf{BS}_{\overline{\mathsf{SF}}}$ as

KeyGen. $\mathsf{KG}_{\overline{\mathsf{SF}}}$ first sets $m_{\max} = 1^n$ as the maximum of the lexicographical order over n-bit strings. It then executes the key generation algorithm of the underlying blind signature scheme $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$ and returns $(sk_{\overline{\mathsf{SF}}}, pk_{\overline{\mathsf{SF}}}) = (sk_{\mathsf{BS}}, (pk_{\mathsf{BS}}, m_{\max}))$.

Signing Protocol. The interactive signing protocol remains unchanged except for one modification. The user algorithm checks *after* the last move of the protocol (and after computing the signature σ) that $m \leq m_{\max}$ and, if so, returns the signature σ, and $\perp$ otherwise.

Verification. The verification algorithm returns the result of $\mathsf{Vf}_{\mathsf{BS}}$.

The modified scheme is clearly complete, as the case $m > m_{\max}$ for an honest signer never occurs and because the initial protocol is complete. Obviously, if the blind signature scheme BS is unforgeable, then $\mathsf{BS}_{\overline{\mathsf{SF}}}$ is also unforgeable. This is easy to see as the malicious user may simply ignore the break condition.

Concerning blindness, first note that the malicious signer $\mathcal{S}^*$ is allowed to choose the public key and thus to pick some other value $m^*_{\max}$. As a malicious signer $\mathcal{S}^*$ is not informed which of the executions has failed (if any), setting some other value $m^*_{\max}$ than the predetermined maximum and possibly causing an abort does not lend any additional power to $\mathcal{S}^*$. To see this, note that the user algorithm does not abort prematurely if $m > m_{\max}$. Hence, from the (malicious) signer's point of view, the interaction is indistinguishable from an honest execution. It therefore follows that $\mathsf{BS}_{\overline{\mathsf{SF}}}$ still satisfies blindness.

We finally show that the modified scheme does not fulfill selective-failure blindness. Consider an malicious signer $\mathcal{S}^*$ in experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$. In the first step the adversary $\mathcal{S}^*$ computes a key pair $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$, it sets $m^*_{\max} = 10^{n-1}$ and picks two messages $m_0 = 0^n, m_1 = 1^n$ such that $m_0 \leq m^*_{\max} < m_1$. It outputs a public key $pk_{\overline{\mathsf{SF}}} = (pk_{\mathsf{BS}}, m^*_{\max})$ together with the message m_0, m_1 as defined in the first step of the experiment. Next, $\mathcal{S}^*$ has black-box access to two honest user instances (as described in experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$) where the first algorithm takes as input $(pk_{\overline{\mathsf{SF}}}, m_b)$ and the second user algorithm receives $(pk_{\overline{\mathsf{SF}}}, m_{1-b})$. In both executions $\mathcal{S}^*$ acts like the honest signer with key $sk_{\overline{\mathsf{SF}}} = sk_{\mathsf{BS}}$. Then $\mathcal{S}^*$ is eventually informed which of the executions has failed, i.e., receives left or right (as $\mathcal{S}^*$ has access to honest user instances, the case where both executions fail cannot occur by the completeness condition). The adversary $\mathcal{S}^*$ returns $b^* = 1$ if the left instance has failed, otherwise it returns $b^* = 0$.

It follows straightforwardly that the adversary $\mathcal{S}^*$ succeeds in predicting b with probability 1. □

3.3 Selective-Failure Blindness for Multiple Executions

The presumably natural way to extend selective-failure blindness to an arbitrary number of executions with user instances would be as follows. The malicious signer chooses q messages as well as a public key pk_{BS} and interacts with q user instances. We denote by π be a random permutation over $\{1, 2, \ldots, q\}$. The i-th user instance is initiated with the message $m_{\pi(i)}$ and the public key pk_{BS}. If at least one of the user instances aborts, then the adversary is given a binary vector v of length q indicating which of the user algorithms aborted. In the case that each execution allows the user to create a valid signature, then the adversary is given all message-signature pairs in non-permuted order.

In the final step the adversary tries to link a message-signature pair to an execution. There are two possible venues to formalize this. The first one, which we believe reflects much better the idea that the adversary should not be able to determine the order of signed messages, is to ask the adversary two output two indices i_0, i_1 such that $\pi(i_0) < \pi(i_1)$. The second version would be to ask the adversary to predict the connection much more explicitly, demanding to output indices (i, j) such that $\pi(i) = j$. Note that for the case of two executions both notions are equivalent.

Here we give the "order-based" definition and show in in the full version that the two definitions are equivalent, assuming the following strengthening: During the signature issuing and in the final processing phase we give the malicious signer access to an oracle Reveal which for input i returns $\pi(i)$ and the user's signature σ_i if the execution has already finished successfully. This corresponds to the case that some coins in e-cash systems may have been spent meanwhile. Note that the reveal oracle takes as input a state $\mathsf{st}^{\mathsf{rev}}$ where each signature is stored. The adversary's final choice i_0, i_1 must not have been disclosed, of course.

Definition 4. *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *is called* multi-execution selective-failure blind *if it is unforgeable (as in Definition 2) and the following holds:*

Multi-Execution SF-Blindness. *For any efficient algorithm* $\mathcal{S}^*$ *(working in modes* find, issue, *and* reveal*) the probability that experiment* $\mathsf{MSFBlind}_{\mathcal{S}^*}^{\mathsf{BS}}(n)$ *returns* 1 *is negligibly close to* $\frac{1}{2}$, *where*

```
Experiment $\mathsf{MSFBlind}_{\mathcal{S}^*}^{\mathsf{BS}}(n)$
    $(pk_{\mathsf{BS}}, M, \beta_{\mathit{find}}) \leftarrow \mathcal{S}^*(\mathit{find}, 1^n)$ where $M = (m_1, \ldots, m_q)$ with $m_i \in \{0,1\}^n$
    Select a random permutation $\pi$ over $\{1, 2, \ldots, q\}$
    $\beta_{\mathit{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(1)}) \rangle^1, \ldots, \langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(q)}) \rangle^1, \mathsf{Reveal}(\cdot, \pi, \mathsf{st}^{\mathit{rev}})}(\mathit{issue}, \beta_{\mathit{find}})$
        and let $\sigma_{\pi(1)}, \ldots, \sigma_{\pi(q)}$ denote the (possibly undefined) local outputs
        of $\mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(1)}), \ldots, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(q)})$, immediately stored in $\mathsf{st}^{\mathit{rev}}$
        once an execution finishes ($\mathsf{st}^{\mathit{rev}}$ is initially set to $(\perp, \ldots, \perp)$);
        $\mathsf{Reveal}(\cdot, \pi, \mathsf{st}^{\mathit{rev}})$ is an oracle, which on input $i$ returns $(\pi(i), \mathsf{st}_i^{\mathit{rev}})$.
    Return to $\mathcal{S}^*$ all signatures $v = (\sigma_1, \ldots, \sigma_q)$ iff all executions
        have yielded valid signatures; otherwise return a vector $v \in \{0,1\}^q$
        where the $i$-th entry is 1 if the $i$-th signature is valid, and 0 otherwise.
    $(i_0, i_1) \leftarrow \mathcal{S}^{*, \mathsf{Reveal}(\cdot, \pi, \mathsf{st}^{\mathit{rev}})}(\mathit{reveal}, v, \beta_{\mathit{issue}})$
Return 1 iff $\pi(i_0) < \pi(i_1)$ and $\mathcal{S}^*$ has never queried $\mathsf{Reveal}$ about $i_0, i_1$.
```

The definition of multi-execution selective-failure blindness for the case $q = 2$ covers the standard definition of blindness. An adversary $\mathcal{A}$ breaking blindness can be used to build an adversary $\mathcal{S}^*$ breaking multi-execution selective-failure blindness as follows. The malicious signer $\mathcal{S}^*$ executes $\mathcal{A}$ in a black-box way and follows the blindness experiment until $\mathcal{S}^*$ receives either the signatures σ_0, σ_1 or the vector v. In case these two valid signatures are given to $\mathcal{S}^*$, it forwards both pairs to $\mathcal{A}$ and otherwise it outputs $\perp$. Finally, $\mathcal{S}^*$ outputs the decision bit b' returned by $\mathcal{A}$. The definition of selective-failure blindness is (semantically) identical to the definition of multi-execution selective failure blindness for the case $q = 2$.

Proposition 3. *A blind signature scheme which is selective-failure blind, is also multi-execution selective-failure blind.*

The proof appears in the full version. The idea is that one is able to guess the two challenge values i_0, i_1 chosen by $\mathcal{S}^*$ with sufficiently high probability in advance and one can thus reduce it to the two-execution case.

3.4 Relation to A-Posteriori Blindness

In the following we discuss the relation between selective-failure blindness and a-posteriori blindness [15]. Roughly speaking, a-posteriori blindness advocates that blindness only needs to hold for non-aborted sessions. Hazay et al. formalize this basic idea in an experiment where the adversary first outputs a public key *pk* together with a message distribution $\mathcal{M}$. The malicious signer then concurrently interacts with ℓ honest user instances, where each user instance gets as input the public key *pk* and a message sampled according to $\mathcal{M}$. Afterwards, when the signer has finished all ℓ interactions, it receives ℓ' message-signature pairs in a randomly permuted order, where $1 \leq \ell' \leq \ell$ denotes the number of non-aborted executions. The adversary wins the game if it associates one non-aborted execution to a messages-signature pair. A detailed discussion about a-posteriori blindness in the concurrent setting is given in [15].

From a syntactically point of view there are numerous differences between the definition of selective-failure blindness and a-posteriori blindness. Firstly, the adversary in our security definition picks the messages, whereas in the experiment of a-posteriori blindness it only chooses a message distribution. Secondly, in contrast to a-posteriori blindness, the malicious signer in our case receives the information which of the user instances have aborted. In an e-cash scenario, this corresponds to the case where a user (who may have completed all rounds of the protocol) could not derive a valid coin and informs the signing bank about this problem. Thirdly, we propose two different notions of multi-execution selective-failure blindness. The first definition (Definition 4) is an ordering-based definition where the adversary has to distinguish the order of two different executions. The second definition (see the full version) is a prediction-based definition where the malicious signer has to link an execution to a message-signature pair.

Finally, the attacker in our definitions has access to a reveal oracle that discloses the message used during a specific execution. Such an oracle is also not considered in the definition of a-posteriori blindness. In the real world, this oracle represents side information the signer obtains, e.g., customer A opens up a bank account before customer B. Then customer B cannot withdraw coins before having opened up an account and every meanwhile spent coin has to be from customer A. Note that these side information provide the malicious signer also with information about the non-aborted executions. From a technical point of view, the reveal oracle allows us to prove the equivalence between selective-failure blindness for two executions and for multiple executions, as well as the equivalence of the two types of multi-execution selective-failure blindness definitions.

A natural question is whether the definition of a-posteriori blindness and the definition of multi-execution selective-failure blindness are equivalent for the special case of two executions. To answer this question we briefly recall the counter example of Hazay el al. which shows that a-posteriori blindness *does not* imply regular blindness. This example consists of a scheme that satisfies a-posteriori blindness but that easily violates blindness. In this scheme, the honest user algorithms validates the first received message from the signer. In the case that this message is improper, then it sends the message m to the signer and aborts afterwards. Since a-posteriori blindness only guarantees blindness for non-aborted sessions, this scheme remains a-posteriori blind. However, it follows easily that this scheme is not blind. Hence, a-posteriori blindness cannot be equivalent to selective-failure blindness, because selective-failure blindness does imply regular blindness.

4 From Blindness to Selective-Failure Blindness

In this section we show how to turn every secure blind signature scheme BS into a selective-failure blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$. The high-level idea is to modify BS slightly into $\mathsf{BS}_{\mathsf{SF}}$ by executing BS with a non-interactive commitment com of the message m (instead of the message itself).

Definition 5 (Commitment Scheme). *A (non-interactive) commitment scheme consists of a tuple of efficient algorithms* $\mathcal{C} = (\mathsf{KG}_{\mathsf{com}}, \mathsf{Com}, \mathsf{Vf}_{\mathsf{com}})$ *where*

Key Generation. *Algorithm* $\mathsf{KG}_{\mathsf{com}}(1^n)$ *on input the security parameter outputs a key* pk_{com}.

Commitment Phase. *Algorithm* Com *takes as input* pk_{com} *as well as* $m \in \{0,1\}^n$ *and outputs* $(\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m)$.

Verification. $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{com})$ *outputs a bit.*

It is assumed that the commitment scheme is complete, *i.e., for any* $n \in \mathbb{N}$, *any* $pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$, *for any message* $m \in \{0,1\}^n$ *and any* $(\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m)$ *we have* $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{com}) = 1$.

Security of commitment schemes is defined by secrecy and unambiguity. Secrecy guarantees that the receiver cannot learn the message from the commitment and unambiguity says that the sender cannot change the message anymore once the commitment phase is over. Here we use a slightly different way to define secrecy compared to the literature, but it is easy to see by a hybrid argument that our definition is equivalent:

Definition 6 (Secure Commitment). *A (non-interactive) commitment scheme* $\mathcal{C} = (\mathsf{KG}_{\mathsf{com}}, \mathsf{Com}, \mathsf{Vf}_{\mathsf{com}})$ *is called secure if the following holds:*

Secrecy. *For any efficient algorithm* $\mathsf{R}^*_{\mathsf{real}}$ *(working in modes* find *and* guess*) the probability that experiment* $\mathsf{Secrecy}^{\mathcal{C}}_{\mathsf{R}^*_{\mathsf{real}}}(n)$ *evaluates to* 1 *is negligibly close to* $1/2$*.*

```
Experiment $\mathsf{Secrecy}^{\mathcal{C}}_{\mathsf{R}^*_{\mathsf{real}}}(n)$
    $(m_0, m_1, pk_{\mathsf{com}}, \beta_{\mathit{find}}) \leftarrow \mathsf{R}^*_{\mathsf{real}}(\mathit{find}, 1^n)$
    $b \leftarrow \{0,1\}$
    $\mathsf{com}_b \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_b)$ and $\mathsf{com}_{1-b} \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_{1-b})$
    $b^* \leftarrow \mathsf{R}^*_{\mathsf{real}}(\mathit{guess}, \beta_{\mathit{find}}, \mathsf{com}_0, \mathsf{com}_1)$
    Return 1 iff $b = b^*$.
```

Unambiguity. *For any efficient algorithm* $\mathsf{S}^*_{\mathsf{real}}$ *the probability that experiment* $\mathsf{Unambiguity}^{\mathcal{C}}_{\mathsf{S}^*_{\mathsf{real}}}(n)$ *evaluates to* 1 *is negligible.*

```
Experiment $\mathsf{Unambiguity}^{\mathcal{C}}_{\mathsf{S}^*_{\mathsf{real}}}(n)$
    $pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$
    $(m, m', \mathsf{decom}, \mathsf{decom}') \leftarrow \mathcal{S}^*(pk_{\mathsf{com}})$
    Return 1 iff
        $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{Com}) = 1$ and
        $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m', \mathsf{decom}', \mathsf{Com}) = 1$ as well as $m \neq m'$.
```

Note that such commitment schemes exist under standard assumptions like pseudorandom generators [19] or hash functions [8]. In order to use a commitment in a blind signature scheme —which we defined to take messages of n bits— we need that the commitment scheme is *length-invariant*, meaning that for n-bit messages the commitment itself is also n bits. This can always be achieved by using a collision-resistant hash function (with n bits output) on top.

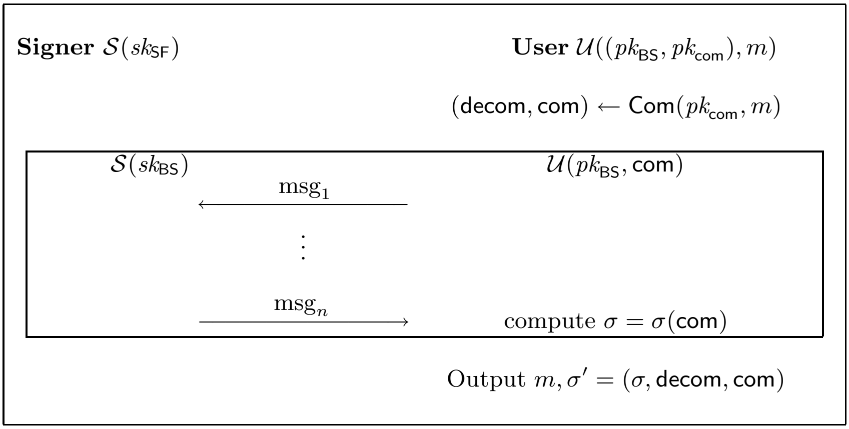

Signer $\mathcal{S}(sk_{\mathsf{SF}})$
User $\mathcal{U}((pk_{\mathsf{BS}}, pk_{\mathsf{com}}), m)$
$(\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m)$
$\mathcal{S}(sk_{\mathsf{BS}})$
$\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com})$
msg_1
$\vdots$
msg_n
compute $\sigma = \sigma(\mathsf{com})$
Output $m, \sigma' = (\sigma, \mathsf{decom}, \mathsf{com})$

Fig. 1. Issue protocol of the blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$

Construction 1 (Selective-Failure Blind Signature Scheme $\mathsf{BS}_{\mathsf{SF}}$). *Let* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle\mathcal{S},\mathcal{U}\rangle, \mathsf{Vf}_{\mathsf{BS}})$ *be a blind signature scheme and let* $\mathcal{C}$ *be a length-invariant commitment scheme. Define the blind signature scheme* $\mathsf{BS}_{\mathsf{SF}}$ *through the following three procedures:*

Key Generation. *The generation algorithm* $\mathsf{KG}_{\mathsf{SF}}(1^n)$ *executes the key generation algorithm of the blind signature scheme* BS, $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$*. It also runs the key generation algorithm for the commitment scheme,* $pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$*. It returns the private key* $sk_{\mathsf{SF}} = sk_{\mathsf{BS}}$ *and the public key* $pk_{\mathsf{SF}} = (pk_{\mathsf{BS}}, pk_{\mathsf{com}})$*.*

Signature Issue Protocol. *The interactive signature issue protocol for message* $m \in \{0,1\}^n$ *is described in Figure 1.*

Signature Verification. *The verification algorithm* $\mathsf{Vf}_{\mathsf{SF}}(pk_{\mathsf{SF}}, m, \sigma')$ *for* $\sigma' = (\sigma, \mathsf{decom}, \mathsf{com})$ *returns* 1 *iff* $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, \sigma, \mathsf{com}) = 1$ *and* $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{com}) = 1$*.*

Theorem 1. *If* BS *is a secure blind signature scheme and* $\mathcal{C}$ *is a secure, length-invariant commitment scheme, then the scheme* $\mathsf{BS}_{\mathsf{SF}}$ *in Construction 1 is a selective-failure blind signature scheme.*

We note that, if the starting blind signature scheme provides statistical blindness, and the commitment scheme is also statistically-hiding, then the derived protocol achieves selective-failure blindness in a statistical sense. This can be seen from the proof of the theorem, which is split into two claims, covering unforgeability and selective-failure blindness:

Claim 1: $\mathsf{BS}_{\mathsf{SF}}$ is unforgeable.

In the proof we distinguish between two cases. The first case occurs if the adversary $\mathcal{U}^*$ succeeds in outputting $k+1$ valid pairs $m_i, \sigma'_i = (\sigma_i, \mathsf{decom}_i, \mathsf{com}_i)$ such that the commitments com_i are pairwise different. But then we can break the unforgeability of the underlying blind signature scheme BS. In the second case $\mathcal{U}^*$ succeeds and at least two commitments $\mathsf{com}_i, \mathsf{com}_j$ (with $i \neq j$) are identical. But then we can break the unambiguity of the commitment scheme $\mathcal{C}$.

Proof. Assume to the contrary that the resulting selective-failure blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$ is *not* unforgeable. Then there exists an adversary $\mathcal{U}^*$ breaking unforgeability with noticeable probability, i.e., on input pk_{SF} the algorithm $\mathcal{U}^*$ returns $k+1$ valid signatures $\sigma'_i = (\sigma_i, \mathsf{decom}_i, \mathsf{com}_i)$ for messages m_i after at most k interactions with the honest signer $\mathcal{S}$. Note that here we do not deal with user aborts and count any initiated interaction; the case of counting only completed interactions is taken care of in the next section.

We first take a look at the success probability of $\mathcal{U}^*$, we have

$$\psi(n) := \mathrm{Prob}\left[\mathsf{Forge}_{\mathcal{U}^*}^{\mathsf{BS}_{\mathsf{SF}}}(n) = 1\right]$$

where $\psi(n)$ is noticeable. This probability can be separated according to the two exclusive events that $\mathcal{U}^*$ succeeds and all commitments com_i are different, with the corresponding probability denoted by $\psi_0(n)$, and into the case where $\mathcal{A}_{\mathsf{SF}}$ succeeds and at least two commitments are identical (with probability $\psi_1(n)$) According to our assumption that $\psi(n)$ is noticeable, $\psi_0(n)$ or $\psi_1(n)$ (or both) must be noticeable.

We next construct out of $\mathcal{U}^*$ algorithms $\mathcal{A}_{\mathsf{UNF}}$ and $\mathcal{A}_{\mathsf{UNA}}$ against unforgeability of BS and unambiguity of the commitment scheme $\mathcal{C}$.

Attacking Unforgeability. The adversary $\mathcal{A}_{\mathsf{UNF}}$ takes as input the public key pk_{BS} of the blind signature scheme BS and works as follows. It executes the key generation algorithm of the commitment scheme $pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$ and runs a black-box simulation of $\mathcal{U}^*$ on input $pk_{\mathsf{SF}} = (pk_{\mathsf{BS}}, pk_{\mathsf{com}})$. The signer instances in the attack of $\mathcal{U}^*$ are simulated with the help of the external signer instances accessible by $\mathcal{A}_{\mathsf{UNF}}$, i.e., adversary $\mathcal{A}_{\mathsf{UNF}}$ relays the communication between $\mathcal{U}^*$ and its signer instance oracle $\mathcal{S}(sk_{\mathsf{BS}})$ (as described in experiment $\mathsf{Forge}_{\mathcal{U}^*}^{\mathsf{BS}}$). When $\mathcal{U}^*$ finishes its attack, it outputs $k+1$ message-signatures pairs m_i, σ_i' after at most k interactions. Now $\mathcal{A}_{\mathsf{UNF}}$ parses each σ_i' as $(\sigma_i, \mathsf{decom}_i, \mathsf{com}_i)$ and returns the $k+1$ pairs com_i, σ_i and stops.

Assume that $\psi_0(n)$, the probability that $\mathcal{U}^*$ succeeds and all com_i's are distinct, is noticeable. Then, since the simulation is perfect from the viewpoint of $\mathcal{U}^*$, adversary $\mathcal{A}_{\mathsf{UNF}}$ succeeds in outputting $k+1$ valid pairs com_i, σ_i for distinct "messages" com_i with noticeable probability, too, contradicting the unforgeability property of the underlying blind signature scheme. Note also that the numbers of initiated and completed executions are identical in both cases.

Attacking Unambiguity. In order to break the unambiguity of $\mathcal{C}$, the adversary $\mathcal{A}_{\mathsf{UNA}}$ takes as input the public key pk_{com} of the commitment scheme $\mathcal{C}$ and works as follows. It executes the key generation algorithm of the blind signature scheme $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$ as well as a the honest signer algorithms $\mathcal{S}(sk_{\mathsf{BS}})$ and runs a black-box simulation of $\mathcal{U}^*$ on input $pk_{\mathsf{SF}} = (pk_{\mathsf{BS}}, pk_{\mathsf{com}})$. Note that running the program of the honest signer on input sk_{BS} simulates each execution with a signer instance. Algorithm $\mathcal{U}^*$ eventually returns $k+1$ message-signature pairs (m_i, σ_i') after at most k interactions with $\mathcal{S}$. The adversary $\mathcal{A}_{\mathsf{UNA}}$ then checks if there are valid signatures with $\mathsf{com}_i = \mathsf{com}_j$ for some $i \neq j$ and, if so, outputs two tuples $(m_i, \mathsf{decom}_i, \mathsf{com}_i), (m_j, \mathsf{decom}_j, \mathsf{com}_j)$ such that $m_i \neq m_j$ and $\mathsf{com}_i = \mathsf{com}_j$. If not, it outputs a failure message.

For the analysis note that the simulation again perfectly mimics the original attack of $\mathcal{U}^*$. Hence, if $\psi_1(n)$ is noticeable, then such $\mathsf{com}_i = \mathsf{com}_j$ with valid decommitments for $m_i \neq m_j$ appear with noticeable probability, and the commitment adversary $\mathcal{A}_{\mathsf{UNA}}$ therefore finds an ambiguous commitment with this probability, too. But this clearly violates the security of the commitment scheme $\mathcal{C}$. □

Claim 2: $\mathsf{BS}_{\mathsf{SF}}$ is selective-failure blind.

The high-level idea of the proof is as follows. We again distinguish between two cases. In the first case the adversary $\mathcal{A}_{\mathsf{SF}}$ succeeds with noticeable probability and both message-signature pairs are valid. But then we show how to break the blindness property of the underlying blind signature scheme BS. We next argue that in the case where $\mathcal{A}_{\mathsf{SF}}$ succeeds with noticeable probability and forces at least one of the user algorithms to fail, then we are able to break the secrecy of the commitment scheme (because then the only information available to the signer are the commitments of the messages).

Proof. Assume towards contradiction that the resulting blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$ is *not* selective-failure blind, and that there exists a successful adversary $\mathcal{A}_{\mathsf{SF}}$ against selective-failure blindness. Let

$$\delta(n) := \mathrm{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1\right] = \tfrac{1}{2} + \epsilon(n)$$

where $\epsilon(n) = \delta(n) - \frac{1}{2}$ is noticeable. We divide the success case according to the two exclusive events that $\mathcal{A}_{\mathsf{SF}}$ succeeds and that both message-signature pairs are valid (event valid) and into the case where $\mathcal{A}_{\mathsf{SF}}$ succeeds and at least one of the signatures is not valid (event $\neg\mathsf{valid}$). Then,

$$\begin{aligned}
&\mathrm{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1\right] - \tfrac{1}{2}\\
&\qquad = \mathrm{Prob}[\mathsf{valid}] \cdot \left(\mathrm{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \,\middle|\, \mathsf{valid}\right] - \tfrac{1}{2}\right)\\
&\qquad\qquad + \mathrm{Prob}[\neg\mathsf{valid}] \cdot \left(\mathrm{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \,\middle|\, \neg\mathsf{valid}\right] - \tfrac{1}{2}\right).
\end{aligned}$$

According to our assumption that $\delta(n)$ is noticeable, either the first term, denoted $\delta_0(n)$, or the second term $\delta_1(n)$ has to be noticeable (or both are noticeable). We next turn $\mathcal{A}_{\mathsf{SF}}$ into algorithms $\mathcal{A}_{\mathsf{blind}}$ and $\mathcal{A}_{\mathsf{com}}$ against regular blindness and secrecy of the commitment scheme, respectively.

Attacking Blindness. The adversary $\mathcal{A}_{\mathsf{blind}}$ works as follows. It runs a black-box simulation of $\mathcal{A}_{\mathsf{SF}}$, which initially outputs two messages (m_0, m_1) together with a public key pk_{SF}. The adversary $\mathcal{A}_{\mathsf{blind}}$ extracts pk_{BS} and pk_{com} from pk_{SF} and calculates the commitments (and decommitments) $(\mathsf{decom}_0, \mathsf{com}_0) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_0)$ and $(\mathsf{decom}_1, \mathsf{com}_1) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_1)$. It outputs $\mathsf{com}_0, \mathsf{com}_1$ and pk_{BS}. It is given access to user instances $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_b)$ and $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_{1-b})$ for a unknown bit b and relays the communication between these instances and $\mathcal{A}_{\mathsf{SF}}$. If, at the end, at least one of the (external) user algorithms fails, then $\mathcal{A}_{\mathsf{blind}}$ outputs a random bit and stops. Otherwise, it augments σ_0, σ_1 to $\sigma'_0 = (\sigma_0, \mathsf{decom}_0, \mathsf{com}_0)$ and $\sigma'_1 = (\sigma_1, \mathsf{decom}_1, \mathsf{com}_1)$ and returns the two signatures σ'_0, σ'_1 (obtained by the external user algorithms) to $\mathcal{A}_{\mathsf{SF}}$. The final output of $\mathcal{A}_{\mathsf{blind}}$ consists of the bit b^* returned by $\mathcal{A}_{\mathsf{SF}}$.

Note that $\mathcal{A}_{\mathsf{blind}}$ simulates the experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n)$ by executing the blindness experiment for the underlying blind signature scheme BS and by computing the commitments internally. Hence, the case where both message-signature pairs are valid is the one where experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n)$ is identical to experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{blind}}}(n)$. If one of the signatures is invalid, then $\mathcal{A}_{\mathsf{blind}}$ returns a random bit. Therefore, the success probability of $\mathcal{A}_{\mathsf{blind}}$ in experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{blind}}}(n)$ can be calculated as:

$$\begin{aligned}
&\mathrm{Prob}\left[\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{blind}}}(n) = 1\right]\\
&\quad = \mathrm{Prob}[b = b^* \wedge \ \neg\mathsf{valid}] + \mathrm{Prob}[b = b^* \wedge \ \mathsf{valid}]\\
&\quad = \mathrm{Prob}[b = b^* \ |\, \mathsf{valid}] \cdot \mathrm{Prob}[\mathsf{valid}] + \mathrm{Prob}[b = b^* \ |\, \neg\mathsf{valid}] \cdot \mathrm{Prob}[\neg\mathsf{valid}] .\\
&\quad = \mathrm{Prob}[\mathsf{valid}] \cdot \mathrm{Prob}[b = b^* \ |\, \mathsf{valid}] + \tfrac{1}{2} \cdot (1 - \mathrm{Prob}[\mathsf{valid}])\\
&\quad = \mathrm{Prob}[\mathsf{valid}] \cdot \mathrm{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \;\middle|\; \mathsf{valid}\right] + \tfrac{1}{2} \cdot (1 - \mathrm{Prob}[\mathsf{valid}])\\
&\quad = \tfrac{1}{2} + \mathrm{Prob}[\mathsf{valid}] \cdot \left(\mathrm{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \;\middle|\; \mathsf{valid}\right] - \tfrac{1}{2}\right)\\
&\quad = \tfrac{1}{2} + \delta_0(n).
\end{aligned}$$

According to our assumption that $\delta_0(n)$ is noticeable it follows that $\mathcal{A}_{\mathsf{blind}}$ breaks the blindness of the underlying blind signature scheme BS with noticeable probability. This, however, contradicts our assumption that BS is a *secure* blind signature scheme.

Attacking Secrecy of the Commitment. In order to break the secrecy of the commitment scheme $\mathcal{C}$, the adversary $\mathcal{A}_{\mathsf{com}}$ executes a black-box simulation of $\mathcal{A}_{\mathsf{SF}}$, which initially outputs two messages (m_0, m_1) as well as a public key pk_{SF}. The adversary $\mathcal{A}_{\mathsf{com}}$ extracts the keys pk_{com} and pk_{BS} from pk_{SF} and outputs $(m_0, m_1, pk_{\mathsf{com}})$ for the secrecy experiment of the commitment scheme. It then receives two commitments $\mathsf{com}_0, \mathsf{com}_1$, one for message m_b and the other one for message m_{1-b} (without knowing which commitment corresponds to which message).

The adversary now runs (in the role of the honest user $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_0)$ and $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_1)$) the selective-failure blindness experiment with $\mathcal{A}_{\mathsf{SF}}$. At the end of the issue protocol each user instance returns either a signature for the commitment or $\perp$. In the case that both user algorithms return a valid signature, then $\mathcal{A}_{\mathsf{com}}$ outputs a random bit b^* and stops. Otherwise, if both user algorithms have failed, then $\mathcal{A}_{\mathsf{com}}$ sends the value both to $\mathcal{A}_{\mathsf{SF}}$. In the case that the first user algorithm has failed, then $\mathcal{A}_{\mathsf{com}}$ returns left to $\mathcal{A}_{\mathsf{SF}}$ and else (if the second user algorithm has failed), it forwards right to $\mathcal{A}_{\mathsf{SF}}$. The final output of $\mathcal{A}_{\mathsf{com}}$ consists of the bit b^* returned by $\mathcal{A}_{\mathsf{SF}}$.

The adversary $\mathcal{A}_{\mathsf{com}}$ simulates the experiment of selective-failure blindness perfectly, up to the point where it obtains the (possibly undefined) signatures. Given that at least one of them is invalid, the simulation corresponds to the case $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n)$ (given $\neg\mathsf{valid}$) for the same choice b as in the commitment experiment. Else, $\mathcal{A}_{\mathsf{com}}$ outputs a random bit. A simple calculation similar to the previous case now shows that

$$\mathrm{Prob}\left[\mathsf{Secrecy}^{\mathcal{C}}_{\mathsf{R}^*_{\mathsf{real}}}(n) = 1\right] = \tfrac{1}{2} + \delta_1(n).$$

If $\delta_1(n)$ is noticeable, it follows that $\mathcal{A}_{\mathsf{com}}$ breaks the secrecy of the commitment scheme with noticeable probability, contradicting the security of $\mathcal{C}$. □

5 Unforgeability and User Aborts

In this section we consider executions in which an adversarial controlled user may abort sessions and the unforgeability requirement with respect to *initiated* or *completed* executions with the signer. For sake of distinction we call the requirement where the adversary has to find $k + 1$ valid message-signature pairs after k initiated executions *weak unforgeability*, and the originally given definition charging only completed executions *unforgeability under user aborts*.

We show in the following that every three-move blind signature scheme, which is weakly unforgeable is also unforgeable under user aborts. Note that in three-move schemes, for a meaningful protocol, the first message is always sent by the signer. As such we may think of two-move schemes as having an additional first move in which the signer simply sends an empty message (although the claim for two-move schemes follows straightforwardly anyway).

We remark that we leave the scheduling of transmissions fully up to the adversary controlling the users, i.e., the adversary decides when to send messages to the signer and when the signer's messages are delivered to the user. Only the signer's output ok is given immediately after the signer's final message has been delivered.

Theorem 2. *Every secure blind signature scheme with at most three moves is unforgeable under user aborts.*

The proof idea is that we can delay the delivery of the user's message in an execution till we can be sure that the adversary completes this execution. If the execution is not completed, then we can simply disregard the original user message, finish the protocol ourselves as an honest user and create another valid signature in addition to the forgeries of the adversary. The full proof appears in the full version.

We note that the result above is optimal in the sense that for four or more moves no such claim can be made (if there are secure schemes with two moves):

Proposition 4. *Every secure blind signature scheme* BS *with two moves can be converted into a secure blind signature scheme* $\mathsf{BS}_{\overline{\mathsf{UuA}}}$ *with four moves, which is weakly unforgeable but* not *unforgeable under user aborts.*

The claim follows by adding two dummy messages at the end, one from the user to signer and one from the signer to the user, such that a malicious user can abort before these dummy messages are exchanged and is still able to derive a signature. The proof appears in the full version.

The previous proposition does not rule out that there is a *transformation* turning schemes with four or more moves into unforgeable ones under user aborts. An apparent approach is to ignore the original protocol and two run a scheme, which already has this property (like Chaum's two-move blind signature scheme in the random oracle model). Yet, it is preferable of course to have a lightweight transformation adhering to the basics of the underlying protocol (like the avoidance of random oracles or general but expensive multi-party protocols).

6 Selective Failures and Adaptive Oblivious Transfer

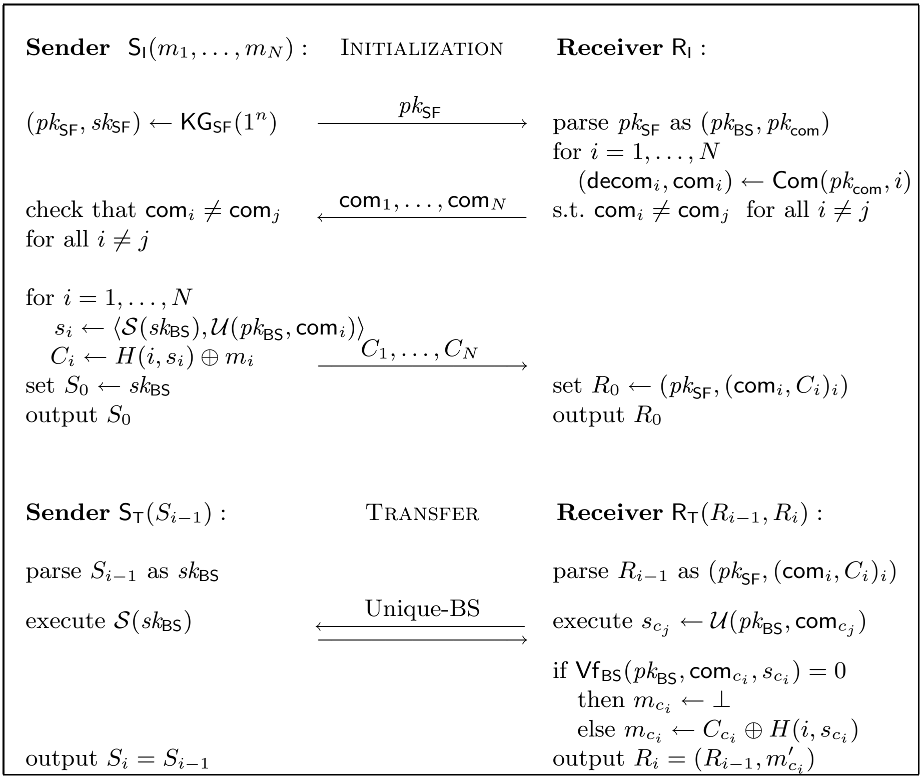
Camenisch et al. [7] also show how to construct an adaptive oblivious transfer protocol out of any unique selective-failure blind signature scheme (in the random oracle model). Roughly speaking, uniqueness means that each message has only one signature per public key. More formally, a blind signature scheme is *unique* [12, 7] if for every (possibly maliciously chosen) public key pk_{BS} and every message $m \in \{0,1\}^*$, there exists at most one signature $s \in \{0,1\}^*$ such that $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m, s) = 1$.

In this section we focus on the question whether our transformation turning every blind signature into one with selective-failure blindness is applicable. We have already mentioned in the introduction that the initial commitment destroys uniqueness of the blind signature scheme because each message may have several valid signatures per key pair. Here we show that is nonetheless possible to build an adaptive k-out-of-N oblivious transfer protocol out of *any* unique blind signature scheme by applying our transformation. The following construction is a modification of the protocol in [7] and, because of the problems with uniqueness, we have to prove the security of this construction from scratch, digging also into the proof of selective-failure blindness for our transformation.

6.1 Simulatable Adaptive Oblivious Transfer

Oblivious Transfer (OT), proposed by Rabin [23], is an interactive protocol between a sender S and a receiver R. The sender in this protocol gets as input N messages $m_1, \ldots, m_N$ and the receiver R wishes to retrieve the message m_c. OT protocols must satisfy the following two security properties: firstly, the sender S does not find out the receiver's choice $c \in \{1, \ldots, N\}$ and, secondly, the receiver only obtains m_c and does not gain any information about the other messages m_i for $i \neq c$. For adaptive k-out-of-N oblivious transfer, $\mathrm{OT}^N_{k\times 1}$, the receiver requests k of these N messages in rounds where the i-th choice is based on the previously obtained messages. We refer the reader to [7, 20] for more information.

As in [7] we consider the real-world/ideal-world paradigm for both sender and receiver security (simulatable oblivious transfer). This paradigm compares the execution of an OT protocol in the real-world with an *ideal implementation* (see for example [4]). In the real-world experiment, both parties jointly execute the interactive protocol, whereas in the ideal-world the functionality is realized through a trusted third party. In our case this means that the sender first hands over the messages to the trusted party, and then the receiver can adaptively obtain messages. To capture failures we let the ideal-world sender in each retrieval also send a bit b, indicating whether the transfer should succeed or abort. We note that this bit is independent of the choice of the receiver, reflecting the fact that the abort should not depend on the receiver's input. Due to the limited space, we review the formal security definition in the full version.



```
Sender $\mathsf{S_I}(m_1,\dots,m_N)$ :                 INITIALIZATION          Receiver $\mathsf{R_I}$ :

$(pk_{\mathsf{SF}}, sk_{\mathsf{SF}}) \leftarrow \mathsf{KG_{SF}}(1^n)$   $\xrightarrow{pk_{\mathsf{SF}}}$   parse $pk_{\mathsf{SF}}$ as $(pk_{\mathsf{BS}}, pk_{\mathsf{com}})$
                                                                            for $i = 1,\dots,N$
                                                                                $(\mathsf{decom}_i, \mathsf{com}_i) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, i)$
check that $\mathsf{com}_i \neq \mathsf{com}_j$   $\xleftarrow{\mathsf{com}_1,\dots,\mathsf{com}_N}$   s.t. $\mathsf{com}_i \neq \mathsf{com}_j$ for all $i \neq j$
for all $i \neq j$

for $i = 1,\dots,N$
    $s_i \leftarrow \langle \mathcal{S}(sk_{\mathsf{BS}}), \mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_i)\rangle$
    $C_i \leftarrow H(i, s_i) \oplus m_i$   $\xrightarrow{C_1,\dots,C_N}$
set $S_0 \leftarrow sk_{\mathsf{BS}}$                                       set $R_0 \leftarrow (pk_{\mathsf{SF}}, (\mathsf{com}_i, C_i)_i)$
output $S_0$                                                                output $R_0$


Sender $\mathsf{S_T}(S_{i-1})$ :                       TRANSFER                Receiver $\mathsf{R_T}(R_{i-1}, R_i)$ :

parse $S_{i-1}$ as $sk_{\mathsf{BS}}$                                        parse $R_{i-1}$ as $(pk_{\mathsf{SF}}, (\mathsf{com}_i, C_i)_i)$

execute $\mathcal{S}(sk_{\mathsf{BS}})$   $\xleftarrow{\text{Unique-BS}}$ $\xrightarrow{}$   execute $s_{c_j} \leftarrow \mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_{c_j})$

                                                                            if $\mathsf{Vf_{BS}}(pk_{\mathsf{BS}}, \mathsf{com}_{c_i}, s_{c_i}) = 0$
                                                                                then $m_{c_i} \leftarrow \perp$
                                                                                else $m_{c_i} \leftarrow C_{c_i} \oplus H(i, s_{c_i})$
output $S_i = S_{i-1}$                                                      output $R_i = (R_{i-1}, m'_{c_i})$
```

Fig. 2. A k-out-of-N oblivious transfer protocol using a random oracle H and any unique blind signature scheme BS

6.2 Construction

Our construction, depicted in Figure 2, is a modification of the $\mathrm{OT}^N_{k\times 1}$ protocol of Camenisch et al. and consists of a black-box construction using *any* unique (not necessarily selective-failure) blind signature scheme. The sender in the first step of the protocol generates a key-pair for the blind signature scheme and sends it to the receiver. The receiver, in return, hands N distinct commitments (for values $1, 2, \dots, N$, represented as n-bit-strings each) over to the sender. These commitments serve as "messages" for the signature generation. Note that distinctiveness of the commitments holds with high probability by the binding property.

After the sender has verified that all commitments are distinct, it encrypts each message in its database by XOR-ing the message m_i with $H(i, s_i)$, where i is the index of the i-th commitment com_i and s_i is the unique signature of message com_i under pk_{BS}. The sender can easily compute this signature locally by running the signature issue protocol with the help of the signing key and an honest user instance for "message" com_i.

After having finished the initialization phase, both parties engage in a transfer phase that consists of a run of the unique blind signature scheme. In the case that the receiver wishes to obtain the i-th message m_i, then it has to choose the commitment com_i (as the message to be signed) during the signature issue protocol.

From a high-level point of view unforgeability guarantees that the receiver cannot receive more messages than interactions took place (sender's security) and blindness guarantees that the sender cannot tell which message has been signed (receiver's security).

Theorem 3. *If the unique blind signature scheme* BS *is unforgeable then the* $OT^N_{k\times 1}$ *scheme depicted in Figure 2 is sender-secure in the random oracle model.*

The proof of this (and the following) theorem appears in the full version.

Theorem 4. *If* BS *is a secure blind signature scheme and* C *is a secure, length-invariant commitment scheme, then the* $OT^N_{k\times 1}$ *scheme depicted in Figure 2 is receiver-secure in the random oracle model.*

Acknowledgments

We thank Heike Busch, Jonathan Katz, and the anonymous reviewers for valuable comments. Both authors are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

References

1. Abe, M.: A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 136–151. Springer, Heidelberg (2001)
2. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
3. Boldyreva, A.: Efficient Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
4. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology 13, 143–202 (2000)
5. Chaum, D.: Blind Signatures for Untraceable Payments. In: Chaum, D. (ed.) Advances in Cryptology — Crypto 1982, pp. 199–203. Plemum, New York (1983)
6. Camenisch, J.L., Koprowski, M., Warinschi, B.: Efficient blind signatures without random oracles. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 134–148. Springer, Heidelberg (2005)
7. Camenisch, J.L., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
8. Damgård, I., Pedersen, T., Pfitzmann, B.: On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. Journal of Cryptology 10(3), 163–194 (1997)

9. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
10. Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
11. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource fairness and composability of cryptographic protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 404–428. Springer, Heidelberg (2006)
12. Goldwasser, S., Ostrovsky, R.: Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 228–245. Springer, Heidelberg (1993)
13. Goldreich, O.: The Foundations of Cryptography, vol. 2. Cambridge University Press, Cambridge (2004)
14. Horvitz, O., Katz, J.: Universally-composable two-party computation in two rounds. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 111–129. Springer, Heidelberg (2007)
15. Hazay, C., Katz, J., Koo, C.-Y., Lindell, Y.: Concurrently-secure blind signatures without random oracles or setup assumptions. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
16. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)
17. Kiayias, A., Zhou, H.-S.: Two-Round Concurrent Blind Signatures without Random Oracles. Number 2005/435 in Cryptology eprint archive (2005), eprint.iacr.org
18. Kiayias, A., Zhou, H.-S.: Equivocal blind signatures and adaptive UC-security. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 340–355. Springer, Heidelberg (2008)
19. Naor, M.: Bit Commitment Using Pseudo-Randomness. Journal of Cryptology 4(2), 151–158 (1991)
20. Naor, M., Pinkas, B.: Computationally Secure Oblivious Transfer. Journal of Cryptology 18(1), 1–35 (2005)
21. Okamoto, T.: Efficient Blind and Partially Blind Signatures Without Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
22. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13(3), 361–396 (2000)
23. Rabin, M.: How to Exchange Secrets by Oblivious Transfer. Technical Report TR-81, Aiken Computation Laboratory (1981)

Security of Sanitizable Signatures Revisited

Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann,
Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk

Darmstadt University of Technology, Germany
www.fischlin.de

Abstract. Sanitizable signature schemes, as defined by Ateniese et al. (ESORICS 2005), allow a signer to partly delegate signing rights to another party, called the sanitizer. That is, the sanitizer is able to modify a predetermined part of the original message such that the integrity and authenticity of the unchanged part is still verifiable. Ateniese et al. identify five security requirements for such schemes (unforgeability, immutability, privacy, transparency and accountability) but do not provide formal specifications for these properties. They also present a scheme that is supposed to satisfy these requirements.

Here we revisit the security requirements for sanitizable signatures and, for the first time, present a comprehensive formal treatment. Besides a full characterization of the requirements we also investigate the relationship of the properties, showing for example that unforgeability follows from accountability. We then provide a full security proof for a modification of the original scheme according to our model.

1 Introduction

Sanitizable signature schemes, introduced by Ateniese et al. [1] and, in a slightly different vein, by Steinfeld et al. [2] and Miyazaki et al. [3], allow a signer to delegate signature rights in a controlled way. Namely, the signer can determine parts of the message which a designated party, the sanitizer, can later modify but such that the authenticity and integrity of the remaining parts is still guaranteed. In particular, even the sanitizer should not be able to change inadmissible parts of the message and produce a valid signature for such illegitimate transformations.

A straightforward application of sanitizable signatures are medical data which should be published in an anonymized but authentic form. Suppose for example that for infectious disease surveillance a hospital is obliged to report excerpts of their patients medical data like dates of birth, genders etc. to an authority. Yet other parts of these data can and should be anonymized, e.g., pseudonyms replacing the patients names or deleting psychiatric information.

Ideally, the administrative department of the hospital assembles the requested information from their records, holding the medical data signed by different health professionals, and sanitizes them without further interaction with their personnel. At the same time the authenticity and integrity of the dedicated data should be preserved. Then, clearly, sanitizable signatures in which the hospital

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 317–336, 2009.

acts as a sanitizer provide a solution. Ateniese et al. [1] provide further applications of sanitizable signature schemes, including multicast, data base outsourcing and secure routing.

Security Requirements. As discussed in [1] meaningful sanitizable signatures come with the usual unforgeability requirement of regular signature schemes:

Unforgeability. It should be infeasible for an outsider (i.e., neither the signer nor the sanitizer) to forge signatures in the name of the signer or the sanitizer.

But the introduction of the sanitizing party and its relationship to the signer entail further desirable security properties. These are:

Immutability. The sanitizer should not be able to produce valid signatures for messages where it has changed other than the designated parts (this can be thought of as an insider attack).

Privacy. Sanitized messages and their signatures should not reveal the original data (i.e., the parts which have been sanitized).

Transparency. It should be infeasible to decide whether a message has been sanitized or not. This may be desirable in applications where one should not be able to discriminate against messages produced by the sanitizer.

Accountability. A party (the signer or the sanitizer) should not be held responsible for messages originating from the other party.

While unforgeability can be formalized straightforwardly from the basic case for regular signatures, as it is done in [1], Ateniese et al. remain rather vague when it comes to the other security requirements. Instead, they introduce technical conditions for the sanitizable signature scheme, aiming to achieve the requirements above. Besides unforgeability these are indistinguishability —roughly saying that signatures generated by the signer are computationally independent of the messages— and the property of identical distributions, saying that the signatures produced by the signer and the sanitizer have identical distributions. This approach is arguable in several ways.

First, without having a formal definition of the security requirements above it is hard to tell if a signature scheme with the technical conditions really achieves the desired goals; as always in cryptography, without a robust security model underneath it is impossible to make precise statements about the hardness of attacks. Secondly, having a more abstract view on the desirable security requirements (instead of the scheme's conditions) facilitates the understanding of their relationships among each other and with other cryptographic primitives. Finally, trying to achieve the security requirements via technical properties seems to be exceedingly restrictive and may exclude otherwise viable solutions.

Our Results. In this paper we revisit the aforementioned security requirements and formalize them according to common game-based approaches. As part of this, we simplify the unforgeability experiment from [1]. We also make several refinements for accountability. First, we augment the model by new algorithms

Proof and Judge where Proof allows to provide evidence to Judge that a message has been sanitized. Then we distinguish between *sanitizer-* and *signer-accountability*, saying that a malicious sanitizer resp. signer cannot falsely accuse the other party. The original approach in [1] only seems to discuss our notion of sanitizer-accountability.

Concerning the relationship of the now-defined security requirements we obtain some useful and also some unexpected results: First, we prove that transparency implies privacy, i.e., any transparent sanitizable signature scheme is also private and for such schemes there is no need to look at the privacy property separately. Secondly, we show that the two accountability types together imply unforgeability, which is in contrast to the position of Ateniese et al. [1] who argue that unforgeability implies accountability. Having a clean model tells us that it is the other way around, and that accountability needs to be considered.

As for the other security properties, immutability, transparency, sanitizer- and signer-accountability we show that each property is independent of the other ones. That is, for each property we present a sanitizable scheme which satisfies all the other requirements except for the one in question. Technically we assume that there are schemes having all properties and then modify the scheme to annihilate the one property. Finally, we show that unforgeability does not follow from sanitizer- or signer-accountability alone (but only if both versions of accountability hold simultaneously). This gives us a complete characterization of the relationship of the notions.

We also revisit the sanitizable signature scheme presented in [1] in light of our formal definitions. We show that a modification of their scheme indeed meets our requirements for immutability, transparency, sanitizer-accountability and signer-accountability. This already implies, via our relationship results, that the scheme is also unforgeable and private and thus a secure sanitizable scheme.

Related Work. As mentioned before, Miyazaki et al. [3] also use the notion of sanitizable signature schemes, but refer to a slightly different approach. According to their notion only deletions of message parts are considered (instead of modifications) and, secondly, the sanitizer is usually not bound to change designated parts of the message but can decide which portions should be deleted. The basic security properties of such sanitizable signature schemes are unforgeability and privacy (following the terminology above). Independently, several similar proposals like content extraction signatures [2] and redactable signatures [4] have been made.

The two approaches for sanitizable signatures and their solutions resemble each other, making the distinction somewhat obscure. This is especially true since further properties have been added to the models in subsequent works, like the requirement that the sanitizer's identity remains hidden [5] in the sanitizable signature model of [3], resembling the above notion of transparency. Nonetheless, one can divide the literature about sanitizable signatures roughly into the works following the approach by Ateniese et al., e.g., [6,7], and the works based on the approach by Miyazaki et al., including [8,9,10,5,11].

We adhere to the notion of sanitizable signature of Ateniese et al. [1], covering message modifications and security requirements like accountability. Some improvements concerning the scheme's efficiency have been made [6] and some extensions concerning multiple, a-posteriori determined censors have been suggested [7]. None of these proposals goes beyond the original approach to model the security properties formally, though. We note that some of the previous works in the vein of Miyazaki et al. [3] come with security models, especially for privacy and unforgeability [2,5,12]. Yet, they often provide limited security guarantees, like privacy requirements holding for a single message-signature pair only. In contrast our models allow more sophisticated attacks where for instance privacy should still hold for multiple message-signature pairs and even if the attacker can ask for further signatures.

Independently of our work, Yuen et al. [13] also revisit the security of sanitizable signatures, but focus on new constructions.

2 Preliminaries

In this section we define sanitizable signatures. Like a regular signature scheme a sanitizable signature scheme allows to sign messages under the secret signer key sk_{sig}, generated together with the public verification key pk_{sig}. The signing process itself includes a public key pk_{san} of a designated sanitizer and a description ADM of division into blocks and admissible blocks which the sanitizer is allowed to change with the help of its secret key sk_{san}. Any such modification takes the original message and signature and some modification information MOD and produces a signature σ' for the modified message m'.

In the sequel we assume for simplicity that the description ADM of admissible blocks defines the block length $t \in \mathbb{N}$ and contains a set of block numbers from $\mathbb{N}$ which can be changed, and that all messages are aligned to block length (say, by standard padding techniques). The modification information MOD is then a list of pairs $(j, m'[j])$ consisting of a block number j and the new content $m'[j]$ for this block. We say that MOD *matches* ADM if all the block numbers in MOD are admissible according to ADM and the length of the blocks in MOD equals the value in ADM. The case of a more general transformation, where the modifications are modeled as arbitrary algorithms, is straightforward and discussed in Appendix A.

In addition, to settle disputes about the origin of a message-signature pair, an algorithm Proof enables the signer to produce a proof π that a signature has been created by the sanitizer. The proof π is generated from a set of previously signed messages. A Judge algorithm then uses the proof π to decide if a *valid* message-signature pair (m, σ) has been created by the signer or the sanitizer (the lack of such a proof is interpreted as a signer origin). We note that Judge is usually only called for valid pairs (m, σ); for invalid pairs settling the dispute is beyond the scheme's scope.

Definition 1 (Sanitizable Signature Scheme). *A sanitizable signature scheme* SanSig *consists of seven efficient algorithms* (KGen_{sig}, KGen_{san}, Sign, Sanit, Verify, Proof, Judge) *such that:*

Key Generation. *There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a pair of keys, a private key and the corresponding public key:*

$$(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^n), \qquad (pk_{san}, sk_{san}) \leftarrow \mathsf{KGen}_{san}(1^n)$$

Signing. *The* Sign *algorithm takes as input a message* $m \in \{0,1\}^*$*, the secret key* sk_{sig} *of the signer, the public key* pk_{san} *of the sanitizer as well as a description* $\textsc{adm} \in \mathbb{N} \times 2^{\mathbb{N}}$ *of the block length* t *and admissibly modifiable message blocks from* $\{0,1\}^t$*. It outputs a signature (or* $\perp$*, indicating an error):*

$$\sigma \leftarrow \mathsf{Sign}(m, sk_{sig}, pk_{san}, \textsc{adm}).$$

We assume that $\textsc{adm}$ *is recoverable from any signature* $\sigma \neq \perp$*.*

Sanitizing. *Algorithm* Sanit *takes a message* $m \in \{0,1\}^*$*, a signature* σ*, the public key* pk_{sig} *of the signer and the secret key* sk_{san} *of the sanitizer. It modifies the message* m *according to the modification instruction* $\textsc{mod} \subseteq \mathbb{N} \times \{0,1\}^t$ *(where* t *is the block length described in* $\textsc{adm}$*) and determines a new signature* σ' *for the modified message* m'*. Then* Sanit *outputs* m' *and* σ' *(or possibly* $\perp$ *in case of an error).*

$$(m', \sigma') \leftarrow \mathsf{Sanit}(m, \textsc{mod}, \sigma, pk_{sig}, sk_{san})$$

Verification. *The* Verify *algorithm outputs a bit* $d \in \{\mathtt{true}, \mathtt{false}\}$ *verifying the correctness of a signature* σ *for a message* m *with respect to the public keys* pk_{sig} *and* pk_{san}*.*

$$d \leftarrow \mathsf{Verify}(m, \sigma, pk_{sig}, pk_{san})$$

Proof. *The* Proof *algorithm takes as input the secret signing key* sk_{sig}*, a message* m *and a signature* σ *as well a set of (polynomially many) additional message-signature pairs* $(m_i, \sigma_i)_{i=1,2,\ldots,q}$ *and the public key* pk_{san}*. It outputs a string* $\pi \in \{0,1\}^*$*:*

$$\pi \leftarrow \mathsf{Proof}(sk_{sig}, m, \sigma, (m_1, \sigma_1), \ldots, (m_q, \sigma_q), pk_{san})$$

Judge. *Algorithm* Judge *takes as input a message* m *and a valid signature* σ*, the public keys of the parties and a proof* π*. It outputs a decision* $d \in \{\mathtt{Sig}, \mathtt{San}\}$ *indicating whether the message-signature pair has been created by the signer or the sanitizer:*

$$d \leftarrow \mathsf{Judge}(m, \sigma, pk_{sig}, pk_{san}, \pi)$$

For a sanitizable signature scheme the usual correctness properties should hold, saying that genuinely signed or sanitized messages are accepted and that a genuinely created proof by the signer leads the judge to decide in favor of the signer.

Signing Correctness. For any security parameter $n \in \mathbb{N}$, any key pair $(sk_{\mathrm{sig}}, pk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^n)$, any key pair $(sk_{\mathrm{san}}, pk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^n)$, any message $m \in \{0,1\}^*$, any $\textsc{adm} \in \mathbb{N} \times 2^{\mathbb{N}}$ and any $\sigma \leftarrow \mathsf{Sign}(m, sk_{\mathrm{sig}}, pk_{\mathrm{san}}, \textsc{adm})$ we have

$$\mathsf{Verify}(m, \sigma, pk_{\mathrm{sig}}, pk_{\mathrm{san}}) = \mathtt{true}.$$

Sanitizing Correctness. For any security parameter $n \in \mathbb{N}$, any key pair $(sk_{\text{sig}}, pk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^n)$, any key pair $(sk_{\text{san}}, pk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^n)$, any message $m \in \{0,1\}^*$, any σ with $\mathsf{Verify}(m, \sigma, pk_{\text{sig}}, pk_{\text{san}}) = \texttt{true}$, any $\textsc{mod} \subseteq \mathbb{N} \times \{0,1\}^t$ matching $\textsc{adm}$ from σ, and any pair $(m', \sigma') \leftarrow \mathsf{Sanit}(m, \textsc{mod}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$ we require

$$\mathsf{Verify}(m', \sigma', pk_{\text{sig}}, pk_{\text{san}}) = \texttt{true}.$$

Proof Correctness. For any security parameter $n \in \mathbb{N}$, any key pair $(sk_{\text{sig}}, pk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^n)$, any key pair $(sk_{\text{san}}, pk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^n)$, any message $m \in \{0,1\}^*$, any signature σ, any $\textsc{mod}$ matching $\textsc{adm}$ from σ, any $(m', \sigma') \leftarrow \mathsf{Sanit}(m, \textsc{mod}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$ with $\mathsf{Verify}(m', \sigma', pk_{\text{sig}}, pk_{\text{san}}) = \texttt{true}$, and any (polynomially many) $m_1, \ldots, m_q$ and $\textsc{adm}_1, \ldots, \textsc{adm}_q$ with $\sigma_i \leftarrow \mathsf{Sign}(m_i, sk_{\text{sig}}, pk_{\text{san}}, \textsc{adm}_i)$ and $(m, \sigma) = (m_i, \sigma_i)$ for some i, any $\pi \leftarrow \mathsf{Proof}(sk_{\text{sig}}, m', \sigma', m_1, \sigma_1, \ldots, m_q, \sigma_q, pk_{\text{san}})$ we require:

$$\mathsf{Judge}(m', \sigma', pk_{\text{sig}}, pk_{\text{san}}, \pi) = \texttt{San}.$$

3 Security Requirements

According to Ateniese et al. [1] there are several security requirements that a secure sanitizable signature needs to satisfy. Informally, these are:

Unforgeability. No outsider should be able to forge the signer's or the censor's signature. This is analogously to the standard security requirement for signatures.

Immutability. The censor is allowed to modify predefined, admissible parts of a message, but he should not be able to modify other parts of the message. For example, a sanitizer who is in charge of blackening names in medical documents should not be able to modify the actual medical data.

Privacy. Nobody should be able to restore sanitized parts of a message. For example, if we have pseudonyms in medical documents then, of course, the original names should not be recoverable.

Transparency. The idea of sanitizable signatures is that, within well-defined limits, the sanitizer inherits the signing authority. Sometimes knowledge of this fact makes the sanitized data less valuable, e.g., an original business plan coming from the CEO is a more desirable target for a spy than the sanitized plan from the administration office. Transparency now says that no one except for the signer and the sanitizer should be able to distinguish signatures from the signer and the sanitizer.

Accountability. If the signer and the censor have an argument about the origin of a valid message-signature pair (m, σ), then accountability demands that this dispute can be settled correctly by the Judge. As an example consider a public servant acting as a sanitizer, but publishing unauthorized information in the name of the government.

We next define these notions formally. We note that we call a sanitizable scheme *secure* if it is simultaneously immutable, unforgeable, private, transparent, sanitizer-accountable and signer-accountable according to the definitions below.

We note that our definitions usually consider three parties, the signer, the sanitizer and the adversary (for some properties the adversary takes over the role of one of the other two parties). In practice, though, one usually has many parties, e.g., a sanitizer assigned to many signers. Our definitions are robust in this regard as we leave much power to the adversary and its queries, say, asking the honest signer to sign a message for a chosen public sanitizer key and thus for different sanitizers. By this, our models can be easily mapped to the case of multiple parties by standard guessing strategies (i.e., trying to predict the "target" signer-sanitizer pair and simulating the other honest parties). As an example we show that our notion of immutability also provides security against the "additional sanitizing attack" of [9], a typical non-malleability attack for three parties.

3.1 Unforgeability

The unforgeability notion for sanitizable signatures follows the classical notion for regular signature schemes. It says that nobody should be able to compute a tupel (m^*, σ^*) such that $\mathsf{Verify}(m^*, \sigma^*, pk_{\mathrm{sig}}, pk_{\mathrm{san}}) = \mathtt{true}$ without having the secret keys $sk_{\mathrm{sig}}, sk_{\mathrm{san}}$. This must hold even if one can see additional signatures for other messages. We also give the adversary access to a Proof box (as proofs could potentially leak information about the secret signing key). Yet, except for this secret key the adversary fully determines the other input data, including the message-signature pairs and the public keys. This allows to capture for example scenarios where several sanitizers are assigned to the same signer.

Definition 2 (Unforgeability). *A sanitizable signature scheme* SanSig *is* unforgeable *if for any efficient algorithm* $\mathcal{A}$ *the probability that the following experiment returns* 1 *is negligible (as a function of* n*):*

```
Experiment $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathsf{SanSig}}(n)$
    $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^n)$
    $(pk_{san}, sk_{san}) \leftarrow \mathsf{KGen}_{san}(1^n)$
    $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{sig}, \cdot, \cdot), \mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{san}), \mathsf{Proof}(sk_{sig}, \dots)}(pk_{sig}, pk_{san})$
        letting $(m_i, \mathrm{ADM}_i, pk_{san,i})$ and $\sigma_i$ for $i = 1, 2, \dots, q$
        denote the queries and answers to and from oracle Sign,
        and $(m_j, \mathrm{MOD}_j, \sigma_j, pk_{sig,j})$ and $(m'_j, \sigma'_j)$ for $j = q+1, \dots, r$
        denote the queries and answers to and from oracle Sanit.
    return 1 if
        $\mathsf{Verify}(m^*, \sigma^*, pk_{sig}, pk_{san}) = \mathtt{true}$ and
        for all $i = 1, 2, \dots, q$ we have $(pk_{san}, m^*) \neq (pk_{san,i}, m_i)$ and
        for all $j = q+1, \dots, r$ we have $(pk_{sig}, m^*) \neq (pk_{sig,j}, m'_j)$.
```

3.2 Immutability

The censor can use the Sanit algorithm to change message blocks which the signer declared as modifiable. If a malicious censor tries to modify other blocks this should not yield a correct signature. In the attack model below the malicious sanitizer $\mathcal{A}$ interacts with the signer to receive signatures σ_i for messages m_i, descriptions $\textsc{adm}_i$ and keys $pk_{\text{san},i}$ of its choice, before eventually outputting a valid pair $(pk^*_{\text{san}}, m^*, \sigma^*)$ such that message m^* is not a "legitimate" transformation of one of the m_i's under the same key $pk^*_{\text{san}} = pk_{\text{san},i}$. The latter is formalized by demanding that each m_i and m^* differ in at least one inadmissible block (or that $pk^*_{\text{san}} \neq pk_{\text{san},i}$).

Definition 3 (Immutability). *A sanitizable signature scheme* SanSig *is* immutable *if for any efficient algorithm $\mathcal{A}$ the probability that the following experiment returns* 1 *is negligible (as a function of n):*

```
Experiment $\mathsf{Immutability}^{\mathsf{SanSig}}_{\mathcal{A}}(n)$
    $(pk_{sig}, sk_{sig}) \leftarrow \mathit{KGen}_{sig}(1^n)$
    $(pk^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathit{Sign}(\cdot, sk_{sig}, \cdot), \mathit{Proof}(sk_{sig}, \ldots, pk_{sig}, \cdot)}(pk_{sig})$
        letting $(m_i, \textsc{adm}_i, pk_{san,i})$ and $\sigma_i$ for $i = 1, 2, \ldots, q$
        denote the queries and answers to and from oracle $\mathit{Sign}$.
    return 1 if
        $\mathit{Verify}(m^*, \sigma^*, pk_{sig}, pk^*_{san}) = \mathtt{true}$ and
        for all $i = 1, 2, \ldots, q$ we have
            $pk^*_{san} \neq pk_{san,i}$, or
            $m^*[j_i] \neq m_i[j_i]$ for some $j_i \notin \textsc{adm}_i$
            //where shorter messages are padded with blocks of the special symbol $\perp \notin \{0,1\}^*$
```

Thwarting Additional Sanitizing Attacks. Testifying to the fact that our definition is quite robust in the multi-party setting we discuss that our notion of immutability implies the "additional sanitizing attack" of Miyazaki et al. [9]. Suppose we have three parties in a department, the signer and two sanitizers. Both sanitizers are authorized in principle to modify messages, but for a specific message m only the first sanitizer is permitted to do so (say that this message contains information affecting the second sanitizer). Assume now that a requesting party asks for the non-sensitive parts of message m, and that the first sanitizer with public key pk_{san} is honest and changes the message m to derive a new signature σ' for m'. But now the second sanitizer with public key pk^*_{san} intercepts this reply, maliciously deletes the information about him in message m' and produces a signature σ^* for this bowdlerized message m^*. Only this pair m^*, σ^* is sent to the requesting party, looking like an authorized reply to the requesting party.

Our notion of immutability is strong enough to capture "additional sanitizing attacks" (assuming unique public keys of parties). Namely, in our definition we declare the adversary successful if it manages to find a new public key pk^*_{san} different from the sanitizer's public key pk_{san} such that the final output verifies correctly under this new key pk^*_{san}. An adversary can now mount the additional sanitizing attack by generating the keys of the honest sanitizer internally (in a sense, giving even more control to the adversary), calling the signer to create the document for the key pk_{san} of the honest sanitizer and then outputting the further censored message m^* with σ^* under a public key pk^*_{san}. Hence, immutability guarantees that such a case cannot succeed and, in particular, that the scheme is secure against "additional sanitizing attacks".

3.3 Privacy

Privacy roughly means that it should be infeasible to recover information about the sanitized parts of the message. As information leakage through the modified message itself can never be prevented, we only refer to information which is available through the sanitized signature. There are two possible flavors in formalizing privacy for sanitizable signatures. One approach follows semantic security of encryption schemes and is called *semantic privacy*. It says that for any adversary $\mathcal{A}$ seeing sanitized signatures there is a simulator $\mathcal{S}$ which is denied the signatures, but which is still as successful in predicting some information about the original message as $\mathcal{A}$. This notion is discussed comprehensively in the full version of the paper.

The other approach is based on the indistinguishability notion for encryption. In this case, an adversary can choose pairs (m_0, MOD_0), (m_1, MOD_1) of messages and modifications together with a description ADM and has access to a "left-or-right" box. This oracle either returns a sanitized signature for the left tuple ($b = 0$) or for the right tuple ($b = 1$). The task of the attacker is to predict the random bit b significantly better than by guessing. Here we need the additional constraint that for each call to the left-or-right box the resulting modified messages are identical for both tuples and the modifications both match ADM, else the task would be trivial. We write $(m_0, \text{MOD}_0, \text{ADM}) \equiv (m_1, \text{MOD}_1, \text{ADM})$ for this.

Below we formalize the more handy indistinguishability notion and discuss in the full paper that the simulation-based approach is equivalent (as in case of encryption). In our definition of privacy we grant the adversary also access to a signature and a sanitizer oracle, enabling the adversary to create signatures which can be sanitized afterwards. We note that the adversary does not get to choose the signature $\sigma_{j,b}$ for inputs to the left-or-right box. Instead, this signature is first computed from scratch. This corresponds to the "hospital setting" mentioned in the introduction, where the medical data and, in particular, their signatures are kept confidentially and only the sanitized document is released. One may define a stronger version where the adversary gets to choose $\sigma_{j,0}, \sigma_{j,1}$, but it seems much harder to realize this requirement efficiently.

As in case of unforgeability and immutability we also grant the adversary access to Proof. Hence, since we let the adversary also determine the input to this box the adversary may input the data received from the Sign box here, but cannot use any of the initially computed and secret signatures in the calls to the left-or-right box (unless the adversary accidently guesses one). The reason is again that proofs usually leak information about the signatures but the signatures in the left-or-right box should remain secret (as in the hospital example).

Definition 4 (Privacy). *A sanitizable signature scheme* SanSig *is* private *if for any efficient algorithm $\mathcal{A}$ the probability that the following experiment returns* 1 *is negligibly close to $\frac{1}{2}$:*

```
*Experiment* $\mathsf{Privacy}_{\mathcal{A}}^{\mathsf{SanSig}}(n)$
    $(pk_{sig}, sk_{sig}) \leftarrow \mathit{KGen}_{sig}(1^n)$
    $(pk_{san}, sk_{san}) \leftarrow \mathit{KGen}_{san}(1^n)$
    $b \leftarrow \{0,1\}$
    $a \leftarrow \mathcal{A}^{\mathit{Sign}(\cdot,sk_{sig},\cdot,\cdot),\mathit{Sanit}(\cdot,\cdot,sk_{san},\cdot),\mathit{Proof}(sk_{sig},\cdots),\mathit{LoRSanit}(\cdot,\cdot,\cdot,sk_{sig},sk_{san},b)}(pk_{sig}, pk_{san})$
        where oracle $\mathit{LoRSanit}(\cdot,\cdot,\cdot,sk_{sig},sk_{san},b)$
        on input $(m_{j,0}, \text{MOD}_{j,0},(m_{j,1}, \text{MOD}_{j,1})$ and $\text{ADM}_j$
        first computes $\sigma_{j,b} \leftarrow \mathit{Sign}(m_{j,b}, sk_{sig}, pk_{san}, \text{ADM}_j)$ and then
        returns $(m'_j, \sigma'_j) \leftarrow \mathit{Sanit}(m_{j,b}, \text{MOD}_{j,b}, \sigma_{j,b}, pk_{sig}, sk_{san})$,
        and where $(m_{j,0}, \text{MOD}_{j,0}, \text{ADM}_j) \equiv (m_{j,1}, \text{MOD}_{j,1}, \text{ADM}_j)$,
        i.e., are mapped to the same modified message.
    return 1 if $a = b$.
```

3.4 Transparency

For transparency the original work of Ateniese et al. [1] distinguishes between two notions, called weak and strong transparency. In the case of weak transparency an adversary, given a signed message m with a valid signature σ, should not be able to correctly guess whether m has been sanitized or was simply signed. In the case of strong transparency, the adversary should not even be able to tell which parts of the message are potentially mutable. Since the latter seems an overly strong requirement —observe that this implies that the information ADM must be hidden and must not be recoverable from σ, for example— we call weak transparency simply transparency here and formalize only this notion.

We define transparency by the following adversarial game. We consider an adversary $\mathcal{A}$ with access to Sign, Sanit and Proof oracles with which the adversary can create signatures for (sanitized) messages and learn proofs. In addition, $\mathcal{A}$ gets access to a Sanit/Sign box which contains a secret random bit $b \in \{0,1\}$ and which, on input a message m, a modification information MOD and a description ADM

– for $b = 0$ runs the signer to create $\sigma \leftarrow \mathsf{Sign}(m, sk_{\text{sig}}, pk_{\text{sig}}, \text{ADM})$, then runs the sanitizer and returns the sanitized message m' with the new signature σ', and
– for $b = 1$ acts as in the case $b = 0$ but also signs m' from scratch with the signing algorithm to create a signature σ' and returns the pair (m', σ').

Adversary $\mathcal{A}$ eventually produces an output a, the guess for b. A sanitizable signature is now said to be *transparent* if for all efficient algorithms $\mathcal{A}$ the probability for a right guess $a = b$ in the above game is negligibly close to $\frac{1}{2}$.

Definition 5 (Transparency). *A sanitizable signature scheme* SanSig *is* transparent *if for any efficient algorithm $\mathcal{A}$ the probability that the following experiment returns* 1 *is negligibly close to $\frac{1}{2}$:*

```
***Experiment*** $\mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SanSig}}(n)$
    $(pk_{sig}, sk_{sig}) \leftarrow \mathit{KGen}_{sig}(1^n)$
    $(pk_{san}, sk_{san}) \leftarrow \mathit{KGen}_{san}(1^n)$
    $b \leftarrow \{0, 1\}$
    $a \leftarrow \mathcal{A}^{\mathit{Sign}(\cdot,sk_{sig},\cdot,\cdot),\mathit{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{san}),\mathit{Proof}(sk_{sig},\ldots),\mathit{Sanit/Sign}(\cdot,\cdot,\cdot,sk_{sig},sk_{san},pk_{sig},pk_{san},b)}$
        with input $(pk_{sig}, pk_{san})$
        where oracle $\mathit{Sanit/Sign}$ for input $m_k, \mathrm{MOD}_k, \mathrm{ADM}_k$
        computes $\sigma_k \leftarrow \mathit{Sign}(m_k, sk_{sig}, pk_{san}, \mathrm{ADM}_k)$
        then $(m'_k, \sigma'_k) \leftarrow \mathit{Sanit}(m_k, \mathrm{MOD}_k, \sigma_k, pk_{sig}, sk_{san})$,
        then, if $b = 1$, replaces $\sigma'_k$ by $\sigma'_k \leftarrow \mathit{Sign}(m'_k, sk_{sig}, pk_{san}, \mathrm{ADM}_k)$,
        and finally returns $(m'_k, \sigma'_k)$.
    return 1 if $a = b$
```

We note that, analogously to the case of privacy, we have σ_k be created by the signer locally in the Sanit/Sign box. A stronger requirement would enable the adversary to determine this signature as part of the input. Yet, this notion again does not reflect the "hospital scenario" nor does it seem to be easy to realize efficiently. Similarly, the adversary cannot use these signatures in the Proof box.

Also note that, with the definition above, schemes with deterministic signature or sanitizing algorithms cannot be transparent, because an adversary could then easily compare answers from the Sanit/Sign box with outputs of the signature sanitizing oracle. Yet, since some applications may need transparency even if a message has been signed or sanitized before, we provide the stronger requirement. The weaker guarantee would then also demand from the adversary's queries to the signing and sanitizing boxes that for all k we have $m'_k \neq m_i$ for all i and $m'_k \neq m'_j$ for all j.

3.5 Accountability

Accountability says that the origin of a (sanitized) signature should be undeniable. There are two types of accountability:

Sanitizer-Accountability. If a message has not been signed by the signer, then even a malicious sanitizer should not be able to make the judge accuse the signer.

Signer Accountability. If a message and its signature have not been sanitized, then even a malicious signer should not be able to make the judge accuse the sanitizer.

Both notions are formalized below through two similar, yet slightly different adversarial games.

In the sanitizer-accountability game let $\mathcal{A}_{\mathsf{Sanit}}$ be an efficient adversary playing the role of the malicious sanitizer. Adversary $\mathcal{A}_{\mathsf{Sanit}}$ has access to a Sign oracle and a Proof oracle. Its task is to output a valid message-signature pair m^*, σ^* together with a key pk^*_{san} (with (pk^*_{san}, m^*) being different from messages previously signed by the Sign oracle) such that the proof produced by the signer via Proof still leads Judge to decided "Sig", i.e., that the signature has been created by the signer.

Definition 6 (Sanitizer-Accountability). *A sanitizable signature scheme* SanSig *is* sanitizer-accountable *if for any efficient algorithm $\mathcal{A}_{\mathsf{Sanit}}$ the probability that the following experiment returns* 1 *is negligible (as a function of n):*

```
Experiment San-Accountability$^{\mathsf{SanSig}}_{\mathcal{A}_{Sanit}}(n)$
    $(pk_{sig}, sk_{sig}) \leftarrow KGen_{sig}(1^n)$
    $(pk^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{Sign(\cdot, sk_{sig}, \cdot, \cdot), Proof(sk_{sig}, \dots)}_{Sanit}(pk_{sig})$
        letting $(m_i, \text{ADM}_i, pk_{san,i})$ and $\sigma_i$ for $i = 1, 2, \dots, q$
        denote the queries and answers to and from oracle Sign
    $\pi \leftarrow Proof(sk_{sig}, m^*, \sigma^*, (m_1, \sigma_1), \dots, (m_q, \sigma_q), pk_{san})$
    return 1 if
        $(pk^*_{san}, m^*) \neq (pk_{san,i}, m_i)$ for all $i = 1, 2, \dots, q$, and
        $Verify(m^*, \sigma^*, pk_{sig}, pk^*_{san}) = \mathtt{true}$, and
        $Judge(m^*, \sigma^*, pk_{sig}, pk^*_{san}, \pi) = \mathtt{Sig}$
```

In the signer-accountability game a malicious signer $\mathcal{A}_{\mathsf{Sign}}$ gets a public sanitizing key pk_{san} as input. It is allowed to query a sanitizing oracle about tuples $(m_i, \text{MOD}_i, \sigma_i, pk_{\text{sig},i})$ receiving answers (m'_i, σ'_i). Adversary $\mathcal{A}_{\mathsf{Sign}}$ finally outputs a tuple $(pk^*_{\text{sig}}, \pi^*, m^*, \sigma^*)$ and is considered to succeed if Judge accuses the sanitizer for the new key-message pair pk^*_{sig}, m^* with a valid signature σ^*. Note that our model allows the proof π to contain information about the original message.

Definition 7 (Signer-Accountability). *A sanitizable signature scheme* SanSig *is* signer-accountable *if for any efficient algorithm $\mathcal{A}_{\mathsf{Sign}}$ the probability that the following experiment returns* 1 *is negligible (as a function of n):*

```
Experiment Sig-Accountability$^{\mathsf{SanSig}}_{\mathcal{A}_{Sign}}(n)$
    $(pk_{san}, sk_{san}) \leftarrow KGen_{san}(1^n)$
    $(pk^*_{sig}, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{Sanit(\cdot, \cdot, \cdot, \cdot, sk_{san})}_{Sign}(pk_{san})$
        letting $(m'_i, \sigma'_i)$ for $i = 1, 2, \dots, q$
        denote the answers from oracle Sanit.
    return 1 if
        $(pk^*_{sig}, m^*) \neq (pk_{sig,i}, m'_i)$ for all $i = 1, 2, \dots, q$, and
        $Verify(m^*, \sigma^*, pk^*_{sig}, pk_{san}) = \mathtt{true}$ and
        $Judge(m^*, \sigma^*, pk^*_{sig}, pk_{san}, \pi^*) = \mathtt{San}$
```

4 Relationships of the Security Requirements

In this section we show that except for the privacy and the unforgeability requirement all other notions are independent (in the sense that none of them follows

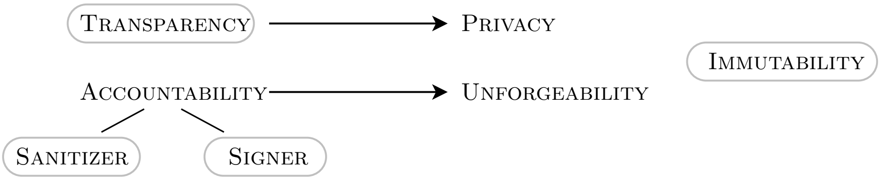


Fig. 1. Summary of the relations among the security properties of sanitizable signatures. Arrows represent implications, frames represent the independence from other requirements.

from the other properties, even if they all hold at the same time). We first show that privacy follows from transparency alone, and unforgeability holds if the two versions of accountability hold simultaneously. We then show the independence of the other requirements.

We stress again that our results are in contrast to the claim by Ateniese et al. [1] that, for example, accountability follows from the unforgeability requirement. Our results show that unforgeability follows from accountability whereas the other direction is not true. It is not clear if Ateniese et al. [1] consider signer-accountability at all, or merely refer to sanitizer-accountability. However, as we have argued both versions of accountability are desirable to avoid framing attacks from either side, and in either case we also show that sanitizer-accountability alone does not imply unforgeability.

Implications. We show that privacy follows from transparency. The idea is that for a transparent scheme one cannot distinguish between signatures created by the signer and ones produced by the sanitizer. Hence, we can essentially replace the left-or-right sanitizing oracle in the privacy experiment by the procedure which creates the signatures for the sanitized message with the help of the signer algorithm. But since the privacy experiment requires the sanitized messages to be identical, the answer is always a fresh signature for the same message, independent of the left-or-right question, and privacy follows.

As mentioned above, unforgeability is implied by the two versions of accountability. The idea behind the result is that, given a successful forgery, the judge cannot really decide if this forgery has been produced by the signer or the sanitizer. Else the judge was biased towards outputting `Sig` or `San` for indecisive cases too often, contradicting either the sanitizer- or signer-accountability.

Seperations. We further show that all the other security requirements are independent, i.e., no property follows from a combination of the other properties. Our results all assume that there exist secure sanitizable signature scheme obeying all properties (which, according to the next section, exist under common cryptographic assumptions) and then show that there is a scheme inheriting all properties except for the one in question.

The proofs of the stated implications and separations appear in the full version. We note that this gives a full characterization of the security requirements.

5 Sanitizable Signatures Based on Chameleon Hashes

In this section we show that our security requirements can be met. Our construction is a modification of the scheme by Ateniese et al. [1] and also uses chameleon hashes. The idea is as follows: Instead of signing the full message in clear we first replace modifiable message blocks $m[i]$ by (randomized) hash values $h[i] = \mathsf{CHash}(pk_{\text{san}}, m[i]; r[i])$ of the blocks. Then we sign this sequence of message blocks and hash values with a regular signature scheme.

The hash values have the special "chameleon" property that, if one has the sanitizer's trapdoor information sk_{san} and $r[i]$, one can easily find collisions, i.e., for given $m'[i]$ one is able to determine $r'[i]$ with $h[i] = \mathsf{CHash}(pk_{\text{san}}, m'[i]; r'[i])$, leaving the hash value invariant. This allows the sanitizer to modify message blocks for which the signer includes the $r[i]$'s in the signature (and only those), and such that the actual signature on the hash values does not need to be modified. We note that implementing the idea is more complicated due to the accountability problem, requiring something related to (but not exactly like) key-exposure freeness [14] from the chameleon hash. The latter also necessitates the usage of tags entering the hash computations.

5.1 Construction

A chameleon hash scheme $\mathcal{CH} = (\mathsf{CHKGen}, \mathsf{CHash}, \mathsf{CHAdapt})$ (with tags) consists of three efficient algorithms such that algorithm CHKGen on input 1^n returns a key pair (sk, pk), algorithm CHash on input pk, a tag $\textsc{Tag} \in \{0,1\}^n$, a message m and randomness r (which is efficiently samplable from some range $\mathcal{R}_{pk}$) returns a hash value $h = \mathsf{CHash}(pk, \textsc{Tag}, m; r)$ and algorithm $\mathsf{CHAdapt}$ on input $sk, \textsc{Tag}, m, r$ and $\textsc{Tag}', m'$ returns r' such that $\mathsf{CHash}(pk, \textsc{Tag}, m; r) = \mathsf{CHash}(pk, \textsc{Tag}', m'; r')$. It also holds that for any $pk, \textsc{Tag}, m, \textsc{Tag}', m'$ the distribution of $\mathsf{CHAdapt}(sk, \textsc{Tag}, m, r, \textsc{Tag}', m')$ (over the choice of r) is the same as the distribution of r itself, also implying that a hash value $\mathsf{CHash}(pk, \textsc{Tag}, m; r)$ (over the choice of r) is distributed independently of $\textsc{Tag}, m$.

Key-exposure freeness [14] now says that it is infeasible to find collisions, even if one gets to see collisions for other values. To be more precise, the security requirement demands that, after having learned collisions for some tags, one cannot create a collision for a new tag. This is a strong and useful notion and, yet, it would not be sufficient to provide security in our setting. Suppose we attach tags to the documents such that the signer modifies messages by finding collisions for the hash value for the corresponding tags. Then a malicious signer could still try to escape accountability by finding further collisions *for the same tag.* We therefore introduce the notion of collision-resistance *under random-tagging attacks*, i.e., where collisions for different tags are created but where one of the two tags is chosen at random (and the other one is provided by the adversary). In the full version we show that such chameleon hashes exist under the RSA assumption in the random oracle model:

Definition 8 (Collision-Resistance under Random-Tagging Attacks). *A chameleon hash scheme* $\mathcal{CH} = (\mathsf{CHKGen}, \mathsf{CHash}, \mathsf{CHAdapt})$ *is* collision-resistant under random-tagging attacks *if for any efficient adversary* $\mathcal{A}$ *the following experiment returns* 1 *with negligible probability only:*

```
Experiment $\mathsf{RndTag}_{\mathcal{A}}^{\mathcal{CH}}(n)$
    $(pk, sk) \leftarrow \mathsf{CHKGen}(1^n)$
    $(\textsc{Tag}, m, r, \textsc{Tag}', m', r') \leftarrow \mathcal{A}^{\mathsf{OAdapt}(sk,\cdot,\cdot,\cdot,\cdot)}(pk)$
        where oracle $\mathsf{OAdapt}$ for the $i$-th query $(\textsc{Tag}_i, m_i, r_i, m_i')$
        with $\textsc{Tag}_i \in \{0,1\}^n$ samples $\textsc{Tag}_i' \leftarrow \{0,1\}^n$ and
        computes $r_i' \leftarrow \mathsf{CHAdapt}(sk, \textsc{Tag}, m, r, \textsc{Tag}', m')$.
        Return $(\textsc{Tag}_i', r_i')$.
    return 1 if
        $(\textsc{Tag}, m) \neq (\textsc{Tag}', m')$ and
        $\mathsf{CHash}(pk, \textsc{Tag}, m; r) = \mathsf{CHash}(pk, \textsc{Tag}', m'; r')$ and
        $\{(\textsc{Tag}, m), (\textsc{Tag}', m')\} \neq \{(\textsc{Tag}_i, m_i), (\textsc{Tag}_i', m_i')\}$ for $i = 1, 2, \ldots$ and
        $\{(\textsc{Tag}, m), (\textsc{Tag}', m')\} \neq \{(\textsc{Tag}_i', m_i'), (\textsc{Tag}_j', m_j')\}$ for $i, j = 1, 2, \ldots$.
```

The condition $\{(\textsc{Tag}, m), (\textsc{Tag}', m')\} \neq \{(\textsc{Tag}_i, m_i), (\textsc{Tag}_i', m_i')\}$ rules out trivial duplication attacks in which the adversary simply copies the data from the interaction with the oracle. The other condition $\{(\textsc{Tag}, m), (\textsc{Tag}', m')\} \neq \{(\textsc{Tag}_i', m_i'), (\textsc{Tag}_j', m_j')\}$ prevents trivial "transitivity" attacks where the adversary calls the oracle about the same $(\textsc{Tag}_i, m_i, r_i)$ twice, but with different m_i', m_j'. Then the oracle's answers collide, as they yield the same value $\mathsf{CHash}(pk, \textsc{Tag}_i, m_i; r_i)$ individually.

In our construction we also need that the tags generated by the signer and the ones by the sanitizer look identical (from the outside) but are generated differently (and that this is provable to a judge). Otherwise a malicious signer would be able to claim that a sanitized message has been the original. We resolve this by letting the tags of the sanitizer be truly random, whereas the tags of the signer need to be created pseudorandomly (with a pseudorandom generator PRG mapping n-bit inputs to $2n$-bit outputs). In addition, the seed for the pseudorandomly generated labels should be recoverable for the signer from the signature and the secret key, such that we use a pseudorandom function PRF (mapping n-bit inputs to n-bit outputs for n-bit keys) to derive the seed for PRG from a nonce $\textsc{Nonce}$, included in the signature.

Finally, we also need a regular signature scheme $\mathcal{S} = (\mathsf{SKGen}, \mathsf{SSign}, \mathsf{SVf})$ being existentially unforgeable under adaptive chosen-message attacks. Below we let $(a_1, a_2, \ldots)$ be some encoding of bit strings $a_1, a_2, \ldots$ into $\{0,1\}^*$ such that (in contrast to concatenation $a_1||a_2||\ldots$) all individual components are recoverable:

Construction 1 (Sanitizable Signature Scheme). *Define the following sanitizable signature scheme* $\mathsf{SanSig} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$*:*

Key Generation. *Algorithm* KGen_{sig} *on input* 1^n *generates a key pair* $(pk, sk) \leftarrow \mathsf{SKGen}(1^n)$ *of the underlying signature scheme, picks a key* $\kappa \leftarrow \{0,1\}^n$

for the pseudorandom function and returns $(pk_{sig}, sk_{sig}) = (pk, (sk, \kappa))$. *Algorithm* KGen_{san} *for input* 1^n *returns a pair* $(pk_{san}, sk_{san}) \leftarrow \mathsf{CHKGen}(1^n)$ *of the chameleon hash scheme.*

Signing. *Algorithm* Sign *on input* $m \in \{0,1\}^{t\ell}, sk_{sig}, pk_{san}, \textsc{adm}$ *picks* $\textsc{Nonce} \leftarrow \{0,1\}^n$ *at random, computes* $x = \mathsf{PRF}(\kappa, \textsc{Nonce})$ *and* $\textsc{Tag} = \mathsf{PRG}(x)$, *and picks* $r[j]$ *for each* j *in* $\textsc{adm}$ *at random. It computes*

$$h[j] = \begin{cases} \mathsf{CHash}(pk_{san}, \textsc{Tag}, (j, m[j], pk_{sig}); r[j]) & \textit{if } j \textit{ is in } \textsc{adm} \\ m[j] & \textit{else} \end{cases}$$

for each block $m[j] \in \{0,1\}^t$ *and* $\sigma_0 \leftarrow \mathsf{SSign}(sk_{sig}, (h, pk_{san}, \textsc{adm}))$ *for* $h = (h[1], h[2], \ldots, h[\ell])$. *It returns* $\sigma = (\sigma_0, \textsc{Tag}, \textsc{Nonce}, \textsc{adm}, r[j_1], \ldots, r[j_k])$ *where each* j_i *is in* $\textsc{adm}$.

Sanitizing. *Algorithm* Sanit *on input a message* m, *information* $\textsc{mod}$, *a signature* $\sigma = (\sigma_0, \textsc{Tag}, \textsc{Nonce}, \textsc{adm}, r[j_1], \ldots, r[j_k])$, pk_{sig} *and* sk_{san} *first checks that each modification in* $\textsc{mod}$ *is admissible according to* $\textsc{adm}$ *and that* σ_0 *is a valid signature for* $(h, pk_{san}, \textsc{adm})$. *If not, it stops with output* $\perp$. *Else, for each* j *in* $\textsc{adm}$ *it lets* $m'[j]$ *be the modified block of* $m[j]$ *(possibly* $m'[j] = m[j]$*), picks new values* $\textsc{Nonce}' \leftarrow \{0,1\}^n$ *and* $\textsc{Tag}' \leftarrow \{0,1\}^{2n}$ *and replaces each* $r[j]$ *in the signature by*

$$r'[j] \leftarrow \mathsf{CHAdapt}(sk_{san}, \textsc{Tag}, (j, m[j], pk_{sig}), r[j], \textsc{Tag}', (j, m'[j], pk_{sig})).$$

It outputs m' *and* $\sigma' = (\sigma_0, \textsc{Tag}', \textsc{Nonce}', \textsc{adm}, r'[j_1], \ldots, r'[j_k])$.

Verification. *Algorithm* Verify *on input a message* $m \in \{0,1\}^{t\ell}$ *and a signature* $\sigma = (\sigma_0, \textsc{Tag}, \textsc{Nonce}, \textsc{adm}, r[i_1], \ldots, r[i_k])$, pk_{sig} *and* pk_{san} *computes*

$$h[j] = \begin{cases} \mathsf{CHash}(pk_{san}, \textsc{Tag}, (j, m[j], pk_{sig}); r[j]) & \textit{if } j \textit{ is in } \textsc{adm} \\ m[j] & \textit{else} \end{cases}$$

and then outputs $\mathsf{SVf}(pk_{san}, (h, pk_{san}, \textsc{adm}), \sigma_0)$ *for* $h = (h[1], \ldots, h[\ell])$.

Proof. *Algorithm* Proof *on input* sk_{sig}, m, σ *and a sequence* (m_i, σ_i) *as well as* pk_{san} *searches the sequence to find a tuple* $(\textsc{Tag}_i, (j, m_i[j], pk_{sig}), r[j])$ *such that*

$$\mathsf{CHash}(pk_{san}, \textsc{Tag}_i, (j, m_i[j], pk_{sig}), r_i[j])$$

$$= \mathsf{CHash}(pk_{san}, \textsc{Tag}, (j, m[j], pk_{sig}), r[j])$$

for some distinct pair $(\textsc{Tag}, (j, m[j], pk_{sig}))$ *in* m, σ *and where* $\textsc{Tag}_i = \mathsf{PRG}(x_i)$ *for* $x_i = \mathsf{PRF}(\kappa, \textsc{Nonce}_i)$ *for the value* $\textsc{Nonce}_i$ *in* σ_i. *If it finds such data it returns this colliding tuple together with* x_i, *i.e.,*

$$\pi = (\textsc{Tag}_i, (j, m_i[j], pk_{sig}), r_i[j], x_i),$$

else it outputs $\perp$.

Judge. *The judge on input* $m, \sigma, pk_{sig}, pk_{san}$ *and* $\pi = (\textsc{Tag}_\pi, (j, m_\pi[j], pk_{sig,\pi}), r_\pi[j], x_\pi)$ *checks that* $pk_{sig} = pk_{sig,\pi}$*, that* π *describes a non-trivial collision under* $\mathsf{CHash}(pk_{san}, \cdot, \cdot, \cdot)$ *for the pair* $(\textsc{Tag}, j, m[j], pk_{sig}), r[j])$ *in* σ*, i.e.,*

$$\mathsf{CHash}(pk_{san}, \textsc{Tag}_\pi, (j, m_\pi[j], pk_{sig,\pi}); r_\pi[j])$$

$$= \mathsf{CHash}(pk_{san}, \textsc{Tag}, (j, m[j], pk_{sig}); r[j]),$$

that the block j *is admissible, and that* $\textsc{Tag}_\pi = \mathsf{PRG}(x_\pi)$ *for the given value* x_π *in the proof. If so, it outputs* `San`*, else it returns* `Sig`*.*

Completeness of signatures generated by the signer follows easily from the completeness of the underlying signature scheme, completeness of signatures generated by the sanitizer follows from the fact that algorithm $\mathsf{CHAdapt}$ always returns a collision, and completeness for proofs holds as one always finds convincing data then.

5.2 Security

It remains to prove security:

Theorem 2. *The sanitizable signature scheme in Construction 1 is secure, i.e., it is immutable, transparent, sanitizer- and signer-accountable (and thus private and unforgeable), assuming that the chameleon hash function is collision-resistant under random-tagging attacks, that* PRG *and* PRF *are pseudorandom and that the signature scheme is existentially unforgeable under adaptive chosen-message attacks.*

Proof. We stepwise go through the properties. Most times we merely outline the security proof because a formalization is straightforward.

Immutability. Assume that the scheme is not immutable according to our definition and that there exists a successful adversary $\mathcal{A}$ against this property. We show that this contradicts the unforgeability of the underlying signature scheme. There are two cases: Assume that $\mathcal{A}$ succeeds by outputting $(pk^*_{\text{san}}, m^*, \sigma^*)$ such that $(pk^*_{\text{san}}, \textsc{adm}^*, h^*)$ is different from all other data $(pk_{\text{san},i}, \textsc{adm}_i, h_i)$ appearing in the attack. Then the valid signature σ^*_0 included in σ^* is for a message $(h^*, pk^*_{\text{san}}, \textsc{adm}^*)$ which has not been signed with the underlying signature scheme before. This, however, contradicts the unforgeability of this signature scheme (observing that we can simulate Proof perfectly without knowledge of the secret key of the signature scheme).

Next assume $(pk^*_{\text{san}}, \textsc{adm}^*, h^*)$ is identical to some $(pk_{\text{san},i}, \textsc{adm}_i, h_i)$. Then, since $pk^*_{\text{san}} = pk_{\text{san},i}$ the messages m^* and m_i must differ in at least one inadmissible block j_i according to $\textsc{adm}_i$. But since $\textsc{adm}^* = \textsc{adm}_i$ this must also be an inadmissible block according to $\textsc{adm}^*$ in m^*. Therefore $h^*[j_i] = m^*[j_i]$ must be different from $h_i[j_i] = m_i[j_i]$, contradicting the fact $h^* = h_i$. Hence, the second case cannot occur and the scheme is immutable.

Transparency. Transparency holds because with overwhelming probability all values $\textsc{Nonce}$ picked by the signer are distinct and thus all x-values are computationally indistinguishable from independent and randomly chosen values. In this case all the generator's outputs, too, are indistinguishable from random $2n$-bit strings (as chosen by the sanitizer). Given this the claim now follows from the distributional property of $\mathsf{CHAdapt}$, that the sanitizing process goes through all admissible block and updates them, and the fact that the distribution of the input $(h, pk_{\text{san}}, \textsc{adm})$ to the signing step is independent of the message. Hence, the distribution of the reply is computationally indistinguishable in the two cases for the $\mathsf{Sanit/Sign}$ box, independently of further queries to the signature, sanitizing or proof oracles (using the fact that the guessing the $\textsc{Nonce}$ values in the signatures computed internally in the $\mathsf{Sanit/Sign}$ box is infeasible).

Sanitizer-Accountability. Assume that the scheme was not sanitizer-accountable and there was a successful adversary $\mathcal{A}$, i.e., such that Proof algorithm cannot find a non-trivial collision in the chameleon hashes for $(pk^*_{\text{san}}, m^*, \sigma^*)$ and the $(pk_{\text{san},i}, m_i, \sigma_i)$ queries. First note that if $(h^*, pk^*_{\text{san}}, \textsc{adm}^*) \neq (h_i, pk_{\text{san},i}, \textsc{adm}_i)$ for all i, the valid signature σ^*_0 in σ^* for this tuple would constitute a successful forgery against the signature scheme (using again the fact that Proof can be easily simulated without the secret signing key).

Hence, there must be some i with $(h^*, pk^*_{\text{san}}, \textsc{adm}^*) = (h_i, pk_{\text{san},i}, \textsc{adm}_i)$. In particular, since a success requires $(pk^*_{\text{san}}, m^*) \neq (pk_{\text{san},i}, m_i)$ we must have $m^*[j] \neq m_i[j]$ for some block j. Furthermore, because $\textsc{adm}^* = \textsc{adm}_i$ and inadmissible message blocks are output in clear and cannot be distinct, it holds that

$$\begin{aligned} h^*[j] &= \mathsf{CHash}(pk^*_{\text{san}}, \textsc{Tag}^*, (j, m^*[j], pk_{\text{sig}}); r^*[j]) \\ &= \mathsf{CHash}(pk^*_{\text{san}}, \textsc{Tag}_i, (j, m_i[j], pk_{\text{sig}}); r_i[j]) = h_i[j] \end{aligned}$$

for some $r^*[j]$ in σ^* and $r_i[j]$ in σ_i. This, however, implies that Proof finds such a non-trivial collision with overwhelming probability. Given this, it is clear that Proof can also output x_i from the genuine signature data.

Signer-Accountability. We finally show signer-accountability, this time using the security under random-tagging attacks of of the chameleon hash function. Assume that there is a successful attacker making the Judge accuse the sanitizer for a message which has not been sanitized by the legal sanitizer.

First note that for the adversary's successful output $pk^*_{\text{sig}}, m^*, \sigma^*$ (with tag $\textsc{Tag}^*$) and $\pi^* = (\textsc{Tag}_\pi, (j, m_\pi[j], pk_{\text{sig}\pi}), r_\pi[j], x_\pi)$ with overwhelming probability $\textsc{Tag}_\pi \neq \textsc{Tag}'_i$ for all i. This is so because with overwhelming probability no $\textsc{Tag}'_i$ lies in the range of PRG and there cannot be a valid preimage x_π for $\textsc{Tag}_\pi = \textsc{Tag}'_i$. In particular, it follows that $\{\textsc{Tag}^*, \textsc{Tag}_\pi\} \neq \{\textsc{Tag}'_i, \textsc{Tag}'_j\}$ for all i, j.

Assume that $\{\textsc{Tag}^*, \textsc{Tag}_\pi\} \neq \{\textsc{Tag}_i, \textsc{Tag}'_i\}$ for all $i = 1, 2, \ldots, q$. Then, because we also have $\{\textsc{Tag}^*, \textsc{Tag}_\pi\} \neq \{\textsc{Tag}'_i, \textsc{Tag}'_j\}$ this would straightforwardly contradict the security of the chameleon hash (noting that we can easily simulate the sanitizer algorithm with the help of the OAdapt oracle). Hence, assume that $\{\textsc{Tag}^*, \textsc{Tag}_\pi\} = \{\textsc{Tag}_i, \textsc{Tag}'_i\}$ for some i and, since the random tags picked by the honest sanitizer are unique with overwhelming probability, we can assume that i is unique.

Because $\textsc{Tag}_\pi \neq \textsc{Tag}'_i$ we must have $\textsc{Tag}^* = \textsc{Tag}'_i$ and $\textsc{Tag}_\pi = \textsc{Tag}_i$. Since $(pk^*_{\mathrm{sig}}, m^*) \neq (pk_{\mathrm{sig},i}, m'_i)$ for a success there must be some j with $(\textsc{Tag}^*, (j, m^*[j], pk^*_{\mathrm{sig}})) \neq (\textsc{Tag}'_i, (j, m'_i[j], pk_{\mathrm{sig},i}))$. However, assuming that all sanitizer tags are unique and observing that with overwhelming probability $\textsc{Tag}'_i \neq \textsc{Tag}_i$ and that for the same tag the prepended block numbers are distinct, it follows that the adversary has generated a new collision $(\textsc{Tag}^*, (j, m^*[j], pk^*_{\mathrm{sig}}))$, $(\textsc{Tag}'_i, (j, m_\pi[j], pk^*_{\mathrm{sig}}))$ which has not been queried previously. This would again contradict the security of the chameleon hash function and signer-accountability follows. □

Acknowledgments

We thank the anonymous reviewers and the crypto group at Bristol for valuable comments. Marc Fischlin, Anja Lehmann and Dominique Schröder are supported by the Emmy Noether Programme Fi 940/2-1 of the German Research Foundation (DFG).

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
3. Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H.: Digital documents sanitizing problem. In: Technical Report ISEC2003-20, IEICE (2003)
4. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
5. Miyazaki, K., Hanaoka, G., Imai, H.: Invisibly sanitizable digital signature scheme. IEICE Transactions 91-A(1), 392–402 (2008)
6. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
7. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor sanitizable signatures and their application to content protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008)
8. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: Piats: A partially sanitizable signature scheme. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 72–83. Springer, Heidelberg (2005)

9. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions 88-A(1), 239–246 (2005)
10. Izu, T., Kunihiro, N., Ohta, K., Takenaka, M., Yoshioka, T.: A sanitizable signature scheme with aggregation. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 51–64. Springer, Heidelberg (2007)
11. Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: ASIACCS, pp. 353–362. ACM Press, New York (2008)
12. Suzuki, M., Isshiki, T., Tanaka, K.: Sanitizable signature with secret information. In: Proceedings of the Symposium on Cryptography and Information Security (2006)
13. Yuen, T.H., Susilo, W., Liu, J.K., Mu, Y.: Sanitizable signatures revisited. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 80–97. Springer, Heidelberg (2008)
14. Ateniese, G., de Medeiros, B.: On the key exposure problem in chameleon hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005)

A General Message Modifications

In this section we outline how to adapt our security notions for more general message modifications. To this end we assume that $\textsc{adm}$ and $\textsc{mod}$ are (descriptions of) efficient algorithms such that $\textsc{adm}(\textsc{mod}) \in \{0, 1\}$ indicates if the modification is admissible and matches $\textsc{adm}$, i.e., $\textsc{adm}(\textsc{mod}) = 1$. The function $\textsc{mod}$ maps any message m to the modified message $m' = \textsc{mod}(m)$.

The notion of unforgeability remains unchanged. For immutability we demand as before that the adversary's output $(pk^*_{\text{san}}, m^*, \sigma^*)$ describes a valid message-signature pair under keys $pk_{\text{sig}}, pk^*_{\text{san}}$. With the general message modification we now require for all queries to the signing oracle for $i = 1, 2, \ldots, q$ that $pk^*_{\text{san}} \neq pk_{\text{san},i}$ or $m^* \notin \{\textsc{mod}(m_i) \mid \textsc{mod} \text{ with } \textsc{adm}(\textsc{mod}) = 1\}$. Note that, under this general definition, it may not be efficiently verifiable if the adversary has succeeded.

The notion of privacy under general modifications demands that for each pair $(m_{j,0}, \textsc{mod}_{j,0}, \textsc{adm}_j)$, $(m_{j,1}, \textsc{mod}_{j,1}, \textsc{adm}_j)$ submitted to the left-or-right oracle the modifications are admissible and yield the same message, i.e., we simply adapt the notation $(m_{j,0}, \textsc{mod}_{j,0}, \textsc{adm}_j) \equiv (m_{j,1}, \textsc{mod}_{j,1}, \textsc{adm}_j)$ accordingly. Transparency and the accountability notions remain unchanged.

We note that both security implications (transparency implies privacy and accountability implies unforgeability) are also valid under this more general notion. The separations remain true as block-based descriptions of $\textsc{mod}$ and $\textsc{adm}$ constitute a special case.

Identification of Multiple Invalid Signatures in Pairing-Based Batched Signatures

Brian J. Matt*

Johns Hopkins University
Applied Physics Laboratory
Laurel, MD, 21102, USA
brian.matt@jhuapl.edu

Abstract. This paper describes new methods in pairing-based signature schemes for identifying the invalid digital signatures in a batch, after batch verification has failed. These methods efficiently identify non-trivial numbers of invalid signatures in batches of (potentially large) numbers of signatures.

Our methods use "divide-and-conquer" search to identify the invalid signatures within a batch, but prune the search tree to substantially reduce the number of pairing computations required. The methods presented in this paper require computing on average $O(w)$ products of pairings to identify w invalid signatures within a batch of size N, compared with the $O(w(\log_2(N/w) + 1))$ [for $w < N/2$] that traditional divide-and-conquer methods require. Our methods avoid the problem of *exponential growth* in expected computational cost that affect earlier proposals which, on average, require computing $O(w)$ products of pairings.

We compare the expected performance of our batch verification methods with previously published divide-and-conquer and exponential cost methods for Cha-Cheon identity-based signatures [6]. However, our methods also apply to a number of short signature schemes and as well as to other identity-based signature schemes.

Keywords: Pairing-based signatures, Identity-based signatures, Batch verification, Short signatures, Wireless networks.

1 Introduction

Public-key digital signatures have frequently been used in proposals for securing wireless network protocols. Proposals include methods for performing the following: combating SPAM [11]; securing routing protocols [19,30]; providing secure accounting and charging for use of the wireless network, or securely giving incentives to nodes for desirable (to the network) behavior [22,4]; protecting

* Prepared in part through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD-19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 337–356, 2009.

location and safety messages in vehicular networks [23,21]; and securely transporting ordinary messages in delay (or disruption) tolerant networks [7,26]. Even in wireless networks that have significant performance constraints such as sensor networks, it has been argued on efficiency grounds that signature schemes should be used for message authentication rather than symmetric cryptographic techniques [10,27].

When protocol designers need to select a bandwidth efficient signature scheme, they will be drawn to schemes based on bilinear pairings, such as the short signature schemes [2,5] or the bandwidth efficient identity-based signature schemes [6,5]. However, the computational cost of such schemes, especially the cost of their verification algorithms, can negatively impact the performance of wireless networks (e.g., increased delay, CPU utilization, energy consumption). Therefore, whenever circumstances allow, designers will employ batch verification methods for such pairing-based signature schemes [29,5,14]. Adversaries can attempt to negate the advantages of batch verification by corrupting messages or signatures within a batch. To counter such attacks, efficient methods are needed to identify the valid signatures within a batch that has failed initial batch verification.

To discover the valid signatures in an invalid batch, rather than verifying each signature individually, "divide-and-conquer" (DQ) techniques have been proposed [20,14]. These methods can be significantly faster than verifying individually whenever the ratio of the number of invalid signatures to the batch size is low. These methods require only $O(w(\log_2(N/w)+1))$ batch verifications and, for pairing-based signatures, product of pairings computations [13]. Recent methods for identification of invalid pairing-based signatures require $O(w)$ batch verifications [14]. When the ratio w/N is very low these methods can provide significant performance improvements over DQ methods; however, the cost of performing the batch verifications used in these methods grows exponentially, limiting their use to very small batches, and to batches with only very few invalid signatures.

Our contribution. In this paper, we present two new methods for finding invalid signatures in pairing-based schemes. These methods are based on divide-and-conquer searching, but differ from previous methods in how the (sub-) batches are verified. The average number of product of pairings computations required in our methods is $O(w)$, which is a substantial improvement over previous divide-and-conquer methods when the ratio w/N is low, and is the same complexity as the exponential cost methods. The expected number of multiplications in $\mathbb{F}_{q^d}$ required of the new methods is $O(w\sqrt{N})$, and $O(wN)$, compared to estimates of the cost of the two exponential cost methods, $O(N^{w-1}/(w-1)!)$ and $O(w^{w-1}N^{\frac{w-1}{2}}/(w-1)!)$ [14]. We have specified these methods and compared their performance for Cha-Cheon signatures [6]; however, these methods can be applied to several other pairing-based signature schemes, specifically the batched identity-based and batched short signature schemes discussed in [8].

2 Notation

In this paper we assume that pairing-based schemes use bilinear pairings on an elliptic curve E, defined over $\mathbb{F}_q$, where q is a large prime. $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct subgroups of prime order r on this curve, where $\mathbb{G}_1$ is a subset of the points on E with coordinates in $\mathbb{F}_q$, and $\mathbb{G}_2$ is a subset of the points on E with coordinates in $\mathbb{F}_{q^d}$, for a small integer d (the embedding degree). The pairing e is a map from $\mathbb{G}_1 \times \mathbb{G}_2$ into $\mathbb{G}_T$ where $\mathbb{G}_T$ is a multiplicative group of order r in the field $\mathbb{F}_{q^d}$.

We use the following notation for the components of the costs of (batch) signature verification and invalid signature identification methods for Cha-Cheon signatures. $\mathsf{CstDblPair}$ is the cost of a double product of pairings computation [13]. $\mathsf{CstMult}\mathbb{G}_1(\mathrm{t}_1)$ is the cost of multiplying an element of $\mathbb{G}_1$ by a scalar s of size $|s|$ and $t_1 = \lceil \log_2(|s|) \rceil$; likewise $\mathsf{CstDlbMult}\mathbb{G}_1(\mathrm{t}_1, \mathrm{t}_2)$ is the cost of a pair of multiplications of elements of $\mathbb{G}_1$ by scalars of size t_1 and t_2 simultaneously. $\mathsf{CstAdd}\mathbb{G}_1$ is the cost of adding two elements of $\mathbb{G}_1$, and $\mathsf{CstSub}\mathbb{G}_1$ is the cost of subtracting an element of $\mathbb{G}_1$ from another element. $\mathsf{CstInv}\mathbb{G}_\mathrm{T}$ is the cost of computing an inverse of an element in $\mathbb{G}_T$; $\mathsf{CstMult}\mathbb{G}_\mathrm{T}$ is the cost of multiplying two elements of $\mathbb{G}_T$; and $\mathsf{CstExpt}\mathbb{G}_\mathrm{T}(\mathrm{t}_1)$ is the cost of raising an element of $\mathbb{G}_T$ to the power s.

3 Background

Batch cryptography was introduced by Fiat [9], and the first batch signature scheme was that of Naccache *et al.* [18] for a variant of DSA signatures. Bellare *et al.* [1] presented three generic methods for batching modular exponentiations: *the random subset test*, the *small exponents test* (SET), and the *bucket test*, which are related to techniques in [18,28].

The inputs to the small exponents test are a security parameter l, a generator g of the group G of prime order q, and $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ with $x_i \in Z_q$ and $y_i \in G$. The verifier 1) checks that $g^{x_i} = y_i$ for all i, $1 \leq i \leq N$; 2) chooses n random integers $r_1, \ldots, r_N$ in the range $[0, 2^l - 1]$; 3) computes $x = \sum_{i=1}^{N} x_i r_i$ and $y = \prod_{i=1}^{N} {y_i}^{r_i}$; and 4) tests whether $g^x = y$ and accepts the batch if true, else rejects. If the test rejects a batch, then there is at least one invalid pair (x_i, y_i); the probability that the test accepts a batch containing invalid signatures is at most 2^{-l} [1], if the order of G is a prime [3]. One of the r's can be set to one without loss of security [14]. The small exponents test has appeared in pairing-based signature schemes [2,5] including, in [14], as the batch verifier for the Cha-Cheon signature scheme [6].[1]

Cha-Cheon Identity-Based Signature scheme. $H(m, U)$ is a cryptographic hash that maps a bit string m and a point $U \in \mathbb{G}_1$ to an integer between 1 and r.

[1] For Cha-Cheon signatures with the cost parameters in Section 5, SET always verifies a valid batch more efficiently than the random subset test, and the bucket test verifies a valid batch more efficiently than SET when the batch size exceeds 512.

1. *Setup:* The system manager selects an order r point $T \in \mathbb{G}_2$ and randomly selects an integer s in the range $[1, r-1]$. The manager computes $S = sT$. The public system parameters are T and S. The system manager's secret key is s.
2. *Extract:* Each user is given a key pair. The user's public key, Q, is a point in $\mathbb{G}_1$ that is derived from the user's identity using a public algorithm. The user's private key is $C = sQ$.
3. *Sign:* To sign a message m, the signer randomly generates an integer t in the range $[1, r-1]$ and outputs a signature (U, V) where $U = tQ$ and $V = (t + H(m, U))C$.
4. *Verify:* To verify a signature (U, V) of message m, the verifier derives the signer's public key Q from the alleged signer's identity and computes $h = H(m, U)$. If $e(U + hQ, S) = e(V, T)$ then the signature is accepted. Otherwise, the signature is rejected. This test can be rewritten as $1 = e(U + hQ, S) \cdot e(V, -T)$ which can be computed more efficiently [13].

In [14] the following batch verifier was presented. The verifier obtains N messages m_i, for $i = 1$ to N, and the signatures (U_i, V_i) and signer's identity for each message. The verifier derives each public key Q_i from the signer's identity and checks that U_i and V_i are elements of $\mathbb{G}_1$. The verifier sets $r_1 = 1$ and generates random values r_i from $[0, 2^l - 1]$, for $i = 2$ to N. The batch is valid if

$$1 = \alpha_0 = e\left(\sum_{i=1}^{N} B_i, S\right) \cdot e\left(\sum_{i=1}^{N} D_i, -T\right)$$

where $B_i = r_i\left(U_i + H(m_i, U_i) \cdot Q_i\right)$, $D_i = r_i V_i$, and 1 is the identity in $\mathbb{F}_{q^d}$.

3.1 Identifying Invalid Signatures

The problem of identifying invalid signatures within a batch has only recently been investigated. Work in this area generally falls into three categories: divide-and-conquer methods [20,14], identification code based methods [20], and expo-nent testing methods [15,16,25,14].

Divide-and-Conquer Methods. Pastuszak *et al.* [20] first investigated methods for identifying invalid signatures within a batch. They explored "divide-and-conquer" methods for generic batch verifiers, i.e., methods that work with any of the three batch verifiers studied by Ballare *et al.* In these methods the set of signatures in an invalid batch is repeatedly divided into $d \geq 2$ smaller sub-batches to verify. The most efficient of their techniques, the *Fast DC Verifier* method, exploits knowledge of the results of the first $d-1$ sub-batch verifications to determine whether the verification of the d^{th} sub-batch is necessary, i.e., if a (sub-)batch *batch* is invalid and the first $d-1$ sub-batches of *batch* are all valid, then the d'th sub-batch must be invalid, and the batch verifier for that sub-batch is not computed. Performance measurements of one of the methods of [20] for the Boneh, Lynn and Shacham (BLS) [2] signature scheme have been reported [8]. The authors observed that the divide-and-conquer method they studied outperformed verifying each signature individually when $w/N < .15$ in batches of 1024 BLS signatures using 160-bit MNT curves.

In [14] a more efficient divide-and-conquer method called Binary Quick Search (BQS) was presented; BQS is applicable to small exponents test based verifiers. In this method a batch verifier that compares two quantities, X and Y, is replaced with an equivalent test $A = XY^{-1}$, and the batch is accepted if $A = 1$.[2] The BQS algorithm is always[3] more efficient than any $d = 2^n$ary DC Verifier; When it is necessary to verify the d^{th} child sub-batch, in BQS the sub-batch can be verified by simply computing a single inverse operation and a single multi-term multiplication (or $d - 1$ ordinary multiplications) rather than the much more expensive batch verification required by the Fast DC verifier. The upper bound of the number of batch verifications required by BQS is half that of the Fast DC Verifier for $d = 2$ [14].

Identification Code Based Methods. Pastuszak *et al.* [20] investigated using a Hamming identification code and a two-layer Hamming identification code for identifying invalid signatures in generic batch verification. The Hamming code verifier can identify a single error in a batch of size $2^n - 1$ using $n + 2$ batch verifications, and the two-layer verifier can identify 2 invalid signatures in a batch of $2^n - 2$ signatures using $3n + 3$ batch verifications.

Exponent Testing Methods. The first exponent testing method, developed by Lee *et al.* [15], was capable of finding a single invalid signature within a batch of "DSA-type" signatures. Signatures of this type have verification equations of the form "$g^m = s \mod p$" where m is the message, s is the signature, the generator g has order q, and p and q are primes where $q \mid (p - 1)$. To identity an invalid signature, compute $X = \prod_{i=1}^{N} s_i / g^{\sum_{i=1}^{N} m_i}$ and $Y = \prod_{i=1}^{N} s_i^i / g^{\sum_{i=1}^{N} i \cdot m_i}$ and test whether $Y = X^z$ for $z \in [1, N]$. The Exponentiation method of Law and Matt [14], for the special case of identifying a single invalid signature, is similar to the above method.

Lee et al. [16] applied their approach for DSA-type signatures to identifying a single invalid signature in batches of RSA signatures. They addressed the problem of identifying multiple invalid RSA signatures by using their RSA method in a divide-and-conquer method that is somewhat similar to the Single Pruning Search we present in Section 4. However, Stanek showed in [25] that their approach for RSA signatures is not secure.

In [14] two exponent testing methods for pairing-based batch signatures, the Exponentiation method and the Exponentiation with Sectors method, were presented. Both methods require computing a number of batch verifications that are proportional to the number of invalid signatures w in the batch. The Exponentiation method requires $w+1$ verifications (including the initial batch verification)

[2] For the initial batch verification, if it is more efficient to do so compute X and Y, compare them, and compute $A = XY^{-1}$ if the comparison fails; otherwise A is computed directly, e.g., in Cha-Cheon where $A = \alpha_0$.

[3] Except when $w = 1$ and the invalid signature is located in the rightmost position in the batch then Fast DC verifier and BQS have equal costs.

and the same number of product of pairings computations. Exponentiation with Sectors requires at most $2w+1$ product of pairings computations. Both methods use exhaustive search during batch verification, resulting in exponential cost.

Exponentiation Method. For the Cha-Cheon signature scheme, compute α_0 and test whether α_0 is equal to the identity. If so, the batch is valid. Otherwise compute $\alpha_j, w \geq j \geq 1$,

$$\alpha_j = e\left(\sum_{i=1}^{N} i^j B_i, S\right) e\left(\sum_{i=1}^{N} i^j D_i, -T\right), \tag{1}$$

and perform a test on the values $\alpha_j, \alpha_{j-1}, \ldots \alpha_0$. For $j = 1$, test whether $\alpha_1 = \alpha_0^{z_1}$ has a solution for $1 \leq z_1 \leq N$ using Shanks' giant-step baby-step algorithm [24]. If successful, $w = 1$ and z_1 is the position of the invalid signature. In general the method tests whether

$$\alpha_j = \prod_{t=1}^{j} (\alpha_{j-t})^{(-1)^{t-1} p_t} \tag{2}$$

has a solution where p_t is the tth elementary symmetric polynomial in $1 \leq z_1 \leq \ldots \leq z_j \leq N$. The authors show that the tests can be performed in $O(\sqrt{N})$ for $j = 1$ and $O(N^{j-1}/(j-1)!)$ for $j \geq 2$ multiplications in $\mathbb{F}_{q^d}$. If a test fails increment j, compute α_j, and test. When $j = w$ the test will succeed, and the values of $z_1, \ldots, z_w$ are the positions of the invalid signatures.

Exponentiation with Sectors Method. The Exponentiation with Sectors Method uses two stages. In the first stage, the batch is divided into approximately $\sqrt{N}$ sectors of approximately equal size and the Exponentiation method is used, where each B_i, and D_i within a sector is multiplied by the same constant, to identify the v invalid sectors. In the second stage, the Exponentiation method is used to find the invalid signatures within a batch consisting of the signatures from the v invalid sectors. This method requires $w + v + 1$ product of pairings computations, including the initial verification, where $v \leq \min(w, \sqrt{N})$. During the first stage the tests can be performed in $O(N^{\frac{1}{4}})$ for $j = 1$ and $O(\sqrt{N}^{j-1}/(j-1)!)$ for $j \geq 2$ multiplications in $\mathbb{F}_{q^d}$. During the second stage the number of multiplications required for $w \leq j \geq v$ is $O(\sqrt{v\sqrt{N}})$ for $j = 1$, and $O(v\sqrt{N}^{j-1}/(j-1)!)$ for each $j \geq 2$.

4 An Alternate Approach to Divide-and-Conquer Methods

Divide-and-conquer methods can be viewed as operating on (for simplicity) a binary tree T with $w \geq 1$ invalid signatures whose root node, $root_T$, is the batch, and each pair of child nodes represents the two nearly equal size sub-batches of their parent. Previously published methods such as Binary DC Verifier and BQS identify the invalid signatures within the initial batch by descending through the tree, performing verifications on the sub-batches of the nodes they encounter. When one of the methods reaches a node whose sub-batch is valid, the methods do not visit its descendants, if any. The methods identify the invalid signatures by identifying those nodes that are either the ancestors of the leaf nodes of T that represent invalid signatures, or the leaf nodes themselves. The difference between the published methods are 1) the degree of the tree and 2) how efficiently the nodes of the tree are verified.

The methods we propose view T as consisting of a parent sub-tree PT with root node $root_{PT} = root_T$, and the leaves of PT are the roots of the w maximal sub-trees ST_i, $i = 1, .., w$, of T which represent sub-batches that have a single invalid signature. If the $w = 1$, then $T = ST_1$ and PT is the node $root_T$. The new methods identify invalid signatures by descending through PT and identifying its leaves, and concurrently identifying the single invalid signature in each of the sub-batches these leaves represent.

For signature schemes such as the Cha-Cheon, *node* of T is the root of some ST_i if there exists a value z, $lb \leq z \leq ub$, that is a solution to $\alpha_{1,node} = \alpha_{0,node}^z$. The values lb and ub are the lower and upper bounds of the sub-batch represented by *node* within B, and $\alpha_{j,node} = e\left(\sum_{i=lb}^{ub} i^j B_i, S\right) \cdot e\left(\sum_{i=lb}^{ub} i^j D_i, -T\right)$. Shanks' giant-step baby-step algorithm can determine if such a solution exists in time (multiplications in $\mathbb{F}_{q^d}$) proportional to the square root of the size of the sub-batch. If no solution is found, then *node* is in PT but is not a leaf; hence its children must be tested. We refer to this approach as *single pruning.*

When the children of an interior node p in PT, l and its sibling r, are leaves of PT, there exist values z_l in the range of indexes of the signatures in the left sub-batch and z_r in the range of signatures in the right sub-batch, such that $\alpha_{1,p} = \alpha_{0,l}^{z_l} \cdot \alpha_{0,r}^{z_r}$. The values z_l and z_r can be determined using an algorithm, $PairSolver(Left, Right)$, with cost proportional to the size of the (sub-)batch represented by p. If the algorithm fails, then at least one of the child nodes is not a leaf of PT and they are tested individually using Shanks' algorithm. We refer to this approach as *paired single pruning.*

4.1 Single Pruning Search (SPS) Method

The recursive algorithm below describes the Single Pruning Search (SPS) method on a batch B which is a list of $N = 2^h$, $h \geq 1$, randomly ordered message / signature pairs $((m_1, s_1), \ldots, (m_N, s_N))$ where the signature components for Cha-Cheon are verified elements of $\mathbb{G}_1$. On the initial call to $SPS(X, \alpha_{0,P}, \alpha_{1,P})$, $X = B$, $\alpha_{0,P} = 1$, $\alpha_{1,P} = 1$. SPS uses the following algorithms:

1. $Get_0(X)$ – checks whether α_0 has been computed for X and if so returns it; otherwise it computes α_0 by the most efficient method available, and it may compute α_0^{-1} [17].
2. $Get_1(X)$ – checks whether α_1 has been computed for X and if so returns it; otherwise it computes α_1 by the most efficient method available, and it may compute α_1^{-1} [17].

3. $Shanks(X)$ – if X has a single invalid signature, the algorithm returns the position of the invalid signature; otherwise the algorithm returns 0 [17].
4. $Left(X)$ – returns a sub-batch with the first $len/2$ pairs in X, or $\varnothing$ if $X = \varnothing$.
5. $Right(X)$ – returns a sub-batch with the later $len/2$ pairs in X, or $\varnothing$ if $X = \varnothing$.
6. $Len(X)$ – returns the number of pairs in X, or 0 if $X = \varnothing$.

Algorithm. $SPS(X, \alpha_{0,P}, \alpha_{1,P})$ *(Single Pruning Search)*

Input: X a list of message / signature pairs, $\alpha_{0,P}$ and $\alpha_{1,P}$ in $\mathbb{G}_T$.
Output: A list of the invalid pairs in the batch.
Return: A boolean.

```
if ($Len(X) = 1$) then
    output $X$
    return ($true$)
else
    $\alpha_{0,N} \leftarrow Get_0(X)$
    if ($\alpha_{0,N} = 1$) then
        return ($true$)
    elseif ($X = B$) then
        $inv\alpha_{0,[B]} \leftarrow \alpha_{0,N}^{-1}$
    endif
    $\alpha_{1,N} \leftarrow Get_1(X)$
    if ($\alpha_{0,N} \neq \alpha_{0,P}$) then
        $z \leftarrow Shanks(X)$
        if ($z \neq 0$) then
            output $(m_z, s_z)$
            return ($true$)
        elseif ($X = B$) then
            $inv\alpha_{1,[B]} \leftarrow \alpha_{1,N}^{-1}$
        endif
    endif
    if ($SPS(Left(X), \alpha_{0,N}, \alpha_{1,N})$) then
        $SPS(Right(X), \alpha_{0,N}, \alpha_{1,N})$
    endif
    return ($\alpha_{0,N} \neq \alpha_{0,P}$)
endif
```

$Get_0(X)$ computes products of pairings only for the root node and left children nodes that are tested by SPS. $Get_1(X)$ only computes products of pairings for the root and for each left child X tested by SPS when the α_0 of X is not equal to α_0 of the parent of X.

4.2 Paired Single Pruning Search Method

The recursive algorithm below describes the Paired Single Pruning Search (PSPS) method on a batch B, which is a list of $N = 2^h, h \geq 1$, randomly ordered message /

signature pairs $((m_1, s_1), \ldots, (m_N, s_N))$ where the signature components for Cha-Cheon are verified elements of $\mathbb{G}_1$. On the initial call to $PSPS(X, \alpha_{0,P}, \alpha_{1,P})$, $X = B$, $\alpha_{0,P} = 1$, $\alpha_{1,P} = 1$. PSPS also uses the following algorithms:

1. $PairSolver(Left, Right)$ – returns the positions of two invalid signatures, one in $Left$ and one in $Right$, or returns $(0, 0)$ [17].
2. $Parent(X)$ – returns the parent of X, or $\varnothing$ if X is the initial batch B.

Algorithm. $PSPS(X, \alpha_{0,P}, \alpha_{1,P})$ *(Paired Single Pruning Search)*
Input: X a list of message / signature pairs, $\alpha_{0,P}$ and $\alpha_{1,P}$ in $\mathbb{G}_T$.
Output: A list of the invalid pairs in the batch.
Return: A boolean.

```
if ($Len(X) = 1$) then
    output $X$
    return ($true$)
else
    $\alpha_{0,N} \leftarrow Get_0(X)$
    if $\alpha_{0,N} = 1$ then
        return ($true$)
    elseif ($X = B$) then
        $inv\alpha_{0,[B]} \leftarrow \alpha_{0,N}^{-1}$
    endif
    if ($\alpha_{0,N} = \alpha_{0,P}$) then
        $\alpha_{1,N} \leftarrow Get_1(X)$
    else
        if ($X \neq B$ and $X = Left(Parent(X))$) then
            $(z_l, z_r) \leftarrow PairSolver(X, Right(Parent(X)))$
            if $z_l \neq 0$ then
                output $(m_{z_l}, s_{z_l}), (m_{z_r}, s_{z_r})$
                return ($false$)
            end
        end
        $\alpha_{1,N} \leftarrow Get_1(X)$
        $z \leftarrow Shanks(X)$
        if $z \neq 0$ then
            output $(m_z, s_z)$
            return ($true$)
        elseif ($X = B$) then
            $inv\alpha_{1,[B]} \leftarrow \alpha_{1,N}^{-1}$
        endif
    endif
    if ($PSPS(Left(X), \alpha_{0,N}, \alpha_{1,N})$) then
        $PSPS(Right(X), \alpha_{0,N}, \alpha_{1,N})$
    endif
    return ($\alpha_{0,N} \neq \alpha_{0,P}$)
endif
```

5 Performance

For Cha-Cheon signatures, the divide-and-conquer methods and the exponentiation methods batch verify by first checking that the signature components are in $\mathbb{G}_1$, then computing α_0 for B, and testing whether $\alpha_0 = 1$. With the exception of BQS and the DC Verifiers, they compute their α_0s, as shown in $Get_0(B)$ in Appendix A. The cost (not including the membership tests) is $N \cdot \mathsf{CstDlbMult}\mathbb{G}_1(\mathrm{t}_1, \mathrm{t}_2) + N \cdot \mathsf{CstMult}\mathbb{G}_1(\mathrm{t}_1) + 2(N-1)\,\mathsf{CstAdd}\mathbb{G}_1 + \mathsf{CstDblPair}$ with $t_1 = \lceil \log_2(r)/2 \rceil$ and $t_2 = \lceil \log_2(r) \rceil$.[4]

5.1 Cost of the New Methods When $w \geq 1$

Single Pruning Search Performance. If $w = 1$, the cost of SPS increases by the cost of computing α_0^{-1} ($\mathsf{CstInv}\mathbb{G}_\mathrm{T}$) and α_1 for B ($2(N-1)\,\mathsf{CstAdd}\mathbb{G}_1$ $+\,\mathsf{CstDblPair}$), plus the expected cost of a successful $Shanks(B)$ call, which is approximately $\frac{4}{3}\sqrt{N}\,\mathsf{CstMult}\mathbb{G}_\mathrm{T}$.[5] If $w \geq 2$, the average cost of SPS is the sum of the costs of computing α_0, α_0^{-1}, α_1, a failed $Shanks(B)$ call ($2\sqrt{N}\,\mathsf{CstMult}\mathbb{G}_\mathrm{T}$), α_1^{-1}, plus the sum of the costs generated as SPS investigates the descendents of $root_T$. The following recurrence relation generates these costs:

$$\mathsf{R}_{(\mathsf{S})}(w, M) =$$

$$\begin{cases} 0, & \begin{array}{r} w = 0, 1, \\ w > M \text{ or} \\ w = 2 \text{ and } M = 2; \end{array} \\[2ex] \dfrac{\left[\begin{array}{l} 2\binom{M/2}{2}(\mathsf{R}_{(\mathsf{S})}(2, M/2) + \mathsf{C}_{(\mathsf{S})}(2, 0, M/2)) + \\ \binom{M/2}{1}^2 (\mathsf{C}_{(\mathsf{S})}(1, 1, M/2)) \end{array}\right]}{\binom{M}{2}}, & w = 2 \text{ and } M > 2; \\[4ex] \dfrac{\left[\begin{array}{l} 2\binom{M/2}{w}(\mathsf{R}_{(\mathsf{S})}(w, M/2) + \mathsf{C}_{(\mathsf{S})}(w, 0, M/2)) + \\ 2\binom{M/2}{w-1}\binom{M/2}{1}(\mathsf{R}_{(\mathsf{S})}(w-1, M/2) + \mathsf{C}_{(\mathsf{S})}(w-1, 1, M/2)) + \\ \sum\limits_{i=2}^{w-2} \binom{M/2}{w-i}\binom{M/2}{i}(\mathsf{R}_{(\mathsf{S})}(w-i, M/2) + \mathsf{R}_{(\mathsf{S})}(i, M/2) + \mathsf{C}_{(\mathsf{S})}(w-i, i, M/2)) \end{array}\right]}{\binom{M}{w}}, & w \geq 3, \end{cases}$$

[4] BQS and the DC Verifiers can compute α_0 with the same cost [17].

[5] $Shanks(X)$ tests whether the equation $\alpha_{1,n} = \alpha_{0,n}^z$ has a solution z in the range of l up to u, the bounds of the (sub-)batch X within the original batch. The algorithm alternately computes a value from the series $\alpha_1 \cdot (\alpha_0^{-1})^i$ and the series $(\alpha_0^s)^j \cdot \alpha_0^l$, $s = \lfloor \sqrt{|X|} \rfloor$, stopping when a match is found (single invalid signature) or when both series have been computed. Similarly, $PairSolver(Left, Right)$ alternately computes a value from one of two series, and terminates when a newly computed value from one series is equal to one of the values already computed for the other series, or when both series have been computed.

where for Cha-Cheon:

	Costs		
Argument	$\mathsf{CstDblPair}$	$\mathsf{CstInv}\mathbb{G}_\mathrm{T}$	$\mathsf{CstMult}\mathbb{G}_\mathrm{T}$
$\mathsf{C}_{(\mathsf{S})}(2,0,M/2)$	1		
$\mathsf{C}_{(\mathsf{S})}(1,1,M/2)$	2	2	$\frac{8}{3}\sqrt{M/2}$
$\mathsf{C}_{(\mathsf{S})}(w,0,M/2)$	1		
$\mathsf{C}_{(\mathsf{S})}(w-1,1,M/2)$	2	2	$\frac{10}{3}\sqrt{M/2}$
$\mathsf{C}_{(\mathsf{S})}(w-i,i,M/2)$	2	2	$4\sqrt{M/2}$

For $\mathsf{C}_{(\mathsf{S})}(2,0,M/2)$ and $\mathsf{C}_{(\mathsf{S})}(w,0,M/2)$, $Get_0(X)$ is called for the left child node and no inverse is computed, with cost $\mathsf{CstDblPair}$. For $\mathsf{C}_{(\mathsf{S})}(1,1,M/2)$, both $Get_0(X)$ and $Get_1(X)$ are called for the left child, combined cost is $2\,\mathsf{CstDblPair} + 2\,\mathsf{CstInv}\mathbb{G}_\mathrm{T}$; for the right child, cost is zero, and two successful calls are made to $Shanks(X)$ with combined cost of $\frac{8}{3}\sqrt{M/2}\,\mathsf{CstMult}\mathbb{G}_\mathrm{T}$. $\mathsf{C}_{(\mathsf{S})}(w-1,1,M/2)$ is similar to $\mathsf{C}_{(\mathsf{S})}(1,1,M/2)$ except that one of the calls to $Shanks(X)$ fails to find a solution. Both calls to $Shanks(X)$ fail for $\mathsf{C}_{(\mathsf{S})}(w-i,i,M/2)$.

Paired Single Pruning Search Performance. If $w \leq 1$, PSPS has the same average cost as SPS. If $w \geq 2$, the average cost of PSPS is the sum of the costs of computing α_0, α_0^{-1}, α_1, α_1^{-1}, the cost of the failed *Shanks* test on the batch B, and the sum of the costs generated as PSPS investigates the descendants of $root_T$. The following recurrence relation generates these costs:

$$\mathsf{R}_{(\mathsf{P})}(w,M) =$$

$$\begin{cases} 0, & w = 0,1, \\ & w > M \text{ or} \\ & w = 2 \text{ and } M = 2; \\[2ex] \dfrac{\left[\begin{array}{l} 2\binom{M/2}{2}(\mathsf{R}_{(\mathsf{P})}(2,M/2) + \mathsf{C}_{(\mathsf{P})}(2,0,M/2)) + \\ \binom{M/2}{1}^2(\mathsf{C}_{(\mathsf{P})}(1,1,M/2)) \end{array}\right]}{\binom{M}{2}}, & w = 2 \text{ and } M > 2; \\[4ex] \dfrac{\left[\begin{array}{l} 2\binom{M/2}{w}(\mathsf{R}_{(\mathsf{P})}(w,M/2) + \mathsf{C}_{(\mathsf{P})}(w,0,M/2)) + \\ 2\binom{M/2}{w-1}\binom{M/2}{1}(\mathsf{R}_{(\mathsf{P})}(w-1,M/2) + \mathsf{C}_{(\mathsf{P})}(w-1,1,M/2)) + \\ \sum\limits_{i=2}^{w-2}\binom{M/2}{w-i}\binom{M/2}{i}(\mathsf{R}_{(\mathsf{P})}(w-i,M/2) + \mathsf{R}_{(\mathsf{P})}(i,M/2) + \mathsf{C}_{(\mathsf{P})}(w-i,i,M/2)) \end{array}\right]}{\binom{M}{w}}, & w \geq 3, \end{cases}$$

where for Cha-Cheon:

	Costs		
Argument	CstDblPair	$\mathsf{CstInv}\mathbb{G}_\mathrm{T}$	$\mathsf{CstMult}\mathbb{G}_\mathrm{T}$
$\mathsf{C}_{(\mathsf{P})}(2,0,M/2)$	1		
$\mathsf{C}_{(\mathsf{P})}(1,1,M/2)$	1	1	$M/2$
$\mathsf{C}_{(\mathsf{P})}(w,0,M/2)$	1		
$\mathsf{C}_{(\mathsf{P})}(w-1,1,M/2)$	2	2	$M+\frac{10}{3}\sqrt{M/2}$
$\mathsf{C}_{(\mathsf{P})}(w-i,i,M/2)$	2	2	$M+4\sqrt{M/2}$

For $\mathsf{C}_{(\mathsf{P})}(2,0,M/2)$ and $\mathsf{C}_{(\mathsf{P})}(w,0,M/2)$, $Get_0(X)$ is called for the left child node and no inverse is computed. For $\mathsf{C}_{(\mathsf{P})}(1,1,M/2)$, $Get_0(X)$ is called for the left child, the cost is $\mathsf{CstDblPair}+\ \mathsf{CstInv}\mathbb{G}_\mathrm{T}$, and a successful call is made to $PairSolver(Left, Right)$ with expected cost of $M/2\ \mathsf{CstMult}\mathbb{G}_\mathrm{T}$. For the argument $\mathsf{C}_{(\mathsf{P})}(w-1,1,M/2)$, both $Get_0(X)$ and $Get_1(X)$ are called for both the left and right child, one failed call is made to $PairSolver(Left, Right)$ with cost $M\ \mathsf{CstMult}\mathbb{G}_\mathrm{T}$, and one successful and one failed call are made to $Shanks(X)$. $\mathsf{C}_{(\mathsf{P})}(w-i,i,M/2)$ is similar to $\mathsf{C}_{(\mathsf{P})}(w-1,1,M/2)$ except that both calls to $Shanks(X)$ fail.

5.2 Number of Product of Pairings Computations of SPS and PSPS

Let T be a perfect binary tree, and $\mathrm{PT}_{(2)}$ be the sub-tree of PT, where each node represents 2 or more invalid signatures. For each node in $\mathrm{PT}_{(2)}$, SPS computes an α_0 for its left child, unless the child is a leaf node of T. For each node in the $\mathrm{PT}_{(2)}$ with both child nodes in PT, SPS also computes α_1 for its left child, unless the child is a leaf node of T. This computation occurs $w-1$ times. SPS also computes a pair of αs for the root. Therefore, including the initial batch verification, SPS requires $|\,\mathrm{PT}_{(2)}\,|+(w+1)$ product of pairings computations, if none of the leaf nodes of PT are leaves of T. SPS requires two fewer α computations whenever a pair of leaves of PT are leaves of T.

For a perfect binary tree, the number of ways j pairs leaves of PT can be leaves of T is $\binom{N/2}{j}$, and the number of ways the remaining $w-2j$ invalid signatures can be in the remaining $N/2-j$ distinct 3 node subtrees at the lowest level of T is $\binom{N/2-j}{w-2j}2^{w-2j}$. Therefore, the expected number of occurrences of two sibling leaf nodes of T both representing invalid signatures is

$$\frac{1}{\binom{N}{w}}\sum_{j=1}^{\lfloor w/2\rfloor}\binom{N/2}{j}\binom{N/2-j}{w-2j}2^{w-2j}.$$

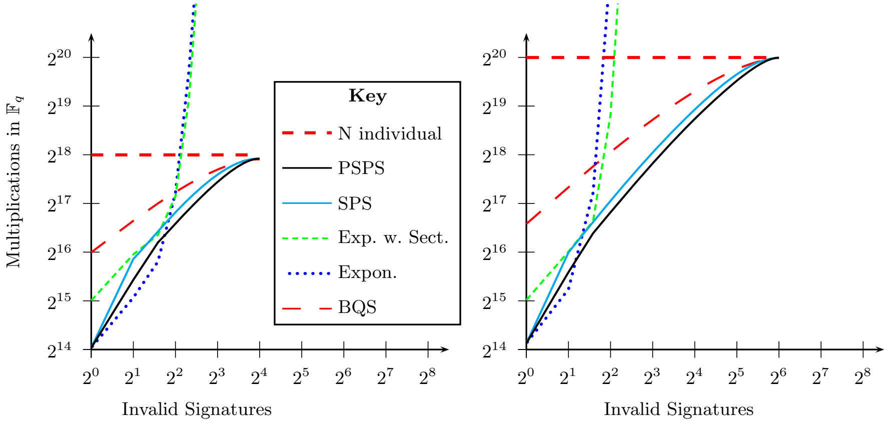
Since $\binom{N-2}{w-2}=\sum_{j=1}^{\lfloor w/2\rfloor}\binom{N/2}{j}\binom{N/2-j}{w-2j}2^{w-2j}$ the expression simplifies to $\frac{w(w-1)}{2(N-1)}$, and the expected number of αs computed by Single Pruning Search when the batch size is a power of 2 is

$$|\,\mathrm{PT}_{(2)}\,|+(w+1)-\frac{w(w-1)}{N-1}\ .$$

In Appendix B we show that $|\,\mathrm{PT}_{(2)}\,| < 2w - 1$; therefore the expected number of product of pairings computations required by SPS is less than $3w$. Since the number of product of pairings computations in the cost functions of PSPS are all less than or equal to the corresponding functions of SPS , the average number of product of pairings computations used by PSPS is also $O(w)$.

5.3 Number of Multiplications in F_q

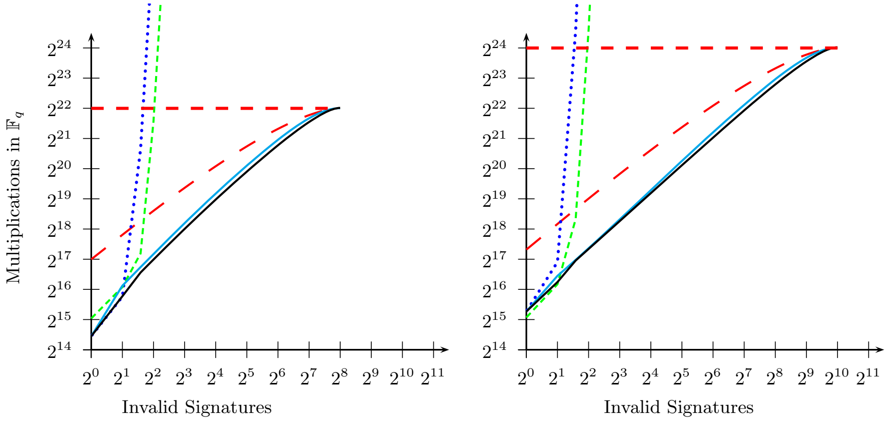
Figure 1 and Figure 2 compare methods analyzed in Section 5.1 and in [17] for finding invalid signatures in a batch once the initial batch verification has failed



(a) Batch Size 16

(b) Batch Size 64

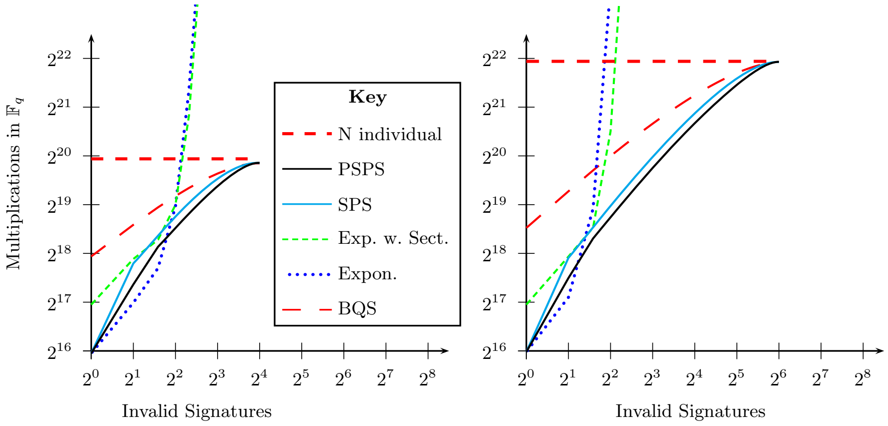
Fig. 1. (a,b) Number of multiplies in $\mathbb{F}_q$, where r and q are 160-bit values and $d = 6$



(c) Batch Size 256

(d) Batch Size 1024

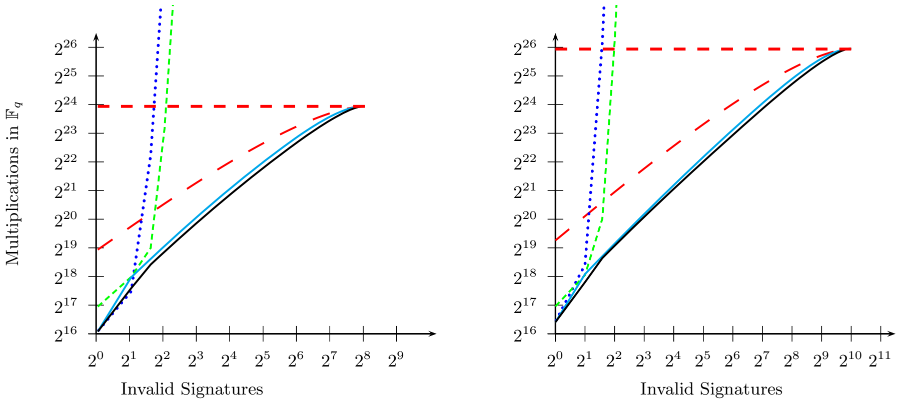
Fig. 1. (c,d) Number of multiplies in F_q, where r and q are 160-bit values and $d = 6$



(a) Batch Size 16

(b) Batch Size 64

Fig. 2. (a,b) Number of multiplies in $\mathbb{F}_q$, where r and q are 256-bit values and $d = 12$



(c) Batch Size 256

(d) Batch Size 1024

Fig. 2. (c,d) Number of multiplies in F_q, where r and q are 256-bit values and $d = 12$

for Cases A and C of [12]. In Case A, the group order r is a 160-bit value, the elliptic curve E is defined over $\mathbb{F}_q$, where q is a 160-bit value, and the embedding degree $d = 6$. In Case C, the group order r is a 256-bit value, q is a 256-bit value, and the embedding degree $d = 12$. All costs are given in terms of the number of multiplications (m) in $\mathbb{F}_q$ using the following estimates from Granger, Page and Smart [12], and Granger and Smart [13].

– For Case A, 1 double product of pairings $= 16,355m$, 1 multiplication in $F_{q^6} = 15m$, 1 inverse in $F_{q^6} = 44m$ (assuming 1 inverse in $F_q = 10m$), 1 elliptic curve addition $= 11m$, and an elliptic point multiplication by a 160-bit value is $1614m$ and by an 80-bit value is $827m$.

- For Case C, 1 double product of pairings $= 62,797m$, 1 multiplication in $F_{q^{12}} = 45m$, 1 inverse in $F_{q^{12}} = 104m$, 1 elliptic curve addition $= 11m$, and an elliptic point multiplication by a 256-bit value is $2535m$ and by an 128-bit value is $1299m$.

6 Conclusion

We have presented two new methods, Single Pruning Search and Paired Single Pruning Search, for identifying invalid signatures in pairing-based batch signature schemes using the small exponents test, and have analyzed their average case performance. These new methods require $O(w)$ product of pairings computations and $O(w\sqrt{N})$ and $O(wN)$ number of multiplications in $\mathbb{F}_{q^d}$. The methods are described for Cha-Cheon signatures, but are applicable to other batch verified signature schemes such as the batch verifiers presented in [8].

These new methods, like BQS and earlier divide-and-conquer methods, can be used when there is uncertainty in the number of invalid signatures in a batch. As shown in the figures in Section 5.3, the new methods significantly outperform the Binary Quick Search method when $w \ll N$, and perform as well as or better than the exponentiation methods except when N and w are small. Unlike the exponentiation methods, with the new methods a batch verifier is not forced to switch methods when tests for small w fail.

In [14] the authors suggested that the exponentiation methods can be used with BQS to provide improved performance after tests for small values of w fail. While this is certainly true, the result can be expensive. For example, with $N = 64$, a batch verifier that assumes that the number of invalid signature in the batch is small would start with the Exponentiation Method, but if the tests for $w = 1$ and $w = 2$ both fail, the verifier would switch to Exponentiation with Sectors Method to test if $w = 3$. If the test for $w = 3$ fails, then the verifier would switch to BQS. If $w = 4$, the cost of this sequence for Case A (ignoring the common cost of signature component validation and α_0 computation) is at least $\approx 3.70 \times 10^5$ multiplications in $\mathbb{F}_q$, compared to $\approx 2.73 \times 10^5$ multiplications if only BQS was used, and $\approx 1.16 \times 10^5$ multiplications with the Paired Single Pruning Search Method.

Ideally, we would have a single efficient method for finding the invalid signatures in a batch that always has the lowest expected cost no matter how many signatures are invalid. Such a method would be especially useful when an adversary is occasionally able to inject bursts of several invalid signatures into some batches. Short of that ideal, but a practical alternative, would be a small set of methods, each of which for some range of batch sizes of interest always provides the lowest expected cost. Currently the Paired Single Pruning Search, provides the lowest expected cost when the batch size is in the range 128 to 512. For batches larger than 512, we would expect batch verifiers to utilize the bucket test for Cha-Cheon and related signature schemes rather than the small exponents test. Finding such a minimal cost method for batches smaller than 128 is as an open problem. Another open problem is to find more efficient methods than the generic DC verifiers of [20] for identifying the invalid signatures in a batch when an initial bucket test verifier fails.

References

1. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
2. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
3. Boyd, C., Pavlovski, C.: Attacking and repairing batch verification schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 58–71. Springer, Heidelberg (2000)
4. Buchegger, S., Boudec, J.-Y.L.: Performance analysis of the CONFIDANT protocol (Cooperation of Nodes: Fairness In Dynamic Ad-hoc NeTworks). In: ACM/SIGMOBILE Third International Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC). ACM, New York (2002)
5. Camenisch, J., Hohenberger, S., Pedersen, M.: Batch verification of short signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007); see also Cryptology ePrint Archive, Report 2007/172, 2007, http://eprint.iacr.org/2007/172
6. Cha, J., Cheon, J.: An identity-based signature from gap diffie-hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2002)
7. Fall, K.: A delay-tolerant network architecture for challenged internets. In: SIGCOMM 2003: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 27–34 (2003)
8. Ferrara, A.L., Green, M., Hohenberger, S., Pedersen, M.O.: On the practicality of short signature batch verification. Cryptology ePrint Archive, Report 2008/015 (2008), http://eprint.iacr.org/2008/015
9. Fiat, A.: Batch RSA. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 175–185. Springer, Heidelberg (1990)
10. Gaubatz, G., Kaps, J.-P., Sunar, B.: Public key cryptography in sensor networks—revisited. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 2–18. Springer, Heidelberg (2005)
11. Gavidia, D., van Steen, M., Gamage, C., Jesi, G.P.: Canning spam in wireless gossip networks. In: Conference on Wireless On demand Network Systems and Services (WONS), pp. 208–220 (2007)
12. Granger, R., Page, D.L., Smart, N.P.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006)
13. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive, Report 2006/172 (2006), http://eprint.iacr.org/2006/172
14. Law, L., Matt, B.J.: Finding invalid signatures in pairing based batches. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 35–53. Springer, Heidelberg (2007)

15. Lee, S.-W., Cho, S., Choi, J., Cho, Y.: Batch verification with DSA-type digital signatures for ubiquitous computing. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS, vol. 3802, pp. 125–130. Springer, Heidelberg (2005)
16. Lee, S., Cho, S., Choi, J., Cho, Y.: Efficient identification of bad signatures in RSA-type batch signature. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E89-A(1), 74–80 (2006)
17. Matt, B.J.: Identification of multiple invalid signatures in pairing-based batched signatures. Cryptology ePrint Archive (2009), http://eprint.iacr.org/2009
18. Naccache, D., M'Raïhi, D., Vaudenay, S., Raphaeli, D.: Can D.S.A. Be improved? In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 77–85. Springer, Heidelberg (1995)
19. Papadimitratos, P., Haas, Z.: Secure routing for mobile ad hoc networks. In: Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002) (January 2002)
20. Pastuszak, J., Michałek, D., Pieprzyk, J., Seberry, J.: Identification of bad signatures in batches. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 28–45. Springer, Heidelberg (2000)
21. Raya, M., Hubaux, J.-P.: Securing vehicular ad hoc networks. Journal of Computer Security, Special Issue on Security of Ad Hoc and Sensor Networks 15(1), 39–68 (2007)
22. Salem, N.B., Buttyan, L., Hubaux, J.-P., Jakobsson, M.: A charging and rewarding scheme for packet forwarding in multi-hop cellular networks, In: ACM/SIGMOBILE 4th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC). ACM Press, New York (2003)
23. Sampigethaya, K., Mingyan, L., Leping, H., Poovendran, R.: Amoeba: Robust location privacy scheme for vanet. IEEE JSAC Special Issue on Vehicular Networks 25(8), 1569–1589 (2007)
24. Shanks, D.: Class number, a theory of factorization and genera. In: Symposium on Pure Mathematics, vol. 20, pp. 415–440. AMS (1971)
25. Stanek, M.: Attacking LCCC batch verification of RSA signatures. Cryptology ePrint Archive, Report 2006/111 (2006), http://eprint.iacr.org/2006/111
26. Symington, S., Farrell, S., Weiss, H., Lovell, P.: Bundle security protocol specification. draft-irtf-dtnrg-bundle-security-04 (work in progress) (September 2007)
27. Wagner, D.: The conventional wisdom about sensor network security.. is wrong. In: IEEE Security and Privacy 2005, and invited panelist, Security in Ad-hoc and Sensor Networks 2005 (2005)
28. Yen, S., Laih, C.: Improved digital signature suitable for batch verification. IEEE Transactions on Computers 44(7), 957–959 (1995)
29. Yoon, H., Cheon, J.H., Kim, Y.: Batch verifications with ID-based signatures. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 223–248. Springer, Heidelberg (2005)
30. Zapata, M.G., Asokan, N.: Securing ad hoc routing protocols. In: WiSE 2002: Proceedings of the 1st ACM workshop on Wireless security, pp. 1–10 (2002)

A Auxiliary Algorithms for SPS and PSPS

The algorithms in Section 4 for the SPS and PSPS methods call $Get_0(X)$ to obtain α_0 (and α_0^{-1}) for X, and $Get_1(X)$ to obtain α_1 (and α_1^{-1}) for X. In this section we describe these algorithms for Cha-Cheon signatures. $Get_0(X)$ and $Get_1(X)$ use the following algorithms:

1. $Lowerindex(X)$ – returns the index within the batch B of the message / signature pair in the lowest position in X.
2. $Upperindex(X)$ – returns the index in B of the pair in the highest position in X.

Algorithm. $Get_0(X)$ *(Obtain α_0 (and α_0^{-1}) for Cha-Cheon)*

```
Input: $X$ a list of message / signature pairs.
Output: None.
Return: The value $\alpha_{0,[X]}$ for $X$.
    $P \leftarrow Parent(X)$; $L \leftarrow Left(P)$; $R \leftarrow Right(P)$
    if $(\alpha_{0,[X]})$ then
        return $(\alpha_{0,[X]})$
    elseif $(X = R)$ then
        $\alpha_{0,[R]} \leftarrow \alpha_{0,[P]} \cdot inv\alpha_{0,[L]}$
        $inv\alpha_{0,[R]} \leftarrow inv\alpha_{0,[P]} \cdot \alpha_{0,[L]}$
        return $(\alpha_{0,[R]})$
    elseif $(X = L)$ then
        $l \leftarrow Lowerindex(X)$; $u \leftarrow Upperindex(X)$
        $\alpha_{0,[L]} \leftarrow e(VB_l - VB_{u+1}, S) \cdot e(VD_l - VD_{u+1}, -T)$
        if ( $\alpha_{0,[L]} \neq \alpha_{0,[P]}$) then
            $inv\alpha_{0,[L]} \leftarrow \alpha_{0,[L]}^{-1}$
        else
            $inv\alpha_{0,[L]} \leftarrow inv\alpha_{0,[P]}$
        endif
        return $(\alpha_{0,[L]})$
    else
        $VB_{len(X)} \leftarrow B_{len(X)}$
        $VD_{len(X)} \leftarrow D_{len(X)}$
        for $i = Len(X) - 1$ downto 1 do
            $VB_i \leftarrow VB_{i+1} + B_i$
            $VD_i \leftarrow VD_{i+1} + D_i$
        endfor
        $\alpha_{0,[B]} \leftarrow e(VB_1, S) \cdot e(VD_1, -T)$
        return $(\alpha_{0,[B]})$
    endif
```

When X is a left child, the cost is at most $2 \cdot \mathsf{CstSub}\mathbb{G}_1 + \mathsf{CstDblPair} + \mathsf{CstInv}\mathbb{G}_\mathrm{T}$. If X is a right child, the cost is at most $2\,\mathsf{CstMult}\mathbb{G}_\mathrm{T}$. Since $\mathsf{CstDblPair} \gg \mathsf{CstSub}\mathbb{G}_1$ and $\mathsf{CstDblPair} \gg \mathsf{CstMult}\mathbb{G}_\mathrm{T}$, we estimate the cost of Get_0 for pair of siblings as $\mathsf{CstDblPair} + \mathsf{CstInv}\mathbb{G}_\mathrm{T}$ in Section 5.

Algorithm. $Get_1(X)$ *(Obtain α_1 (and α_1^{-1}) for Cha-Cheon)*

```
Input: X a list of message / signature pairs.
Output: None.
Return: The value $\alpha_{1,[X]}$ for $X$.
    $P \leftarrow Parent(X)$; $L \leftarrow Left(P)$; $R \leftarrow Right(P)$
    if $(\alpha_{1,[X]})$ then
        return $(\alpha_{1,[X]})$
    elseif $(\alpha_{0,[X]} = \alpha_{0,[P]})$ then
        $\alpha_{1,[X]} \leftarrow \alpha_{1,[P]}$
        $inv\alpha_{1,[X]} \leftarrow inv\alpha_{1,[P]}$
        return $(\alpha_{1,[X]})$
    elseif $(X = R)$ then
        $\alpha_{1,[R]} \leftarrow \alpha_{1,[P]} \cdot inv\alpha_{1,[L]}$
        $inv\alpha_{1,[R]} \leftarrow inv\alpha_{1,[P]} \cdot \alpha_{1,[L]}$
        return $(\alpha_{1,[R]})$
    elseif $(X = L)$ then
        $l \leftarrow Lowerindex(X)$; $u \leftarrow Upperindex(X)$
        $WB_{l,u} \leftarrow UB_u - (u \cdot VB_{u+1} + WB_{1,l-1})$
        $WD_{l,u} \leftarrow UD_u - (u \cdot VD_{u+1} + WD_{1,l-1})$
        if $(l \neq 1)$ then
            $WB_{1,u} \leftarrow WB_{1,l-1} + WB_{l,u}$
            $WD_{1,u} \leftarrow WD_{1,l-1} + WD_{l,u}$
        endif
        $\alpha_{1,[L]} \leftarrow e(WB_{l,u}, S) \cdot e(WD_{l,u}, -T)$
        if ( $\alpha_{1,[L]} \neq \alpha_{1,[P]}$) then
            $inv\alpha_{1,[L]} \leftarrow \alpha_{1,[L]}^{-1}$
        else
            $inv\alpha_{1,[L]} \leftarrow inv\alpha_{1,[P]}$
        endif
        return $(\alpha_{1,[L]})$
    else
        $WB_{1,0} \leftarrow \infty$
        $WD_{1,0} \leftarrow \infty$
        $UB_1 \leftarrow VB_1$
        $UD_1 \leftarrow VD_1$
        for $i = 2$ upto $len(X)$ do
            $UB_i \leftarrow UB_{i-1} + VB_i$
            $UD_i \leftarrow UD_{i-1} + VD_i$
        endfor
        $\alpha_{1,[B]} \leftarrow e\left(UB_{len(X)}, S\right) \cdot e\left(UD_{len(X)}, -T\right)$
        return $(\alpha_{1,[B]})$
    endif
```

To compute α_1s and its inverse for a left child node costs no more than $4 \cdot \mathsf{CstAdd}\mathbb{G}_1 + 2 \cdot \mathsf{CstSub}\mathbb{G}_1 + 2 \cdot \mathsf{CstMult}\mathbb{G}_1(\mathrm{t}_1) + \mathsf{CstInv}\mathbb{G}_\mathrm{T} + \mathsf{CstDblPair}$ with $t_1 = \lceil log_2(Len(X)) \rceil < \lceil log_2(N) \rceil$. If X is a right child, the cost is at most $2\,\mathsf{CstMult}\mathbb{G}_\mathrm{T}$.

We estimate the cost of Get_1 for pair of siblings as $\mathsf{CstDblPair} + \mathsf{CstInv}\mathbb{G}_{\mathrm{T}}$ in Section 5.

B $|\,\mathrm{PT}_{(2)}\,| < 2w - 1$

We show that $|\,\mathrm{PT}_{(2)}\,| < 2w-1$ whenever $N = 2^i$ for $i = 1, 2, \ldots$. Let $S_{(2)}(i, w) = |\,\mathrm{PT}_{(2)}\,|$ for $N = 2^i$ and $0 \leq w \leq N$. Note that $S_{(2)}(i, w) = 0$ when $w = 0, 1$, and $S_{(2)}(1, 2) = 1$. Assume that $S_{(2)}(i, w) < 2w - 1$.

For $w = 2$

$$\begin{aligned} S_{(2)}(i+1, 2) &= \sum_{j=0}^{2} \frac{\binom{2^i}{2-j}\binom{2^i}{j}}{\binom{2^{i+1}}{2}} (S_{(2)}(i, 2-j) + S_{(2)}(i, j) + 1) \\ &< \frac{2\binom{2^i}{2}\binom{2^i}{0}}{\binom{2^{i+1}}{2}}(4) + \frac{\binom{2^i}{1}\binom{2^i}{1}}{\binom{2^{i+1}}{2}}(1) \\ &< 2 + \frac{2^i - 2}{2^{i+1} - 1} \\ &< 3. \end{aligned}$$

For $w \geq 3$

$$\begin{aligned} S_{(2)}(i+1, w) &= \sum_{j=0}^{w} \frac{\binom{2^i}{w-j}\binom{2^i}{j}}{\binom{2^{i+1}}{w}} (S_{(2)}(i, w-j) + S_{(2)}(i, j) + 1) \\ &< \frac{2\binom{2^i}{w}\binom{2^i}{0}}{\binom{2^{i+1}}{w}}((2w) + \frac{2\binom{2^i}{w-1}\binom{2^i}{1}}{\binom{2^{i+1}}{w}}(2w-2) + \sum_{j=2}^{w-2} \frac{\binom{2^i}{w-j}\binom{2^i}{j}}{\binom{2^{i+1}}{w}}(2w-1) \\ &< 2w - 1 - \frac{2\binom{2^i}{w-1}}{w\binom{2^{i+1}}{w}} \ \big((2^i + 1)(w-1)\big) \\ &< 2w - 1. \end{aligned}$$

CCA-Secure Proxy Re-encryption without Pairings*

Jun Shao[1,2] and Zhenfu Cao[1,**]

[1] Department of Computer Science and Engineering
Shanghai Jiao Tong University
[2] College of Information Sciences and Technology
Pennsylvania State University
chn.junshao@gmail.com, zfcao@cs.sjtu.edu.cn

Abstract. In a proxy re-encryption scheme, a semi-trusted proxy can transform a ciphertext under Alice's public key into another ciphertext that Bob can decrypt. However, the proxy cannot access the plaintext. Due to its transformation property, proxy re-encryption can be used in many applications, such as encrypted email forwarding. In this paper, by using signature of knowledge and Fijisaki-Okamoto conversion, we propose a proxy re-encryption scheme *without* pairings, in which the proxy can only transform the ciphertext in one direction. The proposal is secure against chosen ciphertext attack (CCA) and collusion attack in the *random oracle model* based on Decisional Diffie-Hellman (DDH) assumption over $\mathbb{Z}^*_{N^2}$ and integer factorization assumption, respectively. To the best of our knowledge, it is the *first* unidirectional PRE scheme with CCA security and collusion-resistance.

Keywords: Unidirectional PRE, DDH, random oracle, CCA security, collusion-resistance.

1 Introduction

In 1998, Blaze, Bleumer, and Strauss [6] proposed the concept of *proxy re-encryption* (PRE), where a semi-trusted proxy can transform a ciphertext for Alice into another ciphertext that Bob can decrypt.[1] However, the proxy cannot

* Supported by Research Fund for the Doctoral Program of Higher Education No. 20060248008, National Natural Science Foundation of China No. 60673079, Special Foundation of Huawei No. YZCB2006001, and National 973 Program No. 2007CB311201.

** Corresponding author.

[1] In almost all related papers, the concept of PRE is introduced as "PRE allows a semi-trusted proxy to convert a ciphertext under Alice's public key to another ciphertext under Bob's public key". However, all existing unidirectional PRE schemes (including ours) do not exactly follow the definition. In particular, in these unidirectional PRE schemes, there are two kinds of ciphertexts, one is the original ciphertext, and the other is the transformed ciphertext. The transformed ciphertext is not exactly as the ciphertext under Bob's public key, but Bob can decrypt the transformed ciphertext only by his secret key. To the best of our knowledge, only the bidirectional schemes in [6,9] satisfy the definition.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 357–376, 2009.

get the plaintext. According to the direction of transformation, PRE schemes can be classified into two types, one is *bidirectional*, i.e., the proxy can transform from Alice to Bob and vice versa; the other is *unidirectional*, i.e., the proxy can only convert in one direction. Blaze *et al.* [6] also gave another method to classify PRE schemes: *multi-use*, i.e., the ciphertext can be transformed from Alice to Bob to Charlie and so on; and *single-use*, i.e., the ciphertext can be transformed only once.

Due to its transformation property, PRE can be used in many applications, including simplification of key distribution [6], key escrow [21], distributed file systems [2,3], security in publish/subscribe systems [23], multicast [10], secure certified email mailing lists [24,22], the DRM of Apple's iTunes [36], interoperable architecture of DRM [34], access control [35], and privacy for public transportation [19]. Recently, Hohenberger *et al.* got a result of securely obfuscating re-encryption [20], which is the first positive result for obfuscating an encryption functionality and against a series of impossibility results [18,16,4].

Since the introduction of PRE by Blaze, Bleumer, and Strauss [6], there have been many papers [6,21,2,3,17,9,11,25] that have proposed different PRE schemes with different security properties. Some of them are related to chosen ciphertext attack (CCA) security. Ivan and Dodis [21] proposed a CCA security model for PRE and a generic construction of single-use PRE in the security model. Nevertheless, their security model allows the delegatee (Bob) to make use of the proxy as an oracle. As a result, the schemes only secure in their security model are not enough for some applications. For example, in encrypted email forwarding, an adversary (Bob) might hope to gain access to the original encrypted email by re-forming it, sending it to the proxy, and then hoping that the proxy responds with, "Can you forward the following to me again? [Encrypted attachment.]"

To fix the problem, Green and Ateniese [17], Canetti and Hohenberger [9] proposed new CCA security models for ID-based PRE and PRE, respectively. In these two new security models, it requires that the proxy checks the validity of the ciphertext before transformation, which is called public verifiability. Following this intuition, the first CCA secure, single-use, unidirectional ID-based PRE scheme in the *random oracle model* and the first CCA secure, multi-use, bidirectional PRE scheme in the *standard model* are proposed in [17,9], respectively. However, the scheme in [17] suffers from the attack in Remark 2. Furthermore, the generic construction of PRE in [21] cannot be proved secure in the CCA security model in [9]. (See Appendix A for details. Hereafter, we refer CCA security to the definition in [9] or Section 2 of this paper.) Chu and Tzeng [11] proposed a multi-use, unidirectional ID-based PRE scheme, and claimed that it was CCA secure in the standard model. However, we showed that it was not true [31], since its transformed ciphertext $(C_{v1}, R, d_1', d_2, d_2')$ can be modified to another well-formed transformed ciphertext $(C_{v1}, R, d_1'F_2(vk)^r, d_2, d_2'g^r)$ by anyone, where r is a random number from $\mathbb{Z}_p^*$. Recently, Libert and Vergnaud [25] proposed a new unidirectional PRE scheme, which is replayable chosen ciphertext attack (RCCA) secure but *not* CCA-secure. It is fair to say that there is no CCA-secure unidirectional PRE scheme.[2] Furthermore, according to the results in [5,29], the timing of a pairing computation is more than twice of that of a modular exponentiation computation. Hence, the CCA-secure unidirectional PRE schemes without pairings are desired.

Another important security notion on unidirectional PRE is collusion-resistance, which disallows Bob and the proxy to collude to reveal Alice's (long term) secret key, but allows the recovery of Alice's "weak" secret key only. In this case, Alice can delegate decryption rights, while keeping signing rights for the same public key. Till now, there are only a few PRE schemes [2,3,25] holding this security.[3]

Though many PRE schemes have been proposed, we find that no unidirectional PRE scheme without pairings but satisfying CCA security and collusion-resistance simultaneously, even in the random oracle model. In this paper, we attempt to propose such a unidirectional PRE scheme.

1.1 Our Contribution

We present a proxy re-encryption scheme *without* pairings, named scheme $\mathfrak{U}$, which is unidirectional and single-use, and proven CCA-secure and collusion resistant in the *random oracle model* based on Decisional Diffie-Hellman (DDH) assumption over $\mathbb{Z}_{N^2}^*$ and integer factorization assumption, respectively. Here, N is a safe-prime modulus.

The difficulty in constructing a CCA secure PRE scheme is to add the *public verifiability* to original ciphertexts. This public verifiability can prevent malicious Bob from gaining some advantage by using the proxy as an oracle. In pairing setting, such as [9], we can use the gap Diffie-Hellman problem (decisional Diffie-Hellman problem is easy, but computational Diffie-Hellman problem is hard) to achieve this. In particular, the gap Diffie-Hellman problem allows us to check whether $\log_g A = \log_h B$. In this paper, we use *signature of knowledge* [8,1] to provide $\log_g A = \log_h B$, hence obtaining public verifiability for original ciphertexts. In fact, using the signature of knowledge to provide public verifiability is due to Shoup and Gennaro [33]. Furthermore, we use Fujisaki-Okamoto conversion [14,15] to provide the validity check of both original ciphertexts and re-encrypted ciphertexts for the decryptor (Alice or Bob).

Following the construction of the public key encryption scheme with double trapdoors in [7], scheme $\mathfrak{U}$ holds collusion-resistance. In particular, the factors of N are the long term secret key, and an exponent is the "weak" secret key,

[2] When we prepared the camera-ready version, we found another paper [13] dealing the similar problems, and getting the similar results with us. In [13], the authors use Schnorr signature [28] to make the original ciphertext be publicly verifiable, while we use signature of knowledge [8,1]. In our submission version, we have a CCA-secure bidirectional PRE scheme, however, the bidirectional one in [13] beats ours in every aspects. Hence, in the current version, we removed our bidirectional one, which can be found in [30]. Furthermore, the unidirectional scheme in [13] suffers from the attack in Remark 2.

[3] The unidirectional PRE scheme in [13] suffers from the collusion attack.

and revealing the exponent does not hurt the secrecy of the factors of N. To the best of our knowledge, scheme $\mathfrak{U}$ is the *first* unidirectional PRE scheme holding CCA security and collusion-resistance simultaneously.

Finally, we extend scheme $\mathfrak{U}$ to scheme $\mathfrak{U}_T$, where the delegator can revoke the proxy's transformation ability. In particular, the proxy can only transform the ciphertext during a restricted time interval.

1.2 Organization

The remaining paper is organized as follows. In Section 2, we review the definitions related to our proposals. In what follows, we present scheme $\mathfrak{U}$ and its security analysis, and scheme $\mathfrak{U}_T$ and its security analysis, in Section 3 and Section 4, respectively. In Section 5 we compare scheme $\mathfrak{U}$ with previous unidirectional PRE schemes. Finally, we conclude the paper in Section 6.

2 Preliminaries

In this section, we briefly review the definitions related to our proposals, some similar content can be found in [8,1,17,9].

2.1 Public Key Encryption

Definition 1 (Public Key Encryption (PKE)). *A public key encryption scheme* PKE *is a triple of PPT algorithms* (KeyGen, Enc, Dec)*:*

- *KeyGen*$(1^k) \to (pk, sk)$*. On input the security parameter* 1^k*, the key generation algorithm KeyGen outputs a public key* pk *and a secret key* sk*.*
- *Enc*$(pk, m) \to C$*. On input a public key* pk *and a message* m *in the message space, the encryption algorithm Enc outputs a ciphertext* C*.*
- *Dec*$(sk, C) \to m$*. On input a secret key* sk *and a ciphertext* C*, the decryption algorithm Dec outputs a message* m *in the message space or* $\perp$*.*

Correctness. The correctness property is that for any message m in the message space and any key pair $(pk, sk) \leftarrow \texttt{KeyGen}(1^k)$. Then the following condition must hold: $\texttt{Dec}(sk, \texttt{Enc}(pk, m)) = m$.

2.2 Unidirectional Proxy Re-encryption

Definition 2 (Unidirectional PRE). *A unidirectional proxy re-encryption scheme* UniPRE *is a tuple of PPT algorithms* (KeyGen, ReKeyGen, Enc, ReEnc, Dec)*:*

- *KeyGen, Enc, Dec: Identical to those in public key encryption.*
- *ReKeyGen*$(sk_1, pk_2) \to rk_{1\to 2}$*. On input a secret key* sk_1 *and a public key* pk_2*, the re-encryption key generation algorithm ReKeyGen outputs a unidirectional re-encryption key* $rk_{1\to 2}$*.*
- *ReEnc*$(rk_{1\to 2}, C_1) \to C_2$*. On input a re-encryption key* $rk_{1\to 2}$ *and a ciphertext* C_1*, the re-encryption algorithm ReEnc outputs a re-encrypted ciphertext* C_2 *or* $\perp$*.*

Correctness. A correct proxy re-encryption scheme should satisfy two requirements:

$$\texttt{Dec}(sk, \texttt{Enc}(pk, m)) = m,$$

and

$$\texttt{Dec}(sk', \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk'), C)) = m,$$

where $(pk, sk), (pk', sk') \leftarrow \texttt{KeyGen}(1^k)$, and C is the ciphertext of message m for pk from algorithm $\texttt{Enc}$ or algorithm $\texttt{ReEnc}$.

Chosen Ciphertext Security for Unidirectional Proxy Re-Encryption. This security note is a modification of replayable chosen ciphertext security in [25], where the corrupted public keys are *not* decided before start of the Uni-PRE-CCA game, and the adversary is *allowed* adaptive corruption of users[4], and proxies between corrupted and uncorrupted users. But unlike [25], we require that one well-formed ciphertext *cannot* be modified (but can be transformed) to be another well-formed ciphertext. In [25], anyone can modify the transformed ciphertext, such that $(C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma) \rightarrow (C_1, {C_2'}^t, {C_2''}^{t^{-1}}, {C_2'''}^t, C_3, C_4, \sigma)$, where t is a random number from $\mathbb{Z}_p$.

Note that this security model is only for single-use scheme.

Phase 1: The adversary $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

- *Public key generation oracle* $\mathcal{O}_{pk}$: On input an index i,[5] the Challenger takes a security parameter k, and responds by running algorithm $\texttt{KeyGen}(1^k)$ to generate a key pair (pk_i, sk_i), gives pk_i to $\mathcal{A}$ and records (pk_i, sk_i) in table T_K.
- *Secret key generation oracle* $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, where pk is from $\mathcal{O}_{pk}$, the Challenger searches pk in table T_K and returns sk.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}$: On input (pk, pk') by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, the Challenger returns the re-encryption key $rk_{pk \rightarrow pk'} = \texttt{ReKeyGen}(sk, pk')$, where sk is the secret key corresponding to pk.
- *Re-encryption oracle* $\mathcal{O}_{re}$: On input (pk, pk', C) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, the re-encrypted ciphertext $C' = \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk'), C)$ is returned by the Challenger, where sk is the secret key corresponding to pk.
- *Decryption oracle* $\mathcal{O}_{dec}$: On input (pk, C), where pk is from $\mathcal{O}_{pk}$, the Challenger returns $\texttt{Dec}(sk, C)$, where sk is the secret key corresponding to pk.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, and a public key pk^* on which it wishes to be challenged. There are three constraints on the public key pk^*, (i) it is from $\mathcal{O}_{pk}$; (ii) it did not appear in any query to $\mathcal{O}_{sk}$ in Phase 1; (iii) if

[4] The security model in [13] does not allow such adaptive corruption.

[5] This index is just used to distinguish different public keys.

$(pk^*, \bigstar)$ did appear in any query to $\mathcal{O}_{rk}$, then $\bigstar$ did not appear in any query to $\mathcal{O}_{sk}$. The Challenger picks a random bit $b \in \{0, 1\}$ and sets $C^* = \mathtt{Enc}(pk^*, m_b)$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ issues more queries $q_{n_1+1}, \cdots, q_n$ where query q_i is one of:

- $\mathcal{O}_{pk}$: The Challenger responds as in Phase 1.
- $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk is from $\mathcal{O}_{pk}$;
  - $pk \neq pk^*$;
  - (pk^*, pk) is not a query to $\mathcal{O}_{rk}$ before;
  - (pk', pk, C') is not a query to $\mathcal{O}_{re}$ before, where (pk', C') is a `derivative`[6] of (pk^*, C^*).
- $\mathcal{O}_{rk}$: On input (pk, pk') by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if $pk = pk^*$, then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{re}$: On input (pk, pk', C) by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if (pk, C) is a derivative of (pk^*, C^*), then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{dec}$: On input (pk, C), if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk is from $\mathcal{O}_{pk}$;
  - (pk, C) is not a derivative of (pk^*, C^*).

These queries may be also asked adaptively.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as a Uni-PRE-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking `UniPRE` as the following function of the

[6] Derivatives of (pk^*, C^*) are defined as follows [9]:
1. (pk^*, C^*) is a derivative of itself.
2. If (pk, C) is a derivative of (pk^*, C^*) and (pk', C') is a derivative of (pk, C), then (pk', C') is a derivative of (pk^*, C^*).
3. If $\mathcal{A}$ has queried $\mathcal{O}_{re}$ on input (pk, pk', C) and obtained (pk', C'), then (pk', C') is a derivative of (pk, C).
4. If $\mathcal{A}$ has queried $\mathcal{O}_{rk}$ on input (pk, pk'), and $C' = \mathtt{ReEnc}(\mathcal{O}_{re}(pk, pk'), C)$, then (pk', C') is a derivative of (pk, C).

security parameter k: $\text{Adv}_{\texttt{UniPRE},\mathcal{A}}(k) = |\Pr[b = b'] - 1/2|$. Using the Uni-PRE-CCA game we can define chosen ciphertext security for unidirectional proxy re-encryption schemes.

Definition 3 (Uni-PRE-CCA security). *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time Uni-PRE-CCA adversary $\mathcal{A}$ the function $Adv_{\texttt{UniPRE},\mathcal{A}}(k)$ is negligible. As shorthand, we say that* `UniPRE` *is Uni-PRE-CCA secure.*

Remark 1. In [25], the authors considered this model as a static corruption model, since it does not capture some scenarios, such as the adversary generate public keys on behalf of corrupted parties. However, we think this model is an *adaptive* corruption model. Since *Adaptive Security* usually refers to the ability of the adversary to choose which parties to corrupt depending on the information gathered so far, *but* the Challenger still generates all parties' key pairs. Allowing adversaries to generate malicious parties' public keys on their own is usually called "chosen-key model" [26].[7]

Besides CCA security, there is another security notion, collusion resistance, for unidirectional PRE schemes.

Definition 4 (Uni-PRE-CR security). [8] *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *is collusion resistant if for any polynomial bounded adversary $\mathcal{A}$, the following probability is negligible:*

$$\begin{aligned} Pr[(sk_1, pk_1) \leftarrow \mathit{KeyGen}(1^k), \{(sk_i, pk_i) \leftarrow \mathit{KeyGen}(1^k)\},\\ \{rk_{i\to 1} \leftarrow \mathit{ReKeyGen}(sk_i, pk_1)\},\\ \{rk_{1\to i} \leftarrow \mathit{ReKeyGen}(sk_1, pk_i)\},\\ i = 2, \cdots,\\ \alpha \rightarrow \mathcal{A}(pk_1, \{pk_i, sk_i\}, \{rk_{1\to i}\}, \{rk_{i\to 1}\}) :\\ \alpha = sk_1]. \end{aligned}$$

Due to its similarity with that of unidirectional PRE schemes, we put the definitions of unidirectional PRE schemes with temporary delegation in the Appendix.

2.3 Signature of Knowledge

In our proposal, we apply the following non-interactive zero-knowledge proof of knowledge, named signature of knowledge of equality of two discrete logarithms [8,1,32].

Definition 5. *Let $y_1, y_2, g, h \in \mathbb{G}$, $\mathbb{G}$ be a cyclic group of quadratic residues modulo N^2 (N is a safe-prime modulus), and $H(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^k$ (k is the security parameter). A pair (c, s), verifying $c = H(y_1||y_2||g||h||g^s y_1^c||h^s y_2^c||m)$ is a signature of knowledge of the discrete logarithm of both $y_1 = g^x$ w.r.t. base g and $y_2 = h^x$ w.r.t. base h, on a message $m \in \{0,1\}^*$.*

[7] We thank an anonymous reviewer of Indocrypt 2008 to point out this.

[8] This security notion is from [2,3], called Master Secret Security.

The party in possession of the secret x is able to compute the signature, provided that $x = \log_g y_1 = \log_h y_2$, by choosing a random $t \in \{0, \cdots, 2^{|N^2|+k} - 1\}$ ($|n|$ is the bit-length of n). And then computing c and s as:

$$c = H(y_1||y_2||g||h||g^t||h^t||m) \text{ and } s = t - cx.$$

We denote $\texttt{SoK.Gen}(y_1, y_2, g, h, m)$ as the generation of the proof.

2.4 Complexity Assumption

The security of our proposal is based on the Decisional Diffie-Hellman assumption (DDH) over $\mathbb{Z}^*_{N^2}$.

DDH Problem. The DDH problem is as follows: Given $\langle g, g^a, g^b \rangle$ for some $a, b \in \mathit{ord}(\mathbb{G})$ and $T \in \mathbb{G}$, decide whether $T = g^{ab}$, where $\mathbb{G}$ is a cyclic group of quadratic residues modulo N^2 (N is a safe-prime modulus), g is a random number of $\mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ε in solving DDH problem if $|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, T) = 0]| \geq \varepsilon$, where the probability is over the random choices of a, b in $\mathit{ord}(\mathbb{G})$, the random choices of g, T in $\mathbb{G}$, and the random bits of $\mathcal{A}$.

Definition 6 (DDH Assumption). *We say that the ε-DDH assumption holds if no PPT algorithm has advantage at least ε in solving the DDH problem.*

Note that the DDH problem over $\mathbb{Z}^*_{N^2}$ is easy if the factors of N is known [7].

2.5 The Public Key Encryption with Double Trapdoors

The basic public key encryption of our proposal is the public key encryption with double trapdoors in [7], named BCP03.

The following description is from [7]. Let $N = pq$ be a safe prime modulus, such that $p = 2p' + 1$, $q = 2q' + 1$, and p, p', q, q' are primes. Assume $\mathbb{G}$ is the cyclic group of quadratic residues modulo N^2, then we have the order of $\mathbb{G}$ is $Np'q'$.

- $\texttt{KeyGen}(1^k) \rightarrow (pk, sk)$. Choose a random element $\alpha \in \mathbb{Z}^*_{N^2}$, a random value $a \in [1, Np'q']$, and set $g = \alpha^2 \mod N^2$ and $h = g^a \mod N^2$. The public key is (N, g, h), and the secret key is a.
- $\texttt{Enc}(pk, m) \rightarrow C$. On input a public key pk and a message $m \in \mathbb{Z}_N$, the ciphertext (A, B) is computed as

$$A = g^r \bmod N^2, \qquad B = h^r(1 + mN) \bmod N^2,$$

  where r is a random number from $\mathbb{Z}_{N^2}$.
- $\texttt{Dec}(sk, C) \rightarrow m$. There are two methods to decrypt.
  - Knowing a, one can compute m by

$$m = \frac{B/(A^a) - 1 \bmod N^2}{N}.$$

- Knowing p', q', one can compute m by

$$m = \frac{D - 1 \bmod N^2}{N} \cdot \pi \bmod N,$$

where $D = \left(\frac{B}{g^{w_1}}\right)^{2p'q'}$, $w_1 = ar \bmod N$, $ar \bmod pqp'q' = w_1 + w_2 N$, π is the inverse of $2p'q' \bmod N$.

Note the values of $a \bmod N$ and $r \bmod N$ can be computed when given $h = g^a \bmod N^2$, $A = g^r \bmod N^2$, and p', q', by the method in [27] (Theorem 1 in [27]).

3 New Unidirectional Proxy Re-encryption Scheme without Pairings

The proposed unidirectional scheme $\mathfrak{U}$ is based on the CPA secure and collusion resistant unidirectional PRE scheme in [2,3] (the first attempt scheme in [2,3]), and with the signature of knowledge [8,1] and Fujisaki-Okamoto conversion [14,15]. The basic public key encryption is scheme BCP03.

The intuition in scheme $\mathfrak{U}$ is as follows. Firstly, since there are two trapdoors (a and the factorization of the modulus) in scheme BCP03, we can use the key sharing technique in [17] to share a. In particular, let $a = r_1 + r_2$, and sent the proxy r_1 and the ciphertext of r_2 under the delegatee's public key. Knowing a cannot hurt the secrecy of the factorization of the modulus, hence, collusion-resistance obtained. Secondly, scheme BCP03 is CPA-secure, hence, we use Fijisaki-Okamoto conversion to make scheme BCP03 be CCA-secure. Thirdly, we use the signature of knowledge to make the original ciphertext be publicly verifiable.

3.1 Scheme $\mathfrak{U}$ with Single-Use

Scheme $\mathfrak{U}$ contains three cryptographic hash functions for all users: $H_1(\cdot) : \{0,1\}^* \to \{0,1\}^{k_1}$, $H_2(\cdot) : \{0,1\}^* \to \{0,1\}^n$, and $H_3(\cdot) : \{0,1\}^* \to \{0,1\}^{k_2}$, where k_1 and k_2 are the security parameter, n is the bit-length of messages to be encrypted. The details are as follows.

`KeyGen`: Choose a safe-prime modulus $N = pq$, three random numbers $\alpha \in \mathbb{Z}^*_{N^2}$, $a, b \in [1, pp'qq']$, a hash function $H(\cdot)$, where $p = 2p' + 1$, $q = 2q' + 1$, p, p', q, q' are primes, and $H(\cdot) : \{0,1\}^* \to \mathbb{Z}_{N^2}$. Furthermore, set $g_0 = \alpha^2 \bmod N^2$, $g_1 = {g_0}^a \bmod N^2$, and $g_2 = {g_0}^b \bmod N^2$. The public key is $pk = (H(\cdot), N, g_0, g_1, g_2)$, the "weak" secret key is $wsk = (a, b)$, and the long term secret key is $sk = (p, q, p', q')$.

`ReKeyGen`: On input a public key $pk_Y = (H_Y(\cdot), N_Y, g_{Y0}, g_{Y1}, g_{Y2})$, a "weak" secret key $wsk_X = a_X$, and a long term secret key $sk_X = (p_X, q_X, p'_X, q'_X)$, it outputs the unidirectional re-encryption key $rk_{X\to Y} = (rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$, where $rk^{(1)}_{X\to Y} = (\dot{A}, \dot{B}, \dot{C})$, and computed as follows:

- Choose two random numbers $\dot{\sigma} \in \mathbb{Z}_N$, $\dot{\beta} \in \{0,1\}^{k_1}$.
- Compute $rk^{(2)}_{X\to Y} = a_X - \dot{\beta} \bmod (p_X q_X p'_X q'_X)$.
- Compute $r_{X\to Y} = H_Y(\dot{\sigma}||\dot{\beta})$, $\dot{A} = (g_{Y0})^{r_{X\to Y}} \bmod (N_Y)^2$, $\dot{C} = H_1(\dot{\sigma}) \oplus \dot{\beta}$,
$$\dot{B} = (g_{Y2})^{r_{X\to Y}} \cdot (1 + \dot{\sigma}N_Y) \bmod (N_Y)^2. \quad (1)$$

`Enc`: On input a public key $pk = (H(\cdot), N, g_0, g_1, g_2)$ and a message $m \in \{0,1\}^n$, the encryptor does the following performances:

- Choose a random number $\sigma \in \mathbb{Z}_N$.
- Compute $r = H(\sigma||m)$, $A = (g_0)^r \bmod N^2$, $C = H_2(\sigma) \oplus m$, $D = (g_2)^r \bmod N^2$,
$$B = (g_1)^r \cdot (1 + \sigma N) \bmod N^2. \quad (2)$$
- Run $(c, s) \leftarrow$ `SoK.Gen`$(A, D, g_0, g_2, (B, C))$, where the underlying hash function is H_3.
- Output the ciphertext $K = (A, B, C, D, c, s)$.

`ReEnc`: On input a re-encryption key $rk_{X\to Y} = (rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ and a ciphertext $K = (A, B, C, D, c, s)$ under key $pk_X = (H_X(\cdot), N_X, g_{X0}, g_{X1}, g_{X2})$, check whether $c = H_3(A||D||g_{X0}||g_{X2}||(g_{X0})^s A^c||(g_{X2})^s D^c||(B||C))$. If not hold, output $\perp$ and terminate; otherwise, re-encrypt the ciphertext to be under key pk_Y as:

- Compute $A' = A^{rk^{(2)}_{X\to Y}} = (g_{X0})^{r(a_X - \dot{\beta})} \bmod (N_X)^2$.
- Output the new ciphertext $(A, A', B, C, rk^{(1)}_{X\to Y}) = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$.

`Dec`: On input a secret key and any ciphertext K, parse $K = (A, B, C, D, c, s)$, or $K = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$.

Case $K = (A, B, C, D, c, s)$**:** Check whether $c = H_3(A||D||g_0||g_2||(g_0)^s A^c|| (g_2)^s D^c||(B||C))$, if not, output $\perp$ and terminate; otherwise,

- if the input secret key is the "weak" secret key a, compute $\sigma = \frac{B/(A^a)-1 \bmod N^2}{N}$.
- if the secret key is the long term secret key (p, q, p', q'), compute $\sigma = \frac{(B/g_0^{w_1})^{2p'q'}-1 \bmod N^2}{N} \cdot \pi(\text{mod} N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $m = C \oplus H_2(\sigma)$, if $B = (g_1)^{H(\sigma||m)} \cdot (1 + \sigma N) \bmod N^2$ holds, output m; otherwise, output $\perp$ and terminate.

Case $K = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$**:** In this case, the decryptor should know the delegator's (Alice's) public key $(H'(\cdot), N', g'_0, g'_1, g'_2)$.

- If the input secret key is the "weak" secret key b, compute $\dot{\sigma} = \frac{\dot{B}/(\dot{A}^b)-1 \bmod N^2}{N}$.
- If the input secret key is the long term secret key (p, q, p', q'), computes $\dot{\sigma} = \frac{(\dot{B}/g_0^{w_1})^{2p'q'}-1 \bmod N^2}{N} \cdot \pi(\text{mod} N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $\dot{\beta} = \dot{C} \oplus H_1(\dot{\sigma})$, if $\dot{B} = (g_1)^{H(\dot{\sigma}||\dot{\beta})} \cdot (1 + \dot{\sigma}N) \bmod N^2$ holds, then compute $\sigma = \frac{B/(A' \cdot A^{\dot{\beta}})-1 \bmod N'^2}{N'}$, $m = C \oplus H_2(\sigma)$; otherwise, output $\perp$ and terminate. If $B = (g'_1)^{H'(\sigma||m)} \cdot (1 + \sigma N') \bmod N'^2$ holds, then output m; otherwise, output $\perp$ and terminate.

Note that $(H(\cdot), N, g_0, g_1, g_2)$ is the public key of the decryptor.

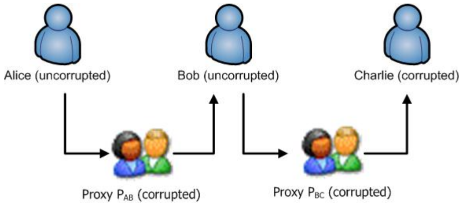

Alice (uncorrupted)
Bob (uncorrupted)
Charlie (corrupted)
Proxy P_{AB} (corrupted)
Proxy P_{BC} (corrupted)

Fig. 1. An example of delegation relationship

Remark 2. The values of $\dot{B}$ and B are computed differently, in particular, in equation (1), the base is g_1, while in equation (2), the base is g_2. This difference aims to resist the following attack: Assume that there is the delegation relationship as in Fig. 1. Alice delegates her decryption rights to Bob via the proxy P_{AB}, and Bob delegates his decryption rights to Charlie via the proxy P_{BC}. Alice and Bob are uncorrupted, the rest parties are corrupted, and the target (challenged) user is Alice. This corruption situation is allowed in the security model in Section 2 (Note that the attacked scheme should be single-use). If the bases in equations (1) and (2) are both g_1, then the adversary can decrypt any ciphertext for Alice as follows. The proxy P_{BC} and Charlie colludes to get Bob's weak secret key a_B, and then they collude with the proxy P_{AB} to get Alice's weak secret key a_A. As a result, the adversary can use a_A to decrypt any ciphertext for Alice. However, in scheme $\mathfrak{U}$, the proxy P_{BC} and Charlie cannot get Bob's weak secret key b_B (which is for decrypting partial re-encryption key), hence, they cannot collude with the proxy P_{AB} to get Alice's weak secret key a_A (which is for decrypting ciphertexts).

Note that the above attack is also allowed in the security model in [17,13], since they only disallow the adversary to corrupt the proxy between the target user and the uncorrupted user. The unidirectional schemes in [17,13] suffer from the above attack. To resist the above attack, we can use the same method in scheme $\mathfrak{U}$, in particular, every user has two public/secret key pairs, one is for decrypting ciphertexts of messages, and the other is for decrypting the partial re-encryption key.

Correctness. The correctness property is easily obtained by the correctness of scheme BCP03 [7] and Fujisaki-Okamoto conversion [14,15].

Due to the limited space, we omit the proofs of the following two theorems here, and give them in the full version.

Theorem 1 (Uni-PRE-CCA security). *In the random oracle model, scheme $\mathfrak{U}$ is CCA-secure under the assumptions that DDH problem over $\mathbb{Z}^*_{N^2}$ is hard, and the signature of knowledge is secure.*

Theorem 2 (Uni-PRE-CR security). *In the random oracle, if N is hard to factor, then scheme $\mathfrak{U}$ is collusion resistant.*

4 Scheme $\mathfrak{U}_T$ with Temporary Delegation

This section describes scheme $\mathfrak{U}_T$, a variant of scheme $\mathfrak{U}$, supporting temporary delegation. Like the temporary unidirectional PRE schemes in [2,3,25], the proxy is only allowed to transform ciphertexts from the delegator to the delegatee during a limited time period. The point of modifying scheme $\mathfrak{U}$ to scheme $\mathfrak{U}_T$ is to make different g_1's for every time period.

Scheme $\mathfrak{U}_T$ also contains three cryptographic hash functions for all users: $H_1(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$, $H_2(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^n$, and $H_3(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^{k_2}$, where k_1 and k_2 are the security parameter, n is the bit-length of messages to be encrypted. The details are as follows.

`KeyGen`: Choose a safe-prime modulus $N = pq$, $T+2$ random numbers $\alpha \in \mathbb{Z}^*_{N^2}$, $a_1, \cdots, a_T, b \in [1, pp'qq']$, a hash function $H(\cdot)$, where $p = 2p'+1$, $q = 2q'+1$, p, p', q, q' are primes, T is the number of time intervals, and $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{Z}_{N^2}$. Furthermore, set $g_0 = \alpha^2 \bmod N^2$, $g_1^{(i)} = {g_0}^{a_i} \bmod N^2$ $(i = 1, \cdots, T)$, and $g_2 = {g_0}^b \bmod N^2$. The public key is $pk = (H(\cdot), N, g_0, g_1^{(i)}\ (i = 1, \cdots, T), g_2)$, the "weak" secret key is $(a_i\ (i = 1, \cdots, T), b)$, and the long-term secret key is $sk = (p, q, p', q')$.

`ReKeyGen`: On input a public key $pk_Y = (H_Y(\cdot), N_Y, g_{Y0}, g_{Y1}^{(1)}, \cdots, g_{Y1}^{(T_Y)}, g_{Y2})$, a "weak" secret key $a_{X,j}$ for time period $j \in \{1, \cdots, T_X\}$, and a secret key $sk_X = (p_X, q_X, p'_X, q'_X)$, it outputs the unidirectional re-encryption key $rk_{X \rightarrow Y,j} = (rk^{(1)}_{X \rightarrow Y,j}, rk^{(2)}_{X \rightarrow Y,j})$ for the j-th time period, which is generated as follows.

- Choose two random numbers $\dot{\sigma}_j \in \mathbb{Z}_N$, $\dot{\beta}_j \in \{0,1\}^{k_1}$.
- Compute $rk^{(2)}_{X \rightarrow Y,j} = a_{X,j} - \dot{\beta}_j \bmod (p_X q_X p'_X q'_X)$.
- Compute

$$r_{X \rightarrow Y,j} = H_Y(\dot{\sigma}_j || \dot{\beta}_j), \dot{A}_j = (g_{Y0})^{r_{X \rightarrow Y,j}} \bmod (N_Y)^2,$$
$$\dot{B}_j = (g_{Y2})^{r_{X \rightarrow Y,j}} \cdot (1 + \dot{\sigma}_j N_Y) \bmod (N_Y)^2, \quad \dot{C}_j = H_1(\dot{\sigma}_j) \oplus \dot{\beta}_j$$

- Set $rk^{(1)}_{X \rightarrow Y,j} = (\dot{A}_j, \dot{B}_j, \dot{C}_j)$.

`Enc`: On input a public key $pk = (H(\cdot), N, g_0, g_1^{(1)}, \cdots, g_1^{(T)}, g_2)$, a time period $j \in \{1, \cdots, T\}$ and a message $m \in \{0,1\}^n$, the encryptor does the following performances:

- Choose a random number $\sigma_j \in \mathbb{Z}_N$.
- Compute

$$r_j = H(\sigma_j || m),$$
$$A_j = (g_0)^r \bmod N^2, \quad B_j = (g_1^{(j)})^r \cdot (1 + \sigma_j N) \bmod N^2,$$
$$C_j = H_2(\sigma_j) \oplus m, \quad D_j = (g_2)^r \bmod N^2.$$

- Run $(c_j, s_j) \leftarrow \texttt{SoK.Gen}(A_j, D_j, g_0, g_2, (B_j, C_j))$, where the underlying hash function is H_3.
- Output the ciphertext $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$ for the j-th time period.

`ReEnc`: On input a re-encryption key $rk_{X\to Y,j} = (rk^{(1)}_{X\to Y,j}, rk^{(2)}_{X\to Y,j})$ and a ciphertext $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$ under key $pk_X = (H_X(\cdot), N_X, g_{X0}, g^{(1)}_{X1}, \cdots, g^{(T_X)}_{X1}, g_{X2})$, where $j \in \{1, \cdots, T_X\}$, the proxy checks whether $c_j = H_3(A_j||D_j||g_{X0}||g_{X2}||(g_{X0})^{s_j}(A_j)^{c_j}||(g_{X2})^{s_j}(D_j)^{c_j}||(B_j||C_j))$. If not hold, output $\perp$ and terminate; otherwise, re-encrypt the ciphertext to be under key pk_Y as:

- Compute $A'_j = (A_j)^{rk^{(2)}_{X\to Y,j}} = (g_{X0})^{r(a_{X,j}-\dot{\beta}_j)} \bmod (N_X)^2$.
- Output the new ciphertext

$$(A_j, A'_j, B_j, C_j, rk^{(1)}_{X\to Y,j}) = (A_j, A'_j, B_j, C_j, \dot{A}_j, \dot{B}_j, \dot{C}_j).$$

`Dec`: On input a secret key and any ciphertext K_j for the j-th time period, where $j \in \{1, \cdots, T\}$, the decryptor parses $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$, or $K_j = (A_j, A'_j, B_j, C_j, \dot{A}_j, \dot{B}_j, \dot{C}_j)$.

Case $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$: Check whether $c_j = H_3(A_j||D_j||g_0||g_2||(g_0)^{s_j}(A_j)^{c_j}||(g_2)^{s_j}{D_j}^{c_j}||(B_j||C_j))$, if not, output $\perp$ and terminate; otherwise,

- if the input secret key is the "weak" secret key a_j, compute $\sigma_j = \frac{B_j/((A_j)^{a_j})-1 \bmod N^2}{N}$.
- if the secret key is the long term secret key (p, q, p', q'), compute $\sigma_j = \frac{(B_j/(g_0)^{w_1})^{2p'q'}-1 \bmod N^2}{N} \cdot \pi(\bmod N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $m = C_j \oplus H_2(\sigma_j)$, if $B_j = (g_1)^{H(\sigma_j||m)} \cdot (1 + \sigma_j N) \bmod N^2$ holds, output m; otherwise, output $\perp$ and terminate.

Case $K_j = (A_j, A'_j, B_j, C_j, \dot{A}_j, \dot{B}_j, \dot{C}_j)$: In this case, the decryptor should know the delegator's (Alice's) public key $(H'(\cdot), N', g'_0, {g^{(i)}_1}', \cdots, {g^{(T')}_1}', g'_2)$.

- If the input secret key is the "weak" secret key b, compute $\dot{\sigma}_j = \frac{\dot{B}_j/((\dot{A}_j)^b)-1 \bmod N^2}{N}$.
- If the input secret key is the long term secret key (p, q, p', q'), computes $\dot{\sigma}_j = \frac{(\dot{B}_j/g_0^{w_1})^{2p'q'}-1 \bmod N^2}{N} \cdot \pi(\bmod N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $\dot{\beta}_j = \dot{C}_j \oplus H_1(\dot{\sigma}_j)$, if $\dot{B}_j = (g_1)^{H(\dot{\sigma}_j||\dot{\beta}_j)} \cdot (1 + \dot{\sigma}_j N) \bmod N^2$ holds, then compute $\sigma_j = \frac{B_j/(A'_j \cdot (A_j)^{\beta_j})-1 \bmod N'^2}{N'}$, $m = C_j \oplus H_2(\sigma_j)$; otherwise, output $\perp$ and terminate. If $B_j = ({g^{(j)}_1}')^{H'(\sigma_j||m)} \cdot (1 + \sigma_j N') \bmod N'^2$ holds, then output m; otherwise, output $\perp$ and terminate.

Note that $(H(\cdot), N, g_0, g^{(i)}_1, \cdots, g^{(T)}_1), g_2)$ is the public key of the decryptor.

Correctness. The correctness property is easily obtained by the same method for scheme $\mathfrak{U}$.

Due to the limited space, we omit the proofs of the following two theorems here, and give them in the full version.

Theorem 3 (Uni-PRETD-CCA security). *In the random oracle model, scheme $\mathfrak{U}_T$ is CCA-secure under the assumptions that DDH problem over $\mathbb{Z}_{N^2}^*$ is hard, and the signature of knowledge is secure.*

Theorem 4 (Uni-PRETD-CR security). *In the random oracle, if N is hard to factor, then scheme $\mathfrak{U}_T$ is collusion resistant.*

5 Comparison

In this section, we compare scheme $\mathfrak{U}$ with the previous CCA-secure unidirectional PRE schemes. Since as mentioned above, the unidirectional PRE schemes in [21,17,11,13] are not CCA-secure, we only compare scheme $\mathfrak{U}$ with the scheme in [25] (named LV08).

In Table 1, we denote t_p, t_{eb}, t_{eN}, t_s, and t_v as the computational cost of a bilinear pairings, an exponentiation over a bilinear group, an exponentiation over $\mathbb{Z}_{N^2}^*$ (N is a safe-prime modulus), a one-time signature and verification, respectively. $\mathbb{G}_e$ and $\mathbb{G}_T$ are the bilinear groups used in scheme LV08. N_X and N_Y are the safe-prime modulus corresponding to the delegator and the delegatee, respectively. svk and σ are the one-time signature's public key and signature. Note that we only consider the case of using weak secret key to decrypt in `Dec` algorithm of scheme $\mathfrak{U}$.

From Table 1, we can see that scheme LV08 is a little bit more efficient than scheme $\mathfrak{U}$. In order to guarantee that N is hard to factor, N should be 1024-bit at least, which makes scheme $\mathfrak{U}$ need more time for an exponentiation and more storage for a ciphertext. However, we emphasize that scheme $\mathfrak{U}$ is CCA-secure and based on the well-studied DDH assumption, while scheme LV08 is RCCA-secure and based on the less-studied 3-quotient decision Bilinear Diffie-Hellman (3-QDBDH) assumption.

Table 1. Comparison between scheme $\mathfrak{U}$ and scheme LV08

Schemes			LV08	$\mathfrak{U}$
Comput. Cost	`ReKeyGen`		$2t_{eb}$	$2t_{eN}$
	`Enc`		$3.5t_{eb} + 1t_s$	$5t_{eN}$
	`ReEnc`		$2t_p + 4t_{eb} + 1t_v$	$4t_{eN}$
	`Dec`	Original	$3t_p + 2t_{eb} + 1t_v$	$5t_{eN}$
		Transformed	$5t_p + 2t_{eb} + 1t_v$	$4t_{eN}$
Ciphertext Size	Original		$1\|svk\| + 2\|\mathbb{G}_e\| + 1\|\mathbb{G}_T\| + 1\|\sigma\|$	$2k + 3\|{N_X}^2\| + \|m\|$
	Transformed		$1\|svk\| + 4\|\mathbb{G}_e\| + 1\|\mathbb{G}_T\| + 1\|\sigma\|$	$k_1 + 3\|{N_X}^2\| + 2\|{N_Y}^2\| + \|m\|$
Security	Security Level		collusion resistant, RCCA	collusion resistant, CCA
	Standard model		Yes	No
	Underlying Assumptions		3-QDBDH	DDH

6 Conclusions

In this paper, by using signature of knowledge and Fijisaki-Okamoto conversion, we proposed the *first* CCA-secure and collusion resistant unidirectional PRE scheme without pairings, which solves a problem proposed in [9,25].

There are still many open problems to be solved, such as designing more efficient CCA-secure, collusion resistant unidirectional PRE schemes without pairings, and CCA-secure multi-use unidirectional PRE schemes [9,25].

Acknowledgements

We thank PKC 2009 chair Stanislaw Jarecki and the anonymous reviewers of PKC 2009 for insightful comments and helpful suggestions.

References

1. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
2. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: Internet Society (ISOC): NDSS 2005, pp. 29–43 (2005)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security (TISSEC) 9(1), 1–30 (2006)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
5. Lynn, H.Y., Scott, B., Barreto, M., Kim, P.S.L.M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–369. Springer, Heidelberg (2002)
6. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
7. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003)
8. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Sommer, G., Daniilidis, K., Pauli, J. (eds.) CAIP 1997. LNCS, vol. 1296, pp. 410–424. Springer, Heidelberg (1997)
9. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: ACM CCS, 2007. Full version: Cryptology ePrint Archieve: Report 2007/171 (2007)
10. Chiu, Y.-P., Lei, C.-L., Huang, C.-Y.: Secure multicast using proxy encryption. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 280–290. Springer, Heidelberg (2005)
11. Chu, C., Tzeng, W.: Identity-based proxy re-encryption without random oracles. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 189–202. Springer, Heidelberg (2007)

12. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
13. Deng, R.H., Weng, J., Liu, S., Chen, K.: Chosen-ciphertext secure proxy re-encryption schemes without pairings. In: CANS 2008. LNCS, vol. 5339, pp. 1–17. springer, Heidelberg (2008), `http://eprint.iacr.org/2008/509`
14. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)
15. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
16. Goldwasser, S., Kalai, Y.T.: On the impossibility of obfuscation with auxiliary input. In: FOCS 2005, pp. 553–562 (2005)
17. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007); full version: Cryptology ePrint Archieve: Report 2006/473
18. Hada, S.: Zero-knowledge and code obfuscation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 443–457. Springer, Heidelberg (2000)
19. Heydt-Benjamin, T.S., Chae, H., Defend, B., Fu, K.: Privacy for public transportation. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 1–19. Springer, Heidelberg (2006)
20. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely obfuscating re-encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 233–252. Springer, Heidelberg (2007)
21. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: Internet Society (ISOC): NDSS 2003 (2003)
22. Khurana, H., Hahm, H.-S.: Certified mailing lists. In: ASIACCS 2006, pp. 46–58 (2006)
23. Khurana, H., Koleva, R.: Scalable security and accounting services for content-based publish subscribe systems. International Journal of E-Business Research 2(3) (2006)
24. Khurana, H., Slagell, A., Bonilla, R.: Sels: A secure e-mail list service. In: ACM SAC 2005, pp. 306–313 (2005)
25. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
26. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential Aggregate Signatures from Trapdoor Permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
27. Paillier, P.: Public-key cryptosystems based on discrete logarithms residues. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
28. Schnorr, C.P.: Efficient identifications and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–251. Springer, Heidelberg (1990)
29. Scott, M.: Computing the tate pairing. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
30. Shao, J.: Proxy re-cryptography, revisited, PhD Thesis, Shanghai Jiao Tong University (December, 2007)
31. Shao, J., Xing, D., Cao, Z.: Analysis of cca secure unidirctional id-based pre scheme. Technical Report of TDT (2008)

32. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
33. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)
34. G. Taban, A.A. Cárdenas, and V.D. Gligor. Towards a secure and interoperable drm architecture. In: ACM DRM 2006, pp. 69–78 (2006)
35. Talmy, A., Dobzinski, O.: Abuse freedom in access control schemes. In: AINA 2006, pp. 77–86 (2006)
36. Smith, T.: Dvd jon: buy drm-less tracks from apple itunes (2005), `http://www.theregister.co.uk/2005/03/18/itunes_pymusique`

A Analysis on Ivan-Dodis Construction

A.1 Ivan-Dodis Construction

The Ivan-Dodis construction is based on any CCA-secure PKE. The details are as follows.

UniPRE.KGen: On input the security parameter 1^k, it outputs two key pairs (pk_1, sk_1) and (pk_2, sk_2).

UniPRE.RKGen: On input the delegator's key pairs (pk_1, sk_1) and (pk_2, sk_2), the delegator sends sk_1 as the re-encryption key to the proxy via a secure channel, and sends sk_2 to the delegatee as the partial key via another secure channel.

UniPRE.Enc: On input public keys (pk_1, pk_2) and a message m, it outputs $\mathtt{PKE.Enc}(pk_1, \mathtt{PKE.Enc}(pk_2, m))$.

UniPRE.ReEnc: On input a re-encryption key sk_1 and a ciphertext C, it outputs a re-encrypted ciphertext $C' = \mathtt{PKE.Dec}(sk_1, C)$.

UniPRE.Dec: On input secret keys (sk_1, sk_2), a partial key sk_2' from its delegator and a ciphertext C, UniPRE.Dec does:

- If C is an original ciphertext, then it outputs $\mathtt{PKE.Dec}(sk_2, \mathtt{PKE.Dec}(sk_1, C))$.
- If C is a re-encrypted ciphertext, then it outputs $\mathtt{PKE.Dec}(sk_2', C)$.

Note that the partial key sk_2 can be encrypted by the delegatee's public key, and forwarded to Bob by the proxy. In this case, the delegatee does not require to store extra secrets for every delegation [2,3].

A.2 Chosen Ciphertext Attacks on the Ivan-Dodis Construction

In this subsection, we will show that the adversary always wins the Uni-PRE-CCA game with the Ivan-Dodis construction's Challenger.

Phase 1: The adversary does not need to make any query in this phase.

Challenge: The adversary outputs two equal length plaintexts m_0, m_1 from the message space, and an uncorrupted public key $pk^* = (pk_1^*, pk_2^*)$.

The Challenger will follow the Uni-PRE-CCA game's specification, i.e., pick a random bit $b \in \{0, 1\}$ and sets $C^* = \texttt{UniPRE.Enc}(pk^*, m_b)$. It sends C^* as the challenge ciphertext to $\mathcal{A}$.

Phase 2: The adversary performs as follows.

1. The adversary queries $\mathcal{O}_{re}$ with (pk^*, pk, C^*), such that pk is uncorrupted. Then as the Uni-PRE-CCA game's specification, the adversary can get the re-encrypted ciphertext C' such that $C' = \texttt{PKE.Dec}(sk_1^*, C^*)$, sk_1^* is the key corresponding to pk_1^*.
2. The adversary computes $\hat{C} = \texttt{PKE.Enc}(pk_1^*, C')$. Note that $\hat{C} \neq C^*$ since `PKE` is CCA-secure, such as the underlying PKE scheme is the Cramer-Shoup scheme [12].
3. The adversary queries $\mathcal{O}_{de}$ with $(pk^*, \hat{C})$ and gets a message m. Note that $(pk^*, \hat{C})$ is not a derivative of (pk^*, C^*), hence this query is valid.

Guess: If $m = m_0$, the adversary $\mathcal{A}$ outputs $b' = 0$; otherwise, output $b' = 1$.

Since $\hat{C}$ and C^* are corresponding to the same message, we always have $b = b'$. As a result, the Ivan-Dodis construction is not CCA-secure for the security model in Section 2.

B Definitions of Unidirectional PRE Schemes with Temporary Delegation

Definition 7 (Unidirectional `PRE` with Temporary Delegation). *A unidirectional proxy re-encryption scheme* `UniPRE` *with temporary delegation is a tuple of PPT algorithms* (`KeyGen, ReKeyGen, Enc, ReEnc, Dec`)*:*

- $\texttt{KeyGen}(1^k) \rightarrow (pk, sk, T)$. On input the security parameter 1^k, the key generation algorithm `KeyGen` outputs a public/secret key pair (pk, sk), and the number of time intervals T.
- $\texttt{Enc}(pk, m, j) \rightarrow C_j$. On input a public key pk, a message m in the message space, and the time period $j \in \{1, \cdots, T\}$, the encryption algorithm `Enc` outputs a ciphertext C_j for the j-th time period.
- $\texttt{ReKeyGen}(sk_1, pk_2, j) \rightarrow rk_{1\rightarrow 2,j}$. On input a secret key sk_1, a public key pk_2, and the time period $j \in \{1, \cdots, T_1\}$, where T_1 is the number of time intervals corresponding to the delegator. The re-encryption key generation algorithm `ReKeyGen` outputs a unidirectional re-encryption key $rk_{1\rightarrow 2,j}$ for the j-th time period.
- $\texttt{ReEnc}(rk_{1\rightarrow 2,j}, C_1^{(j)}) \rightarrow C_2^{(j)}$. On input a re-encryption key $rk_{1\rightarrow 2,}$ and a ciphertext $C_1^{(j)}$ for the j-th time period, where $j \in \{1, \cdots, T_1\}$, T_1 is the number of time intervals corresponding to the delegator. The re-encryption algorithm `ReEnc` outputs a re-encrypted ciphertext $C_2^{(j)}$ for the j-th time period or $\perp$.

- $\texttt{Dec}(sk, C_j) \rightarrow m$. On input a secret key sk and a ciphertext C_j for the j-th time period, where $j \in \{1, \cdots, T\}$, T is the number of time intervals corresponding to the decryptor. The decryption algorithm $\texttt{Dec}$ outputs a message m in the message space or $\perp$.

Correctness. A correct proxy re-encryption scheme should satisfy two requirements: $\texttt{Dec}(sk, \texttt{Enc}(pk, m, j)) = m$, and $\texttt{Dec}(sk', \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk', j), C_j)) = m$, where $(pk, sk, T), (pk', sk', T') \leftarrow \texttt{KeyGen}(1^k)$, C_j is the ciphertext of message m for pk and the j-th time period from algorithm $\texttt{Enc}$ or algorithm $\texttt{ReEnc}$, and $j \in \{1, \cdots, T\}$.

Chosen Ciphertext Security for Unidirectional Proxy Re-Encryption with Temporary Delegation. Following the method in [25], we extend Uni-PRE-CCA game to Uni-PRETD-CCA game, which is described as follows.

Phase 1: The adversary $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

- $\mathcal{O}_{pk}$, $\mathcal{O}_{sk}$: Identical to those Uni-PRE-CCA game.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}$: On input (pk, pk', j) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, and the time period $j \in \{1, \cdots, T\}$, the Challenger returns the re-encryption key $rk_{pk \rightarrow pk', j} = \texttt{ReKeyGen}(sk, pk', j)$, where sk is the secret key corresponding to pk.
- *Re-encryption oracle* $\mathcal{O}_{re}$: On input (pk, pk', C, j) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, and the time period $j \in \{1, \cdots, T\}$, the re-encrypted ciphertext $C' = \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk', j), C)$ is returned by the Challenger, where sk is the secret key corresponding to pk.
- *Decryption oracle* $\mathcal{O}_{dec}$: On input (pk, C, j), where pk is from $\mathcal{O}_{pk}$, and the time period $j \in \{1, \cdots, T\}$, the Challenger returns $\texttt{Dec}(sk, C)$, where sk is the secret key corresponding to pk.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, a public key pk^*, and the time period j^* on which it wishes to be challenged. There are some constraints on the public key pk^* and j^*: (i) pk^* is from $\mathcal{O}_{pk}$; (ii) pk^* did not appear in any query to $\mathcal{O}_{sk}$ in Phase 1; (iii) if $(pk^*, \bigstar, j^*)$ did appear in any query to $\mathcal{O}_{rk}$, then $\bigstar$ did not appear in any query to $\mathcal{O}_{sk}$. The Challenger picks a random bit $b \in \{0, 1\}$ and sets $C^* = \texttt{Enc}(pk^*, m_b, j)$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ issues more queries $q_{n_1+1}, \cdots, q_n$ where query q_i is one of:

- $\mathcal{O}_{pk}$: The Challenger responds as in Phase 1.
- $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
    - pk is from $\mathcal{O}_{pk}$;

  - $pk \neq pk^*$;
  - (pk^*, pk, j^*) is not a query to $\mathcal{O}_{rk}$ before;
  - (pk', pk, C', j^*) is not a query to $\mathcal{O}_{re}$ before, where (pk', C', j^*) is a `derivative`[9] of (pk^*, C^*, j^*).
- $\mathcal{O}_{rk}$: On input (pk, pk', j) by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if $pk = pk^*$ and $j = j^*$, then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{re}$: On input (pk, pk', C, j) by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if (pk, C, j) is a derivative of (pk^*, C^*, j^*), then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{dec}$: On input (pk, C, j), if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk is from $\mathcal{O}_{pk}$;
  - (pk, C, j) is not a derivative of (pk^*, C^*, j^*).

These queries may be also asked adaptively.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as a Uni-PRETD-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking `UniPRE` as the following function of the security parameter k: $\mathrm{Adv}_{\texttt{UniPRE},\mathcal{A}}(k) = |\Pr[b = b'] - 1/2|$. Using the Uni-PRE-CCA game we can define chosen ciphertext security for unidirectional proxy re-encryption schemes.

Definition 8 (Uni-PRETD-CCA security). *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *with temporary delegation is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time Uni-PRETD-CCA adversary $\mathcal{A}$ the function $Adv_{\texttt{UniPRE},\mathcal{A}}(k)$ is negligible. As shorthand, we say that* `UniPRE` *is Uni-PRETD-CCA secure.*

Definition 9 (Uni-PRETD-CR security). *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *with temporary delegation is collusion resistant if for any polynomial bounded adversary $\mathcal{A}$, the following probability is negligible:*

$$\begin{aligned} Pr[(sk_1, pk_1, T_1) \leftarrow \textit{KeyGen}(1^k), \{(sk_i, pk_i, T_i) \leftarrow \textit{KeyGen}(1^k)\}, \\ \{rk_{i\rightarrow 1,j} \leftarrow \textit{ReKeyGen}(sk_i, pk_1, j)\}\ (j = 1, \cdots, T_i), \\ \{rk_{1\rightarrow i,j} \leftarrow \textit{ReKeyGen}(sk_1, pk_i, j)\}\ (j = 1, \cdots, T_1), \\ i = 2, \cdots, \\ \alpha \rightarrow \mathcal{A}(pk_1, \{pk_i, sk_i\}, \{rk_{1\rightarrow i,j}\}, \{rk_{i\rightarrow 1,j}\}) : \\ \alpha = sk_1]. \end{aligned}$$

[9] Derivatives of (pk^*, C^*, j^*) are defined similarly with that in Section 2.2, and just add j^* into every input/output.

Compact CCA-Secure Encryption for Messages of Arbitrary Length

Masayuki Abe[1], Eike Kiltz[2], and Tatsuaki Okamoto[1]

[1] Information Sharing Platform Laboratories
NTT Corporation, Japan
{abe.masayuki,okamoto.tatsuaki}@lab.ntt.co.jp
[2] CWI Amsterdam, The Netherlands
kiltz@cwi.nl

Abstract. This paper proposes a chosen-ciphertext secure variant of the ElGamal public-key encryption scheme which generates very compact ciphertexts for messages of *arbitrary length.* The ciphertext overhead (i.e., the difference between ciphertext and plaintext) is one group element only. Such a property is particularly useful when encrypting short messages such as a PIN or a credit card number in bandwidth-critical environments. On top of the compact overhead, the computational cost for encryption and decryption are almost the same as plain ElGamal encryption. The security is proven based on the strong Diffie-Hellman assumption in the random oracle model.

1 Introduction

One of the most fundamental and best studied public-key encryption schemes is the ElGamal encryption scheme [13] that works over a group $\mathbb{G}$ of prime-order q with a generator g. The public-key is $h = g^x$ for a random secret index $x \in \mathbb{Z}_q$ and a ciphertext for plaintext m consists of the tuple (u, c), where $u = g^r$ and $c = m \oplus H(h^r)$. Here $H : \mathbb{G} \to \{0,1\}^{|m|}$ is some public hash function. Due to its simplicity the scheme is very efficient:

- its constituting encryption and decryption algorithms are roughly as efficient as computing one discrete exponentiation;
- its ciphertext overhead (i.e., the size of the ciphertext minus the size of the plaintext message [23]) consists of only one group element — independently of the length of the plaintext message.

However, the major drawback of the ElGamal scheme is its relatively poor security. The scheme can be proved semantically secure only against *passive attacks* and obviously insecure against stronger *active attacks*, i.e., chosen-ciphertext attacks which is nowadays considered to be the standard security notion. In this work we are interested in the question if it is at all possible to "upgrade" the security properties of the ElGamal encryption scheme to chosen-ciphertext security while at the same time retaining its high computational efficiency and compact ciphertexts.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 377–392, 2009.

1.1 Our Contribution

We propose the most efficient IND-CCA secure ElGamal type encryption scheme whose ciphertext overhead consists of only one group element irrelevant to the length of the plaintext. Such a compact ciphertext overhead with IND-CCA security has been possible only when the message is sufficiently long. (See next section for details and Table 1 for summary.) On the other hand, important messages can be very short; A PIN number typically consists of 4 digits. And such a short information would be often used in applications that have sharp limitations in bandwidth or storage where saving even small amount of bits is critical. All known schemes however can't encrypt short messages securely or require extra overhead to provide chosen cipehrtext security.

In our scheme, the computational efficiency and the size of public/private keys is the same as hashed ElGamal's. It can be instantiated with any passively secure symmetric cipher, such as the one-time pad. Our scheme can be used as a plug-in alternative to ElGamal encryption that brings higher level of security. They share the same key generation algorithm, hash functions, and symmetric cipher. Furthermore, ciphertexts have the same algebraic structure.

IND-CCA-security of our scheme is proven in the random oracle model assuming the hardness of the strong Diffie-Hellman problem [1, 21].[1] The security reduction is almost tight, i.e., the respective advantages are related within a factor of $\log_2 \lambda$, where λ is the security parameter. Even though the reduction is not exactly tight, it only loses $\log_2 \lambda \approx 6$ bits of security for a security level of $\lambda = 80$ bits. Adapting the twinning technique from [9] we furthermore propose a variant of our scheme that can be proved secure under the Computational Diffie-Hellman (CDH) assumption.

1.2 Related Work

We now revisit the known frameworks to immunize the ElGamal scheme secure against chosen-ciphertext attacks, specifically focusing on the above properties of efficiency and ciphertext overhead, and point out their respective shortcomings. Here we mostly restrict ourselves to schemes that are secure in the random oracle model [4]. For quick overview, see Table 1. For an overview concerning schemes based on *trapdoor permutations* (such as OAEP) we refer to [3].

Schemes adding extra integrity. The first method achieves chosen-ciphertext security by adding an ephemeral integrity check to the ciphertext, essentially consisting of the tag of a message authentication code (MAC). Examples of such schemes are given in [4, 22] but the best known instance is probably DHIES [1] by Abdalla, Bellare and Rogaway. DHIES is as efficient as ElGamal in terms of computational efficiency. The scheme, however, has one group element

[1] The strong DH problem is computing g^{ab}, given random (g, g^a, g^b) and a restricted DDH oracle $\mathsf{DDH}_{g,a}(g^{\hat{b}}, g^{\hat{c}})$ which outputs 1 iff $a\hat{b} = \hat{c}$. Note that this is a weaker assumption than the related gap DH problem which grants access to a full DDH oracle.

plus a MAC tag of the ciphertext overhead, which is thus longer than that of ElGamal's. Its security can also be proved based on the strong Diffie-Hellman assumption in the random oracle model.[2] We remark that other schemes obtained using generic transformations (for example the Fujisaki-Okamoto transform [14] or the transformations from [10, 11]) also suffer from a similar overhead.

The KEM/DEM framework. The second method is to construct a hybrid encryption scheme by combining key encapsulation (KEM) with CCA secure data encapsulation (DEM) [12]. The latter is typically constructed by adding a MAC to a one-time (passively) secure symmetric cipher (just like DHIES) but one can instead use a length-preserving strong pseudorandom permutation (PRP, e.g., [16]) to avoid such an overhead [16]. For (hashed) ElGamal this was done by Kurosawa and Matsuo [20] to obtain a variant of DHIES [1] whose ciphertexts are as compact as the ones from ElGamal.

However, there are two drawbacks with this approach. First, the strong PRPs obtained from modes of operation are generally considered to be complicated and not very efficient in practice: for example, they are all "two-pass schemes" meaning that plaintext/ciphertext need to be passed twice at encryption/decryption. Second, more importantly, a strong PRP can only *securely* encrypt messages whose length is at least one block length of the underlying symmetric cipher, i.e., typically at least 128 bits. (Smaller messages can of course be padded to one block thereby adding overhead.) In fact, this drawback is an inherent artifact of the security properties of DEMs that are CCA-secure and length-preserving (DEMs for which ciphertext-size equals plaintext-size): any length-preserving CCA-secure DEM encrypting messages of m bits can be trivially broken in time complexity 2^m.[3] Consequently, length-preserving CCA secure DEMs encrypting short messages (e.g, $m = 10$ bits) cannot exist and therefore the KEM/DEM framework is not suitable for the purpose of obtaining chosen-ciphertext secure encryption with compact ciphertexts for short messages.

Tag-KEM/DEM framework. Another generic method, known as the Tag-KEM/DEM framework [2], accepts the most simple form of a DEM such as a one-time pad. On the other hand, the framework is slightly more demanding on the KEM part. In particular, tag-KEMs whose ciphertexts consist only of a single group element are not known to exist so far.

[2] We remark that there also exists a standard model proof for DHIES based on the "Hashed Oracle Diffie-Hellman" assumption which is an interactive assumption involving a hash function that is close to the strong DH assumption in the random oracle model.

[3] The attack is as follows. Given an a challenge ciphertext $e^* \in \{0,1\}^m$ of an unknown message $m^* \in \{0,1\}^m$ an attacker queries the decryption oracle for all ciphertexts $e \in \{0,1\}^m \setminus \{e^*\}$. Since the cipher is length-preserving and hence a permutation on $\{0,1\}^m$ the challenge message must be the one not output by the decryption oracle. In general, a CCA-secure DEM that adds r bits of redundancy can be broken in with 2^{m+r} decryption queries.

Table 1. Efficiency comparison among some ElGamal-type CCA schemes instantiated in a group of order 2λ bits, where λ is the security parameter. An element in the group is assumed to be representable with 2λ bits, one MAC tag with λ bits. "Computation" counts the number of single exponentiations and costs for other building blocks. Here we adopt the convention that 1 multi-base exponentiation is counted as 1.5 standard exponentiations. "mac" (or "hash") counts the cost for MAC-ing (or Hash-ing, resp.) a (potentially long) message. "sprp" counts the extra cost needed for the strong PRP compared to the simple one-time secure symmetric encryption. All other computations are ignored. The assumptions are as follows. CDH: Computational DH; SDH: Strong-DH; GDH: Gap-DH; DDH: Decisional-DH. All but KD are in the random oracle model. See also footnote 2 for DHIES and KM.

Scheme	Ciphertext Overhead	Message Size	Computation Enc.	Computation Dec.	Assumption
ElGamal [13]	2λ	any	2	1	DDH (CPA only)
KD [19]	5λ	any	3.5+mac	2.5+mac	DDH
DHIES [1]	3λ	any	2+mac	1+mac	SDH
Twin DHIES [9]	3λ	any	3+mac	1.5+mac	CDH
KM [20]	2λ	$\|m\| > \lambda$	2+sprp	1+sprp	SDH
Twin KM [9]	2λ	$\|m\| > \lambda$	3+sprp	1.5+sprp	CDH
Boyen [6]	2λ	$\|m\| > 2\lambda$	3+2·hash	2+2·hash	GDH
Basic scheme (§3.2)	2λ	any	2.5+hash	1+hash	SDH
Twin scheme (§3.4)	2λ	any	3.5+hash	2+hash	CDH

Others. In independent work another Diffie-Hellman type scheme was proposed by Boyen [6]. His construction primarily focuses on a tight security reduction. However, compared to ElGamal it requires twice as much computation for encryption and decryption. More importantly, it inherits the limitation from the KEM/DEM framework and can only securely encrypt messages longer than one group element (or 2λ bits): for small messages Boyen's security proof is no longer valid and does not seem to be fixable with the same ideas.

Summary. Compared to hashed ElGamal, all known Diffie-Hellman type encryption schemes suffer from either non-optimal ciphertext expansion; or decreased computational efficiency; or a restriction in the message length; or a combination of these. Surprisingly, until today no chosen-ciphertext secure encryption scheme is known that encrypts *small messages* with only one group element of ciphertext overhead. The most efficient scheme encrypting small (and there even arbitrary) messages is DHIES whose ciphertext overhead sums to one group element plus λ bits for the MAC. (Here λ is the security parameter, e.g., $\lambda = 80, 128$ bits.)

1.3 Technical Overview

Our scheme follows the Tag-KEM/DEM framework [2], which combines symmetric and asymmetric encryption. Our scheme is designed in a redundancy-free manner so that decryption always outputs a message (and never rejects). If any part of a given ciphertext is modified, the corresponding plaintext message becomes completely unrelated to the original plaintext. Such a property is obtained by using a hash of the symmetric part of the ciphertext (the "tag") to compute the asymmetric part of the ciphertext. Thus, modifying any part of a given ciphertext affects (directly or indirectly) the corresponding session-key and results that the symmetric part is decrypted differently.

Dealing with very short messages brings an unexpected technical difficulty in proving security; since the input to the hash function is short, its output distribution is very limited even when the hash function is modeled as a random oracle. More concretely, the adversary can exhaustively search for a short input that is consistent with a preliminary obtained output. We therefore cannot directly use the "oracle patching technique" (i.e., programmability) when the message is short. On the other hand, the security proof for long inputs can be done using a combination of techniques from [1, 2, 5].

The above difficulty is overcome by constructing two different reductions that deal with short and long messages separately and randomly use one of them hoping that the adversary chooses challenge messages of length that fits to the selected reduction algorithm. More details with intuition are shown in the proof outline given right after Theorem 1 in Section 3.3, and full proof details are in Section 4.

2 Definitions

NOTATION. Throughout the paper we denote by $\lambda \in \mathbb{N}$ the security parameter, i.e., we use choose our primitives such that an adversary requires "complexity" $\approx 2^{\lambda}$ to break them. Typical choices are $\lambda = 80, 128, \ldots$ bits.

PUBLIC KEY ENCRYPTION. We recall the usual definitions for chosen ciphertext security. Let PKE be a public-key encryption scheme. Consider the usual chosen ciphertext attack game, played between a challenger and a adversary A:

1. Given the security parameter λ, the challenger generates a public key/secret key pair, and gives the public key to A;
2. A makes a number of *decryption queries* to the challenger; each such query is a ciphertext c; the challenger decrypts c, and sends the result to A;
3. A makes one *challenge query*, which is a pair of messages (m_0, m_1); the challenger chooses $\beta \in \{0,1\}$ at random, encrypts m_β, and sends the resulting ciphertext c^* to A;
4. A makes more *decryption queries*, just as in step 2, but with the restriction that $c \neq c^*$;
5. A outputs $\hat{\beta} \in \{0,1\}$.

The advantage $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A},\mathsf{PKE}}(\lambda)$ is defined to be $|\Pr[\hat{\beta} = \beta] - 1/2|$. The scheme PKE is said to be secure against chosen ciphertext attack if for all efficient adversaries A, the advantage $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A},\mathsf{PKE}}(\cdot)$ is negligible.

If we wish to analyze a scheme PKE in the random oracle model, then hash functions are replaced by random oracle queries as appropriate, and both challenger and adversary are given access to the random oracle in the above attack game.

Length-preserving symmetric encryption. A symmetric encryption scheme $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ consists of two deterministic algorithms that are used to encrypt and decrypt a message $m \in \{0,1\}^*$ with a symmetric key K from key-space $\{0,1\}^{\lambda_e}$. SE is length-preserving if for all keys K and message lengths $\ell \geq 1$, $\mathsf{E}_K(\cdot)$ is a permutation on $\{0,1\}^\ell$. We require SE to be secure against one-time attacks, where an adversary C has to distinguish $\mathsf{E}_K(m)$ from a randomly chosen bit-string of length $|m|$, where K is randomly chosen and message m is chosen by the adversary. Let $\mathbf{Adv}^{\mathrm{ot}}_{\mathsf{C},\mathsf{SE}}$ be the advantage function of an adversary C in the above distinguishing game. We say SE is secure against one-time attacks if $\mathbf{Adv}^{\mathrm{ot}}_{\mathsf{C},\mathsf{SE}}$ is negligible for all efficient adversaries C.

3 Compact CCA-Secure Encryption

3.1 Building Blocks

Prime-order groups. Let $q \approx 2^{2\lambda}$ be a prime, $\mathbb{G}$ be a (multiplicative) group of order q. In this section all arithmetics are done in $\mathbb{G}$ unless otherwise noted. The Diffie-Hellman problem is given random $(g, g^a, g^b) \in \mathbb{G}^3$, compute $g^{ab} \in \mathbb{G}$. The strong Diffie-Hellman problem [1] is the same as the DH problem, but given access to a an oracle $\mathsf{DDH}_{g,a}(\cdot,\cdot)$ that on input $g^{\hat{b}}, g^{\hat{c}}$ outputs 1 iff $a\hat{b} = \hat{c} \bmod q$. We remark that in pairing groups (also known gap groups [21]) the DDH oracle can be efficiently instantiated and hence the strong-DH problem is equivalent to the DH problem (and the gap-DH problem [21]). The advantage $\mathbf{Adv}^{\mathrm{sdh}}_{\mathsf{B},\mathbb{G}}$ is defined to be the probability that an adversary B solves the above strong-DH problem. The strong-DH assumption holds in $\mathbb{G}$ if for all efficient adversaries B, the advantage $\mathbf{Adv}^{\mathrm{sdh}}_{\mathsf{B},\mathbb{G}}$ is negligible.

Symmetric Encryption. Let $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be a length-preserving symmetric encryption scheme that takes keys $K \in \{0,1\}^{\lambda_e}$ of length $\lambda_e = 2\lambda$ and encrypts arbitrary messages $m \in \{0,1\}^*$. We require that SE treats short and long messages in a different way as follows.

Long Messages. For all messages $m \in \{0,1\}^\ell$ of length $\ell > \lambda_e$ (larger than $|K|$), SE is secure against one-time attacks. A practical construction first expands K to a bit-string $K' \in \{0,1\}^\ell$ using a pseudo-random generator (key-derivation function), and then uses K' as an xor-based one-time pad to mask m, i.e., $e = m \oplus K'$.

Short Messages. For all messages $m \in \{0,1\}^\ell$ of length $\ell \leq \lambda_e$ (shorter than $|K|$), SE acts as a perfectly secure one-time pad by using the first ℓ bits of K as an xor-based one-time pad to mask m.

We remark that it is further possible to relax the condition for encryption of short messages to a computational one.

3.2 The Basic Scheme

Let $\mathbb{G}$ be group of prime order $q \approx 2^{2\lambda}$ and $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be a length-preserving symmetric ciphertext using keys of length $\lambda_e \geq 2\lambda$. Let $G : \{0,1\}^* \to \mathbb{Z}_q$ and $H : \mathbb{G} \to \{0,1\}^{\lambda_e}$ be hash functions that will be modeled as random oracles.

Key-Generation. The public key consists of a random group element $g \in \mathbb{G}$ and $h = g^x \in \mathbb{G}$, where the secret key $x \in \mathbb{Z}_q$ is a random index.

Encryption. To encrypt a plaintext $m \in \{0,1\}^\ell$ of arbitrary length $\ell \geq 1$, a random $r \in \mathbb{Z}_q$ is picked and the symmetric key is computed as

$$K = H(g^r) \in \{0,1\}^{\lambda_e} .$$

Next, the ciphertext is computed as

$$e = \mathsf{E}_K(m), \quad u = \left(g^{G(e)} \cdot h\right)^r .$$

The ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$.

Decryption. Using the secret-key $x \in \mathbb{Z}_q$, decrypting a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done by first computing the symmetric key

$$K = H\left(u^{\frac{1}{G(e)+x}}\right) \in \{0,1\}^{\lambda_e}$$

and then computing the plaintext as $m = \mathsf{D}_K(e)$. Note that decryption never rejects.

We note that if $u = 1 \in \mathbb{G}$, encryption fails. This happens with negligible probability $1/q$. For simplicity this negligible encryption error will be ignored for the remainder of the paper. An error-free variant can be obtained by adjusting x and G.

To verify correctness note that $u = (g^{G(e)}h)^r = g^{r \cdot (G(e)+x)}$ which implies $u^{1/(G(e)+x)} = g^r$.

3.3 Security

Theorem 1. *Assume the strong-DH assumption holds in $\mathbb{G}$, G and H are modeled as random oracles, and* SE *is a symmetric cipher with the properties specified in Subsection 3.1. Then the above PKE scheme is* IND-CCA *secure.*

In particular, suppose A *is an* IND-CCA *adversary that runs in time τ and makes at most q_h hash queries to H, q_g hash queries to G, and q_d decryption queries. Then there exists a strong DH adversary* B *that runs in time at most $\tau + O(q_h q_d \cdot T)$ and makes at most $O(q_h q_d)$ oracle calls and an adversary* C *against* SE *that runs in time at most $\tau + O(q_h q_d \cdot T)$, where we denote by T the unit time to perform one exponentiation in the group $\mathbb{G}$; moreover,*

$$\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A}}(\lambda) \leq 2\lambda_e \cdot \mathbf{Adv}^{\mathrm{sdh}}_{\mathsf{B}}(\lambda) + 2\mathbf{Adv}^{\mathrm{ot}}_{\mathsf{C},\mathsf{SE}}(\lambda) + \frac{q_h + 2q_g}{2^{\lambda_e}} + \frac{q_h}{q} ,$$

where $\lambda_e \geq 2\lambda$ is the length of the symmetric keys used in SE.

Proof Outline. At a high level the reduction works as follows. Given a strong DH problem instance (z, z^a, z^b) the strong DH adversary defines the public-key in such a way that $g = z^a$ and $h = z \cdot g^{-w^*}$, for random w^*. This implicitly defines the private-key as $x = \frac{1}{a} - w^*$. This setup of the public-key is inspired by "IBE techniques" [5] and allows the strong DH adversary to simulate decryption of all ciphertext except the challenge itself. (We remark that it was also successfully applied in the proofs of other encryption scheme, see, e.g., [7, 18, 17, 9].) Furthermore, a technique from [1] is used to "patch" random-oracle and decryption oracle and make them consistent by using the DDH oracle provided by the strong-DH experiment. Here the crucial property is to be able to check efficiently if for a given ciphertext $u \in \mathbb{G}$, an $R \in \mathbb{G}$, and an index $w \neq w^*$ it holds that $R = u^{\frac{1}{w+x}}$ or not. The latter can be done by querying the DDH oracle on $\mathsf{DDH}_{z,a}(u \cdot R^{w^*-w}, R)$. The challenge ciphertext is set to (u^*, e^*), where $u^* = z^b$ and e^* is a random bitstring that equals the length of one of the challenge messages, m_0 or m_1. Furthermore, $G(e^*)$ is defined as w^* such that ${u^*}^{1/(x+G(e^*))} = z^{ab}$. Hence an IND-CCA adversary has no chance in guessing the challenge bit without querying H for z^{ab}.

One difficulty in the proof is that $w^* = G(e^*)$ is implicitly set at the beginning of the reduction. If the IND-CCA adversary decides to get challenged on long messages, it is unlikely that the random ciphertext e^* of the same length is asked to G before the challenge query is made. So the strong DH adversary *never* returns w^* to any query to G made before the challenge query. However, if the IND-CCA adversary decided to get challenged on very short messages, it may decide to query all possible e^* of that length to G before making the challenge query. So the strong DH adversary *has to* return w^* to one of these queries to be successful in this case. Accordingly, the DH adversary has to behave contrarily in the two cases, but which case happens only depends on the adversary.

To handle the above situation, we actually construct two independent strong DH adversaries where at least one of the adversaries will be successful. The first strong DH adversary is successful if the length of the challenge message is smaller than the key-length λ_e. If this is the case the reduction hopes to guess the *length* of the challenge message correctly in order to be able to define the symmetric part of the challenge ciphertext at the beginning of the experiment. Overall, given the challenge message is sufficiently small, this first strong DH adversary wins if it correctly guessed the length of the challenge ciphertext which happens with probability $1/\lambda_e$. The second strong DH adversary is successful if the length of the challenge message is larger than the key-length λ_e. In that case it hopes that the symmetric part of the challenge ciphertext remains hidden from the IND-CCA adversary's view until the actual challenge query is done. This makes it possible to patch $G(e^*)$ after the challenge query was done. Overall, given the challenge message is sufficiently large, this second strong DH adversary wins as long as the IND-CCA adversary did not guess the symmetric part of the challenge ciphertext which happens with probability $q_g/2^{\lambda_e}$. Combining the two strong DH adversaries explains the loss in the security reduction. A formal proof is given in Section 4.

3.4 The Twin Scheme: Security from Standard Diffie-Hellman

Using the "twinning technique" from [9] one can obtain a scheme which can be proven secure under the standard (i.e., not strong) Diffie-Hellman assumption by replacing $H(g^r)$ by $H(g^r, \hat{g}^r)$, where $\hat{g} = g^{\hat{x}}$ is another element from the public key. The twinned scheme is defined follows.

Key-Generation. The public key consists of a random group elements g, $h = g^x$, and $\hat{g} = g^{\hat{x}}$, where the secret key $x, \hat{x} \in \mathbb{Z}_q^2$ are random indices.

Encryption. To encrypt a plaintext $m \in \{0,1\}^\ell$ of arbitrary length $\ell \geq 1$, a random $r \in \mathbb{Z}_q$ is picked and the symmetric key is computed as

$$K = H(g^r, \hat{g}^r) \in \{0,1\}^{\lambda_e} .$$

Next, the ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is computed as

$$e = \mathsf{E}_K(m), \quad u = \left(g^{G(e)} \cdot h\right)^r .$$

Decryption. Using the secret-key $x, \hat{x} \in \mathbb{Z}_q$, decrypting a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done by first computing the symmetric key

$$K = H\left(u^{\frac{1}{G(e)+x}}, u^{\frac{\hat{x}}{G(e)+x}}\right) \in \{0,1\}^{\lambda_e}$$

and then computing the plaintext as $m = \mathsf{D}_K(e)$.

Theorem 2. *Assume the DH assumption holds in $\mathbb{G}$, G and H are modeled as random oracles, and* SE *is a symmetric cipher with the properties specified in Subsection 3.1. Then the above PKE scheme is* IND-CCA *secure.*

Proof sketch. The Strong Twin Diffie-Hellman (STDH) assumption [9] is as follows. Given $z, z^a, z^{\hat{a}}, z^b$, computing the tuple $(z^{ab}, z^{\hat{a}b})$ is computationally infeasible, even with access to a Twin DDH oracle $\mathsf{TDDH}_{z,a,\hat{a}}(z^c, z^d, z^{\hat{d}})$ that outputs 1 iff $ac = d$ and $\hat{a}c = \hat{d}$. Since it was shown in [9] that the standard CDH assumption implies the STDH assumption, it is sufficient to prove security of the above scheme assuming the STDH assumption. We now sketch this reduction, focusing mostly on the differences to the proof of Theorem 1.

Given a STDH problem instance $(z, z^a, z^{\hat{a}}, z^b)$ the STDH adversary defines the public-key in such a way that $g = z^a$, $\hat{g} = z^{\hat{a}}$, and $h = z \cdot g^{-w^*}$, for random w^*. This implicitly defines the private-keys as $x = \frac{1}{a} - w^*$ and $\hat{x} = \frac{\hat{a}}{a}$. The challenge ciphertext is set to (u^*, e^*), where $u^* = z^b$ and e^* is a random bitstring that equals the length of one of the challenge messages, m_0 or m_1. Furthermore, $G(e^*)$ is defined as w^* such that $u^{*1/(x+G(e^*))} = z^{ab}$ and $u^{*\hat{x}/(x+G(e^*))} = z^{\hat{a}b}$. Hence an IND-CCA adversary has no chance in guessing the challenge bit without querying H for the tuple $(z^{ab}, z^{\hat{a}b})$, which is exactly the challenge for the STDH adversary.

The decryption queries are simulated by patching the random oracles using the Twin DDH oracle $\mathsf{TDDH}_{z,a,\hat{a}}(\cdot,\cdot,\cdot)$. Here the crucial property is again to be able to check efficiently if for a given ciphertext $u \in \mathbb{G}$, a tuple $(R_1, R_2) \in \mathbb{G}^2$, and an index $w \neq w^*$ it holds that $(R_1, R_2) = (u^{\frac{1}{w+x}}, u^{\frac{\hat{x}}{w+x}})$ or not. Define $\hat{R} = u \cdot R_1^{w^*-w}$. Using the equation for $x, \hat{x}$, the latter condition can be proved equivalent to $(R_1, R_2) = (\hat{R}^a, \hat{R}^{\hat{a}})$ which can be checked by calling $\mathsf{TDDH}_{z,a,\hat{a}}(\hat{R}, R_1, R_2)$. The remaining proof is similar to the one of Theorem 1. We remark that the concrete security bound of Theorem 2 is essentially the same as the one of Theorem 1 plus some small additive statistical parameter stemming from the twinning technique.

3.5 Efficiency

We note that in both schemes the value $u = (g^{G(e)} \cdot h)^r$ from encryption can be computed using one multi-exponentiation which is about as efficient as one standard exponentiation. For our basic scheme the parameters for key-generation, encryption and decryption, as well as for the size of the public/secret keys are exactly the same as for ElGamal. Furthermore, as in ElGamal, our scheme's ciphertext overhead only consists of one group element, independent of the message. We remark that implemented on certain elliptic curves this accounts to $\log q \approx 2\lambda$ bits overhead for conjectured λ bits of security (which is optimal due to the generic attacks on the discrete logarithm problem in $\mathbb{G}$). For an efficiency comparison with related schemes we refer to Table 1 in Section 1.

4 Proof of Theorem 1

We proceed in games, starting with Game 0 which is the original IND-CCA experiment run with an adversary A. Each Game i, $i \geq 0$, is entirely defined through the algorithms $\textsc{Setup}_i$, $\textsc{Challenge}_i$, $\textsc{Decrypt}_i$, $\textsc{HashH}_i$, and $\textsc{HashG}_i$, describing the adversary's view. If a game does not mention one of the above implementing algorithms then it is assumed to be unchanged compared to the last game. In Game i, $i \geq 0$, we define X_i as the event that $\beta = \hat{\beta}$. We make the general convention that all terms with a superscript asterisks $(^*)$ are (or at least will be in a future game) connected to the challenge ciphertext; for example, K^* is the symmetric key used for generating the challenge ciphertext.

Game 0. This game implements the IND-CCA experiment. We make the following conventions how the appearing random-variables are computed throughout the experiment.
$\triangleright\textsc{Setup}_0$. Given a generator $z \in \mathbb{G}$, the experiment picks random $a \in \mathbb{Z}_q^*$, $w^* \in \mathbb{Z}_q$, $B \in \mathbb{G}$ and defines

$$A = z^a, \quad R^* = B^a \ . \tag{1}$$

Next, it defines the public key as

$$g := A, \quad h := z \cdot A^{-w^*} .$$

(Note that this implicitly defines the secret key as $x = 1/a - w^*$.) We will now equivalently rewrite the rest of the IND-CCA experiment using these definitions of public/secret key.
$\triangleright$CHALLENGE$_0$. The challenge ciphertext (u^*, e^*) for message m_β of length ℓ^* is generated as

$$K^* = H(R^*), \quad e^* = \mathsf{E}_{K^*}(m_\beta), \quad u^* = B \cdot {R^*}^{G(e^*)-w^*}, \tag{2}$$

where R^* and B are taken from (1).
$\triangleright$DECRYPT$_0$. Decryption of a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done as follows.

$$R = u^{\frac{a}{1+a\cdot(G(e)-w^*)}}, \quad K = H(R), \quad m = \mathsf{D}_K(e) \; . \tag{3}$$

$\triangleright$HASHH$_0$/HASHG$_0$. Hash queries to H and G are answered with random values from the respective domains.

One can easily verify that Game 0 is an equivalent rewrite of the original IND-CCA experiment. Hence,

$$|\Pr[X_0] - 1/2| = \mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A}}(\lambda) \; . \tag{4}$$

Game 1. $\triangleright$HASHG$_1$=HASHG$_0$, but abort if $G(e) = G(e^*)$ for any value $e \neq e^*$.

Since there are at most q_g hash queries to G, each of them taking an independent random value in $\mathbb{Z}_q$, we have that the experiment aborts with probability at most q_g/q. Since Game 0 and 1 are equivalent until the new abort happens,

$$|\Pr[X_1] - \Pr[X_0]| \leq q_g/q \; .$$

Game 2. $\triangleright$HASHH$_2$=HASHH$_1$, but abort if $H(R^*)$ is asked before CHALLENGE$_2$ is executed.

Since R^* is perfectly hidden from A's view until CHALLENGE$_2$ is executed, we have

$$|\Pr[X_2] - \Pr[X_1]| \leq q_h/q \; .$$

Game 3. We now use an alternative (but equivalent) simulation of random oracle H and the decryption oracle. To make the simulation consistent we introduce the following two tables which are initially empty.

- $\mathit{List}_H[R] = K$ means $H(R) = K$, for $R \in \mathbb{G}$.
- $\mathit{List}_D[u, e] = K$ means that $K \in \{0,1\}^{\lambda_e}$ is the symmetric key that was used to decrypt ciphertext (u, e), for $u \in \mathbb{G}$ and $e \in \{0,1\}^\ell$.

$\triangleright$DECRYPT$_3$. Decryption of a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done as follows. As in (3) of DECRYPT$_2$, first the "algebraic key" $R = u^{\frac{a}{1+a(G(e)-w^*)}}$ is computed. Next, the symmetric key K is computed according to the following case distinction.

Case D1: If $\mathit{List}_H[R]$ is defined, then use $K = \mathit{List}_H[R]$.

Case D2: If there exists an entry $(\hat{u}, \hat{e})$ in $List_D$ such that $R = \hat{u}^{\frac{a}{1+a(G(\hat{e})-w^*)}}$, then use $K = List_D[\hat{u}, \hat{e}]$.
Case D3: In the second phase, if $R = u^{*\frac{a}{1+a(G(e^*)-w^*)}}$, then use $K = H(R')$, where $R' = (u/u^*)^{1/(G(e)-G(e^*))}$. (Note that $R' = R^*$ and, since $G(e) \neq G(e^*)$, R' is well-defined.)
Case D4: Otherwise, pick a random key $K \in \{0,1\}^{\lambda_e}$.

In all cases define $List_D[u, e] := K$ and return $m = \mathsf{D}_K(e)$.
$\triangleright$HashH$_3$. A hash query $H(R)$ is answered as follows.

Case H1: If $List_H[R]$ is defined then return this value.
Case H2: If there exists an entry $List_D[u, e]$ such that $R = u^{\frac{a}{1+a(G(e)-w^*)}}$ then use $K = List_D[u, e]$.
Case H3: Otherwise, pick a random K.

In all cases define $List_H[R] := K$ and return $H(R) = K$.
$\triangleright$Challenge$_3$=Challenge$_2$, but now Challenge$_3$ explicitly defines $List_H$ $[R^*] := K^*$ for uniform $K^* \in \{0,1\}^{\lambda_e}$ (and not implicitly through a call to $H(R^*)$ as in (2)).

It is easy to verify that this does not change the behavior of the oracles.

$$\Pr[X_3] = \Pr[X_2]\ . \tag{5}$$

The next lemma shows that using a DDH oracle, Decrypt$_3$ and HashH$_3$ can be simulated without explicit knowledge of $a = \log_z A$.

Lemma 1. *Algorithms* Decrypt$_3$ *and* HashH$_3$ *can be efficiently simulated depending on the values z and $A = z^a$, plus making maximal $O(q_H q_D)$ calls to an oracle $\mathsf{DDH}_{z,a}(\cdot,\cdot)$.*

Proof. We start with Decrypt$_3$. The problem is that the simulation cannot compute the value R without knowing a. However, it can perform the checks using $A = z^a$ and the DDH oracle. In case D1, to check if $List_H[R]$ is defined the simulator checks if there exists an entry $\hat{R}$ in $List_H$ satisfying $\hat{R} = (u/\hat{R}^{G(e)-w^*})^a$. The latter one is equivalent to $\mathsf{DDH}_{z,a}(u/\hat{R}^{G(e)-w^*}, \hat{R}) = 1$. In case D2, check if $\mathsf{DDH}_{z,a}(\hat{u}^{G(e)-w^*}/u^{G(\hat{e})-w^*}, u/\hat{u}) = 1$ for some entry $(\hat{u}, \hat{e})$ from $List_D$. The case D3 is similar. Furthermore, simulation of HashH$_3$ can be done in a similar way.

Note that, using the DDH oracle, Game 3 can be simulated only knowing the values $z, B, A = z^a$. The value $R^* = B^a$ is only needed for the execution of Challenge$_3$, i.e., to generate the element u^* of the challenge ciphertext (2) and to define $List_H[R^*] := k^*$. Now we would like to "define" $G(e^*) := w^*$ to be able to use $u^* := B$ (independent of R^*) for the generation of the challenge ciphertext. However, this seems difficult since e^* depends on the input m_0 and m_1 of the adversary. In particular, m_0 could be small in which case e^* (which is of the same length as m_0) can be guessed by A.

Game 4. From this game on we slightly change the way the experiment is executed. The experiment initially flips a random coin $c \in \{0,1\}$ representing a guess if the adversary wants to get challenged on short or long messages. Depending on c, the experiment is executed using the algorithms $\textsc{Decrypt}_4$=$\textsc{Decrypt}_3$, $\textsc{HashH}_4$=$\textsc{HashH}_3$, $\textsc{HashG}_4$=$\textsc{HashG}_3$, $\textsc{Setup}_4^c$=$\textsc{Setup}_3$, and $\textsc{Challenge}_4^c$, where $\textsc{Challenge}_4^0$ and $\textsc{Challenge}_4^1$ are defined as follows.
$\triangleright\textsc{Challenge}_4^0$=$\textsc{Challenge}_3$, with the difference that it aborts (and returns a random bit $\hat{\beta}$) if $|m_0| > \lambda_e$. (Here m_0 is one of the challenge messages.)
$\triangleright\textsc{Challenge}_4^1$=$\textsc{Challenge}_3$, with the difference that it aborts if $|m_0| \leq \lambda_e$.

Since bit c is independent from the adversaries view, we have

$$\Pr[X_3] - \frac{1}{2} = 2(\Pr[X_4] - \frac{1}{2}) \,. \tag{6}$$

In Game i ($i \geq 4$), we define $\Pr[X_i^{\text{short}}] = \Pr[X_i \mid |m_0| \leq \lambda_e]$ and $\Pr[X_i^{\text{large}}] = \Pr[X_i \mid |m_0| > \lambda_e]$ such that

$$\begin{aligned}\Pr[X_4] &= \Pr[X_4^{\text{short}}] \cdot \Pr[|m_0| \leq \lambda_e] + \Pr[X_4^{\text{large}}] \cdot \Pr[|m_0| > \lambda_e] \\ &\leq \max\{\Pr[X_4^{\text{short}}], \Pr[X_4^{\text{large}}]\} \,.\end{aligned}$$

In the following games we will bound $\Pr[X_4^{\text{short}}]$ and $\Pr[X_4^{\text{large}}]$ separately.

Game 5. $\triangleright\textsc{Setup}_5^0$=$\textsc{Setup}_4^0$, but additionally a random bit-string $\ell^*_{\text{guess}} \in \{1,\ldots,\lambda_e\}$ is selected.
$\triangleright\textsc{Challenge}_5^0$=$\textsc{Challenge}_4^0$, but it aborts if $\ell^* \neq \ell^*_{\text{guess}}$, where ℓ^* is the length of the challenge message m_0 submitted by A.

Since ℓ^*_{guess} is picked uniformly from $\{1,\ldots,\lambda_e\}$, independent of the adversary's view, we have

$$\Pr[X_4^{\text{short}}] - \frac{1}{2} = \lambda_e \cdot (\Pr[X_5^{\text{short}}] - \frac{1}{2}) \,. \tag{7}$$

Game 6. $\triangleright\textsc{Setup}_6^0$=$\textsc{Setup}_5^0$, but additionally a bit-string $e^* \in \{0,1\}^{\ell^*_{\text{guess}}}$ is selected, uniformly at random.
$\triangleright\textsc{HashG}_6^0$. Since e^* is now determined at the beginning, $\textsc{HashG}_6^0$ can now program G such that $G(e^*) = w^*$.
$\triangleright\textsc{Challenge}_6^0$=$\textsc{Challenge}_5^0$. If $\ell^* = \ell^*_{\text{guess}}$ (i.e., if $\textsc{Challenge}_5^0$ did not abort) it uses the tuple $(u^* := B, e^*)$ as challenge ciphertext. (Since $G(e^*) = w^*$, this is a correctly distributed ciphertext for m_β.) Furthermore, the experiment defines $\mathit{List}_H[R^*] := K^*$, where $K^* = K_1^* || K_2^*$ with $K_1^* = e^* \oplus m_\beta$ and K_2^* is a random bit-string of length $\lambda_e - \ell^*$.

Consider the distribution of K^* and e^* used in $\textsc{Challenge}_5^0$ and $\textsc{Challenge}_6^0$. By the properties of SE and since $\ell^* = \ell^*_{\text{guess}} \leq \lambda_e$, in $\textsc{Challenge}_5^0$ a random key K^* is used and e^* is computed as $e^* = K_0^* \oplus m_\beta$, where K_1^* is the ℓ^* bit prefix of K^*. In $\textsc{Challenge}_6^0$ a random symmetric ciphertext e^* is used together with a random key K^* that encrypts m_β to e^*. Therefore, K^* and e^* have the same joint distribution in both games and

$$\Pr[X_5^{\text{short}}] = \Pr[X_6^{\text{short}}] \,. \tag{8}$$

Game 7. $\triangleright$CHALLENGE$_7^0$=CHALLENGE$_6^0$, but $List_H[R^*]$ is left undefined.

Clearly, the view of adversary A in Games 6 and 7 is the same until it queries $H(R^*)$ in the second phase. We claim that there exists an adversary B against strong DH with

$$|\Pr[X_7^{\text{short}}] - \Pr[X_6^{\text{short}}]| \leq \mathbf{Adv}_{\mathsf{B}}^{\text{sdh}}(\lambda)\ . \tag{9}$$

Adversary B runs the experiment from this game. By Lemma 1, HASHH$_7$= HASHH$_3$ and DECRYPT$_7$=DECRYPT$_3$ can be simulated using the values z, A, and B. If $H(R^*)$ is queried, B will notice using the DDH oracle and outputs $R^* = z^{ab}$.

Now it is clear that in case $c = 0$ the experiment is independent of the challenge key K^* and hence also of the challenge bit β.

$$\Pr[X_7^{\text{short}}] = 1/2\ . \tag{10}$$

Game 8. $\triangleright$CHALLENGE$_8^1$. For generating the challenge ciphertext (u^*, e^*) for message m_β of length ℓ^* the experiments picks a random $K^* \in \{0,1\}^{\lambda_e}$, defines $List_H[R^*] = K^*$, and computes

$$e^* = \mathsf{E}_{K^*}(m_\beta), \quad u^* = B.$$

The experiment aborts if $G(e^*)$ was already defined. Define this event as F_8^{large}. Otherwise, it defines $G(e^*) := w^*$ such that by (2) (u^*, e^*) is a correct ciphertext for m_β. Then,

$$|\Pr[X_8^{\text{large}}] - \Pr[X_7^{\text{large}}]| \leq \Pr[F_8^{\text{large}}]\ . \tag{11}$$

Game 9. $\triangleright$CHALLENGE$_9^1$=CHALLENGE$_8^1$, but after the challenge ciphertext is defined, the value $List_H[R^*]$ remains undefined.

Clearly, we have $\Pr[F_9^{\text{large}}] = \Pr[F_8^{\text{large}}]$, where F_9^{large} is the event that $G(e^*)$ was queried before the challenge ciphertext is defined. Furthermore, similar to Game 7, there exists an adversary B against the strong DH problem with

$$|\Pr[X_9^{\text{large}}] - \Pr[X_8^{\text{large}}]| \leq \mathbf{Adv}_{\mathsf{B}}^{\text{sdh}}(\lambda)\ . \tag{12}$$

Game 10. $\triangleright$CHALLENGE$_{10}^1$=CHALLENGE$_9^1$, but a random $e^* \in \{0,1\}^{\ell^*}$ is used for the challenge ciphertext (independent of m_β). Since a distinguisher between a random symmetric ciphertext e^* and $e^* = \mathsf{E}_{K^*}(m_\beta)$ (for known m_β) immediately implies an one-time adversary C against SE, we have

$$|\Pr[X_{10}^{\text{large}}] - \Pr[X_9^{\text{large}}]| \leq \mathbf{Adv}_{\mathsf{C},\mathsf{SE}}^{\text{ot}}(\lambda)\ . \tag{13}$$

Furthermore, since Game 10 is now independent of β, we have

$$\Pr[X_{10}^{\text{large}}] = 1/2\ . \tag{14}$$

Finally, e^* is a uniform element from $\{0,1\}^{\ell^*}$ with $\ell^* \geq \lambda_e$. Since adversary A makes maximal q_g hash queries to G, we have

$$\Pr[F_9^{\text{large}}] = \Pr[F_{10}^{\text{large}}] \leq q_g/2^{\lambda_e} \; . \tag{15}$$

Summing up the probabilities we obtain

$$\begin{aligned}\mathbf{Adv}^{\text{cca}}_{\mathsf{A}}(\lambda) &\leq \frac{q_h}{2^{\lambda_e}} + \frac{q_g}{q} + 2\max\{\lambda_e \cdot \mathbf{Adv}^{\text{sdh}}_{\mathsf{B}}(\lambda)\;,\, \mathbf{Adv}^{\text{sdh}}_{\mathsf{B}}(\lambda) + \mathbf{Adv}^{\text{ot}}_{\mathsf{C},\mathsf{SE}}(\lambda) + \frac{q_g}{2^{\lambda_e}}\} \\ &\leq \frac{q_h + 2q_g}{2^{\lambda_e}} + \frac{q_g}{q} + 2\lambda_e \cdot \mathbf{Adv}^{\text{sdh}}_{\mathsf{B}}(\lambda) + 2\mathbf{Adv}^{\text{ot}}_{\mathsf{C},\mathsf{SE}}(\lambda) \; .\end{aligned}$$

Furthermore, the running times of A (and C) is bounded by the running time of B plus the time of performing additional $O(q_H q_D)$ group operations (by Lemma 1).

References

[1] Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)

[2] Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005); also available at IACR e-print 2005/027 and 2004/194

[3] Abe, M., Kiltz, E., Okamoto, T.: Chosen Ciphertext Security with Optimal Ciphertext Overhead. In: Advances in Cryptology – Asiacrypt 2008. LNCS, vol. 5350, pp. 355–371. Springer, Heidelberg (2008); also available at IACR e-print 2008/374

[4] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communication Security, pp. 62–73. Association for Computing Machinery (1993)

[5] Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[6] Boyen, X.: Miniature CCA2 PK encryption: Tight security without redundancy. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 485–501. Springer, Heidelberg (2007)

[7] Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS 2005, pp. 320–329. ACM Press, New York (2005)

[8] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: Proceedings of the 30th annual ACM Symposium on Theory of Computing, pp. 209–218 (1998)

[9] Cash, D.M., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman Problem and Applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)

[10] Coron, J., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: GEM: A generic chosen-ciphertext secure encryption method. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 263–276. Springer, Heidelberg (2002)

[11] Coron, J., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: Optimal chosen-ciphertext secure encryption of arbitrary-length messages. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 17–33. Springer, Heidelberg (2002)

[12] Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)

[13] El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)

[14] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

[15] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28, 270–299 (1984)

[16] Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)

[17] Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)

[18] Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on Gap Hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007), `http://eprint.iacr.org/2007/036`

[19] Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)

[20] Kurosawa, K., Matsuo, T.: How to remove MAC from DHIES. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 236–247. Springer, Heidelberg (2004)

[21] Okamoto, T., Pointcheval, D.: The gap-problems: a new class of problems for the security of cryptographic schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)

[22] Okamoto, T., Pointcheval, D.: REACT: Rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, p. 159. Springer, Heidelberg (2001)

[23] Phan, D.H., Pointcheval, D.: Chosen-ciphertext security without redundancy. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 1–18. Springer, Heidelberg (2003)

Verifiable Rotation of Homomorphic Encryptions

Sebastiaan de Hoogh, Berry Schoenmakers, Boris Škorić, and José Villegas

Dept. of Mathematics and Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
s.j.a.d.hoogh@tue.nl, berry@win.tue.nl, b.skoric@tue.nl,
j.a.villegas@tue.nl

Abstract. Similar to verifiable shuffling (mixing), we consider the problem of verifiable rotating a given list of homomorphic encryptions. The offset by which the list is rotated (cyclic shift) should remain hidden. Basically, we will present zero-knowledge proofs of knowledge of a rotation offset and re-encryption exponents, which define how the input list is transformed into the output list. We also briefly address various applications of verifiable rotation, ranging from 'fragile mixing' as introduced by Reiter and Wang at CCS'04 to applications in protocols for secure multiparty computation and voting.

We present two new, efficient protocols. Our first protocol is quite elegant and involves the use of the Discrete Fourier Transform (as well as the Fast Fourier Transform algorithm), and works under some reasonable conditions. We believe that this is the first time that Fourier Transforms are used to construct an efficient zero-knowledge proof of knowledge.

Our second protocol is more general (requiring no further conditions) and only slightly less efficient than the DFT-based protocol. Unlike the previously best protocol by Reiter and Wang, however, which relies on extensive use of verifiable shuffling as a building block (invoking it *four* times as a sub-protocol), our construction is direct and its performance is comparable to the performance of a *single* run of the best protocol for verifiable shuffling.

1 Introduction

The well-known problem of verifiable shuffling (or, mixing) is to transform a given list of homomorphic encryptions into a list of randomly permuted, random re-encryptions of these encryptions, such that (i) it can be verified that the multiset of plaintexts for the input list and output list are identical, and (ii) the permutation used remains hidden. The original idea of mixing was introduced by Chaum, along with applications in anonymous email and voting [3], and the explicit goal of verifiability was introduced by Sako and Kilian, as part of a paper on voting [24]. Many improved protocols for verifiable shufflers/mixers have been published since, as well as many applications. A basic property is that a cascade of verifiable shufflers is again a verifiable shuffler, which enables mutually distrusting parties to take turns in permuting a given list such that no party knows the permutation for the final output list (unless all parties collude).

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 393–410, 2009.

In this paper we consider the related problem of verifiable rotating a given list of homomorphic encryptions. Rather than requiring the use of a random permutation, as in shuffling, we require that a random rotation $\pi_r(k) = k - r \pmod{n}$, where the offset r, $0 \leq r < n$, is chosen uniformly at random. Clearly, a cascade of verifiable rotators is also a verifiable rotator, for which no party knows by which offset the final output list has been rotated (unless all parties collude).

Verifiable rotation has actually been introduced by Reiter and Wang in the context of mixing [21]. They define 'fragile mixing' as a form of mixing that deters leaking of information: namely, when a single correspondence between an element on the input list and an element of the output list is revealed, then the correspondence between all elements of the input list and output lists is revealed. A fragile mix is therefore restricted to the use of rotations (called "loop permutations" in [21]). The protocol by Reiter and Wang, however, uses *four* invocations of a verifiable shuffle protocol (and some further work) to perform a verifiable rotation. We will reduce this to the work of about one verifiable shuffle, by following a completely different, more direct approach to the problem.

Apart from fragile mixing, however, there are many more applications of verifiable rotations. An important application arises in the context of secure integer comparisons, as noted first in [20]. A common step in many integer comparison protocols [2,1,10,8,20], requires parties to find out whether a special value occurs in a given list of encryptions. For example, whether there is a 0 among otherwise random values. The position of the special value should remain hidden. To this end, the list will be randomly permuted before decrypting the encryptions. However, rather than using a fully random permutation, as commonly proposed, a random rotation suffices to hide the position of the special value.

Similarly, it is easily seen that for protocols such as Mix & Match [16], which involve mixing of truth tables of Boolean gates, it suffices to apply a random rotation to the rows of a truth table, rather than a fully random permutation. The reason is that in the matching stage exactly one row will match, and a random rotation fully hides the corresponding row of the original truth table. The same observation applies to other forms of 'garbled circuit evaluation', which can be seen as variations on Yao's original method [27]. Likewise, in protocols for secure linear programming [17] the position of the pivot in each iteration of the simplex algorithm must be hidden to avoid leakage of information. Again, we note that the use of a rotation instead of a general permutation suffices.

Finally, we note that further applications exist in the context of electronic voting, where randomly rotated lists of encryptions are used in the construction of encrypted ballot forms (see, e.g., Prêt-à-Voter voting systems by Ryan et al. [23,22] and references therein): voters get a receipt in which one out of n positions is marked; due to a random rotation, the marked position does not reveal the identity of the candidate chosen. Recently, [26] presented a secure protocol for determining the winner of an election in a preferential electoral system. The goal at some intermediate point is to hide the position of a distinguished value in every row of a matrix of encrypted values. They achieve that by first rotating the row vectors in the clear, and later the rotation offsets are concealed by using verifiable mixes of each of the column vectors. The use of verifiable rotation would provide an alternative to the use of shuffling steps.

The goal of this paper is thus to design efficient protocols for verifiable rotations. Given a list $X_0, X_1, \ldots, X_{n-1}$ of (homomorphic) encryptions, a rotation is performed by picking an offset r, $0 \leq r < n$, at random, and creating a list of encryptions $Y_0, Y_1, \ldots, Y_{n-1}$, where each Y_k is a random re-encryption of X_{k-r}.[1] As the rotation offset r remains hidden if the underlying cryptosystem is semantically secure, a zero-knowledge proof is used to prove the correctness of a rotation. By using a non-interactive zero-knowledge proof, one obtains a verifiable rotation.

1.1 Our Contributions

We develop two new, efficient protocols for proving the correctness of a rotated list of homomorphic encryptions. These protocols allow for a direct construction of a cascade of verifiable rotators, which can be used to efficiently implement fragile mixing, among other applications, as highlighted above. Whereas the verifiable rotation protocol of [21] required the work of at least four verifiable shuffles, the work of our protocols is approximately equal to that of a single shuffle only—in some cases even slightly better than the currently best protocol for verifiable shuffling, due to Groth [12]. Note that *a priori* it is not clear whether verifiable shuffling is harder than verifiable rotation, or the other way around.

Our first protocol is a Σ-protocol and makes essential use of the Discrete Fourier Transform (DFT). To the best of our knowledge this is the first time that the DFT is actually used in the construction of an efficient zero-knowledge proof. The Σ-protocol turns out to be competitive with the best protocols for shuffling known to date. However, our DFT-based protocol relies on some constraints on the parameters, which should be met in order for the DFT to be applicable. For instance, in case we use ElGamal encryptions for a group of prime order q, we assume that $n \mid q-1$, where n denotes the length of the list of rotated encryptions. This ensures the existence of an n-th root of unity modulo q. Furthermore, to take full advantage of the Fast Fourier Transform (FFT) algorithm, we assume that n is an integral power of two.[2]

Our second protocol is more general and does not put any constraints on the parameters. Although it is not a (3-move) Σ-protocol, it is a 6-move honest-verifier zero-knowledge protocol for which we are able to prove knowledge soundness as well (more precisely, witness extended emulation). This general protocol is computationally only slightly more expensive than the DFT-based Σ-protocol.

[1] Throughout the paper, indices are reduced modulo n. E.g., X_{k-r} is short for $X_{k-r \pmod n}$.

[2] These constraints can be relaxed, for instance, by using the DFT for a quadratic extension of $\mathbb{F}_q$, which exists provided $n \mid q^2 - 1$. Similarly, for the FFT it suffices if all prime factors of n are small, rather than requiring that n is a power of two. We do not consider these relaxations in this paper.

Both protocols can be made non-interactive using the Fiat-Shamir heuristic, yielding efficient (publicly) verifiable non-interactive zero-knowledge proofs, secure in the random oracle model.

2 Preliminaries

We present our protocols assuming homomorphic ElGamal as the underlying cryptosystem. However, our results can be readily extended to other homomorphic cryptosystems, such as the Paillier cryptosystem.

2.1 Discrete Log Setting

Let $G = \langle g \rangle$ denote a finite cyclic (multiplicative) group of prime order q for which the Decision Diffie-Hellman (DDH) problem is assumed to be infeasible.

Homomorphic ElGamal Cryptosystem. For public key $h \in G$, a message $m \in \mathbb{Z}_q$ is encrypted as a pair $(a, b) = (g^r, g^m h^r)$ for $r \in_R \mathbb{Z}_q$. Given the private key $\alpha = \log_g h$, decryption of (a, b) is performed by calculating $b/a^\alpha = g^m$ and then solving for $m \in \mathbb{Z}_q$. As usual, it is assumed that m belongs to a sufficiently small subset of $\mathbb{Z}_q$ to make recovery of m feasible.

This encryption scheme is additively homomorphic: given $(a_1, b_1) = (g^{r_1}, g^{m_1} h^{r_1})$ and $(a_2, b_2) = (g^{r_2}, g^{m_2} h^{r_2})$, an encryption of $m_1 + m_2$ is obtained by pairwise multiplication $(a_1, b_1)(a_2, b_2) = (a_1 a_2, b_1 b_2) = (g^{r_1+r_2}, g^{m_1+m_2} h^{r_1+r_2})$. Homomorphic ElGamal is semantically secure under the DDH assumption.

As a shorthand notation for an encryption $(g^r, g^m h^r)$, we write $\mathsf{E}(m, r)$. Moreover, $\mathsf{E}(m)$ will denote an ElGamal encryption of $m \in \mathbb{Z}_q$, where the randomization is suppressed from the notation.

Pedersen Commitment. Given $\widetilde{h} \in G$, a Pedersen commitment to $m \in \mathbb{Z}_q$ is the value $b = g^m \widetilde{h}^r$ where $r \in \mathbb{Z}_q$. This commitment is opened by revealing m and r. The scheme is unconditionally hiding and computationally binding assuming that $\log_g \widetilde{h}$ cannot be computed. We use $\mathsf{C}(m, r)$ to denote a Pedersen commitment to m using randomness r, and abbreviate this to $\mathsf{C}(m)$ when suppressing the randomization from the notation.

Efficiency. As performance measure for our protocols we will count the number of exponentiations. Note that because of "Shamir's trick" [9], which is a special case of Straus' algorithm [25], the complexity of single, double and even triple exponentiations ($g_1^{r_1}$, $g_1^{r_1} g_2^{r_2}$ and $g_1^{r_1} g_2^{r_2} g_3^{r_3}$) are comparable.

2.2 Zero-Knowledge Proofs of Knowledge

A rotator must convince a verifier that the output list is indeed a rotation of the input list, without giving away any information on the offset used between these lists. For this purpose we use standard notions for zero-knowledge and knowledge soundness.

A proof, or argument, of knowledge for a relation $R = \{(x, w)\}$ is a protocol between a prover and a verifier. Both parties get a value x as common input while the prover gets a witness w as private input such that $(x, w) \in R$. At the end of the protocol, the verifier decides whether it accepts or rejects the proof.

A proof must be complete and sound. Completeness means that given a pair $(x, w) \in R$, the proof is accepting if both prover and verifier follow the protocol. Soundness captures the fact that a cheating prover cannot succeed convincing a verifier if the prover does not know a witness w for x. This is shown by the definition of a knowledge extractor which uses the prover to compute a witness, see, e.g., [11]. A proof is zero-knowledge if there exists a simulator that given x and access to a malicious verifier, produces a view of the protocol that is indistinguishable from the view when the verifier interacts with a real prover. In honest-verifier zero-knowledge (HVZK) proofs the verifier is assumed to be honest but curious. Additionally, an HVZK proof is called special HVZK (SHVZK) when the simulator can produce views for a given challenge of the verifier.

Examples of special honest-verifier zero-knowledge proofs of knowledge are the well-known Σ-protocols [6,4]. These are 3-move protocols where the prover acts first. They satisfy the so-called special-soundness property.

Our first protocol is a Σ-protocol. For our second protocol, we will actually show the existence of a knowledge extractor along the lines of Groth [12,15]. Concretely, we show that our protocols have a witness-extended emulator. This notion, introduced by Lindell [18] implies knowledge soundness as defined by Damgård and Fujisaki [7], as shown in [14].

Informally, the witness-extended emulation property says that given an adversarial prover that produces an acceptable proof with some probability ε, there exists an expected polynomial time algorithm E, called witness-extended emulator, that produces indistinguishable transcripts which are accepting with (essentially) the same probability ε. If the transcript is accepting then a witness is provided as well. The emulator has rewindable black-box access to the prover.

It can easily be shown that a Σ-protocol has the witness-extended emulation property [13]. This fact will be used in our security proofs.

3 DFT-Based Solution

Let $X = (X_0, X_1, \ldots, X_{n-1})$ be a list of homomorphic ElGamal encryptions. A random rotation is performed by picking a random offset r, $0 \leq r < n$, and computing a list of encryptions $Y = (Y_0, Y_1, \ldots, Y_{n-1})$, where $Y_k = X_{k-r}(g^{s_k}, h^{s_k})$, $s_k \in_R \mathbb{Z}_q$, for $k = 0, \ldots, n-1$. The challenge is to provide an efficient proof that the output list Y is correctly formed, for a given input list X.

The key mathematical tool for our first protocol is the Discrete Fourier Transform (DFT). Using the DFT one can express conveniently that two lists of encryptions are rotated version of each other, which allows for an efficient Σ-protocol to make a rotation verifiable.

3.1 Discrete Fourier Transform

Discrete Fourier Transform over Finite Fields. For simplicity, we present the DFT over the field $\mathbb{Z}_q$ for prime q. Suppose $n \mid q-1$ and let $\alpha \in \mathbb{Z}_q$ denote an n-th root of unity modulo q, that is, α is an element of order n in $\mathbb{Z}_q$. So, $\alpha^n = 1 \bmod q$.

The DFT for a sequence $x_k \in \mathbb{Z}_q$ w.r.t. α is a sequence $x'_k \in \mathbb{Z}_q$ defined as

$$x'_k = \sum_{j=0}^{n-1} x_j \alpha^{kj}.$$

Borrowing terminology from Fourier analysis, a Fourier Transform converts a sequence in the time domain into a sequence in the frequency (transformed) domain. The DFT can be seen as a linear transformation given by a Vandermonde matrix $A_{kj} = (\alpha^{kj})$. The inverse DFT (IDFT), which takes a sequence in the frequency domain and converts it back to the time domain, is given by

$$x_k = n^{-1} \sum_{i=0}^{n-1} x'_i \alpha^{-ik}.$$

Rotating Lists of Encryptions. Consider two sequences $x_0, x_1, \ldots, x_{n-1}$ and $y_0, y_1, \ldots, y_{n-1}$ such that $y_k = x_{k-r}$. The key property is now, with $0 \le k < n$:

$$y'_k = \sum_{j=0}^{n-1} y_j \alpha^{kj} = \sum_{j=0}^{n-1} x_{j-r} \alpha^{kj} = \sum_{j=0}^{n-1} x_j \alpha^{k(j+r)} = \alpha^{rk} x'_k = \beta^k x'_k,$$

where $\beta = \alpha^r$. Hence, if we first apply DFT to a sequence $x_0, x_1, \ldots, x_{n-1}$ yielding $x'_0, x'_1, \ldots, x'_{n-1}$, then compute $y'_0, y'_1, \ldots, y'_{n-1}$ by setting

$$y'_k = \beta^k x'_k = \alpha^{rk} x'_k,$$

for $0 \le k < n$, and finally apply IDFT to obtain sequence $y_0, y_1, \ldots, y_{n-1}$, it follows by construction that $y_k = x_{k-r}$ and thus, the two sequences are a rotation of each other.

We use this approach to perform efficient rotations of list of encryptions by means of a cascade of verifiable rotators. Since the coefficients α^{kj} can be computed publicly, one can apply DFT and IDFT to an encrypted sequence using just the homomorphic properties. These transformations are performed once, at the beginning and at the end of the cascade, respectively. The rotators will pass on transformed sequences between each other.

Concretely, each verifiable rotator will perform the following transformation to a given list of encryptions $\mathsf{E}(x'_0), \ldots, \mathsf{E}(x'_{n-1})$:

$$\mathsf{E}(y'_k) = \mathsf{E}(x'_k)^{\beta^k} (g^{s_k}, h^{s_k}), \text{for } k = 0, 1, \ldots, n-1, \tag{1}$$

with $s_k \in_R \mathbb{Z}_q$ and $\beta = \alpha^r$, $r \in_R \{0, 1, \ldots, n-1\}$. The purpose of the random re-encryptions (g^{s_k}, h^{s_k}) is to hide the rotation offset r being used.

The transformation at the beginning of the cascade of rotators can be done publicly, using the homomorphic property:

$$\mathsf{E}(x'_k) = \prod_{j=0}^{n-1} \mathsf{E}(x_j)^{\alpha^{kj}}.$$

Similarly, the transformation at the end of the cascade, if desired, can be done publicly. Below, we will introduce the use of the Fast Fourier Transform (FFT) algorithm to perform these transformation using $n \log n$ exponentiations only. This way the cost of the transformation at the beginning (and possibly at the end) of the cascade is amortized over the length of the cascade. When the length of the cascade is $\Omega(\log n)$, the work will be $O(n)$ per rotator.

3.2 DFT-Based Protocol

Σ-protocol. To make a rotation verifiable, we provide a proof that the list of encryptions are transformed according to Eq. (1). To this end, a rotator needs to prove that it knows a value $\beta \in \mathbb{Z}_q$ with $\beta^n = 1 \pmod q$ and values $s_0, s_1, \ldots, s_{n-1} \in \mathbb{Z}_q$ such that Eq. (1) holds. We show how this can be done very efficiently using standard techniques.

Let $(a_k, b_k) = X'_k$ and $(d_k, e_k) = Y'_k$ be ElGamal encryptions. The rotator has to prove that it knows $\beta \in \mathbb{Z}_q$ such that $\beta^n = 1$, and values $s_0, \ldots, s_{n-1} \in \mathbb{Z}_q$ such that

$$d_k = a_k^{\beta^k} g^{s_k}, e_k = b_k^{\beta^k} h^{s_k}.$$

To prove that the exponents are of the form β^k, the prover produces auxiliary homomorphic Pedersen commitments $\mathsf{C}(\beta), \mathsf{C}(\beta^2), \ldots, \mathsf{C}(\beta^{n-1})$, and proves that this sequence is indeed a sequence of powers of some β. As a stepping-stone, we use the efficient Σ-protocol for showing that $z = xy$ for commitments $\mathsf{C}(x), \mathsf{C}(y), \mathsf{C}(z)$ (see [5]). Starting with a commitment to β, we prove iteratively the knowledge of its powers as follows. Let $c_0 = \mathsf{C}(1)$, then successively construct the commitments $c_1 = \mathsf{C}(\beta) = c_0^{\beta}\widetilde{h}^{t_0}$, $c_2 = c_1^{\beta}\widetilde{h}^{t_1} = \mathsf{C}(\beta^2), c_3 = c_2^{\beta}\widetilde{h}^{t_2} = \mathsf{C}(\beta^3), \ldots, c_{n-1} = c_{n-2}^{\beta}\widetilde{h}^{t_{n-2}} = \mathsf{C}(\beta^{n-1})$ and apply a Σ protocol to show that one and the same exponent β is used for all these commitments when expressed this way.

To prove that $\beta^n = 1$ holds, we let the rotator compute $c_n = c_{n-1}^{\beta}\widetilde{h}^{t_{n-1}} = \mathsf{C}(\beta^n)$ as well and prove that this is a commitment to 1. This can be done by applying a proof of knowledge of the discrete log of c_n/g with respect to the base $\widetilde{h}$, as $c_n/g = \widetilde{h}^{t^*_{n-1}}$, for some value t^*_{n-1} (see below).

Finally, now given $(a_k, b_k) = X'_k$, $c_k = \mathsf{C}(\beta^k)$, and $(d_k, e_k) = Y'_k$, the rotator has to prove that it knows values s_k, t^*_k, β^k such that

$$c_k = g^{\beta^k}\widetilde{h}^{t^*_k}, d_k = a_k^{\beta^k} g^{s_k}, e_k = b_k^{\beta^k} h^{s_k}.$$

where $t^*_k = \sum_{j=0}^{k} \beta^{k-j} t_j$.

The Σ-protocol which combines all these steps is shown in Fig. 1.

Prover	**Common input:**	**Verifier**
(knows β, s_k s.t. $\beta \in \langle \alpha \rangle$,	$(a_k, b_k) = X'_k$	
$d_k = a_k^{\beta^k} g^{s_k}, e_k = b_k^{\beta^k} h^{s_k}$)	$(d_k, e_k) = Y'_k$	
$\widetilde{m}, b \in_R \mathbb{Z}_q$		
$\widetilde{C} = \widetilde{h}^{\widetilde{m}}$		
$m_k, t_k, u_k, v_k, \widetilde{m}_k \in_R \mathbb{Z}_q$		
$c_{k+1} = c_k^{\beta} \widetilde{h}^{t_k}$		
$C_{k+1} = c_k^{b} \widetilde{h}^{m_k}$		
$\widetilde{C}_k = g^{u_k} \widetilde{h}^{\widetilde{m}_k}$		
$D_k = a_k^{u_k} g^{v_k}$		
$E_k = b_k^{u_k} h^{v_k}$		
	$\xrightarrow{\widetilde{C}, \{c_{k+1}, C_{k+1}, D_k, E_k, \widetilde{C}_k\}_{k=0}^{n-1}}$	
$t^*_{-1} = 0$	$\xleftarrow{\lambda}$	$\lambda \in_R \mathbb{Z}_q$
$t^*_k = \sum_{j=0}^{k} \beta^{k-j} t_j$		
$\sigma = b + \lambda\beta,$		
$\eta = \widetilde{m} + \lambda t^*_{n-1}$		
$\psi_k = m_k + \lambda t_k$		
$\mu_k = u_k + \lambda \beta^k$		
$\nu_k = v_k + \lambda s_k$		
$\rho_k = \widetilde{m}_k + \lambda t^*_{k-1}$		
	$\xrightarrow{\sigma, \eta, \{\psi_k, \mu_k, \nu_k, \rho_k\}_{k=0}^{n-1}}$	$\widetilde{h}^{\eta} \overset{?}{=} \widetilde{C}(c_n/g)^{\lambda}$
		$c_k^{\sigma} \widetilde{h}^{\psi_k} \overset{?}{=} C_{k+1} c_{k+1}^{\lambda}$
		$g^{\mu_k} \widetilde{h}^{\rho_k} \overset{?}{=} \widetilde{C}_k c_k^{\lambda}$
		$a_k^{\mu_k} g^{\nu_k} \overset{?}{=} D_k d_k^{\lambda}$
		$b_k^{\mu_k} h^{\nu_k} \overset{?}{=} E_k e_k^{\lambda}$

Fig. 1. Proof of a rotation in the transformed domain of ElGamal encryptions, where $c_0 = g$ and k runs from 0 to $n-1$

Efficiency. From Fig. 1 we can see that the generation of the proof requires $5n$ double exponentiations, while the verification requires $4n$ triple exponentiations.

An issue in the efficiency of a cascade of rotators is the computation of the DFT and possibly the IDFT under the encryptions. Although the DFT and the IDFT can be applied just using homomorphic properties, these steps may be a computational bottleneck as they involve the computation of n n-way exponentiations (which naively costs n^2 single exponentiations). However, e.g., assuming that n is an integral power of 2, one can apply the Fast Fourier Transform algorithm to reduce the computational complexity to $O(n \log n)$ exponentiations.

To illustrate the application of the FFT algorithm, assuming that $n = 2^{\ell}$, then the DFT computation under encryptions can be split in the following way:

$$\mathsf{E}(x'_k) = \prod_{j=0}^{n-1} \mathsf{E}(x_j)^{\alpha^{kj}} = \prod_{j=0}^{n/2-1} \mathsf{E}(x_{2j})^{\alpha^{k2j}} \mathsf{E}(x_{2j+1})^{\alpha^{k(2j+1)}}$$

$$= \prod_{j=0}^{n/2-1} \mathsf{E}(x_{2j})^{\alpha^{2kj}} \left(\prod_{j=0}^{n/2-1} \mathsf{E}(x_{2j+1})^{\alpha^{2kj}} \right)^{\alpha^k} .$$

Noting that α^2 is a $n/2$-th root of unity modulo q, we have reduced the problem of computing the DFT of a sequence of length n to solving two DFTs of sequences of half length. We note that these two DFTs can be computed in parallel because they are independent of each other. If t_n is the number of exponentiations required to compute one element of the DFT of length n, then we get that $t_n = t_{n/2} + 1$. Finally, the total number of exponentiations is nt_n. In particular when n is a power of two, $n \log n$ exponentiations are required in total.

When using a cascade of rotators, the rotators will keep the sequence in the frequency domain. If so desired, the DFT and its inverse need to be applied only before and after the entire cascade. With this observation, we only need to transform at the first and final rotator in a cascade of rotators, and it helps as one can average the total work for transforms over the length of the cascade. In the special case of a cascade of n rotators, the work per rotator will be linear.

In some applications it may be reasonable to assume that the input and output lists are in transformed form. Another observation is that if the final output list of a cascade is going to be decrypted anyway, one may leave out the inverse DFT, and decrypt the transformed sequence. Then one can perform the inverse DFT on the plaintexts (depending on the homomorphic cryptosystem this may give an advantage, for ElGamal one would still need exponentiations).

Extensions. Using n-th roots of unity in extension fields of $\mathbb{Z}_q$ with the condition that $n \mid q-1$ can be weakened. Given a fixed n, we can adjust the extension field that we work with. This will come at the expense of increased communication and computation.

4 General Solution

The DFT-based protocol presented above puts some constraints on the parameters involved. In this section we present a different approach that does not such constraints.

We use a two-stage approach, similar to the approach for verifiable shuffles in [19,12,15]. We first present an auxiliary protocol to prove that a list of known committed values has been rotated. In the proofs of security we also use the Schwartz-Zippel lemma.

Lemma 1 (Schwartz-Zippel). *Let p be a multivariate polynomial of degree d over $\mathbb{Z}_q$. Then the probability that $p(x_1, x_2, \ldots, x_m) = 0$ for randomly chosen $x_1, x_2, \ldots, x_m$ over $\mathbb{Z}_q$ is at most d/q.*

We will use it for the case that $d = 1$ to test that two sequences have the same values. That is, given $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1})$, then if

$$\sum_{j=0}^{n-1} \beta_j x_j = \sum_{j=0}^{n-1} \beta_j y_j,$$

for $\beta_0, \beta_1, \ldots, \beta_{n-1} \in_R \mathbb{Z}_q$, it follows that $(x_0, x_1, \ldots, x_{n-1}) \neq (y_0, y_1, \ldots, y_{n-1})$ with probability at most $1/q$.

4.1 Rotation of Known Committed Values

Let $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ be some publicly known values. The prover produces some commitments $c_0, c_1, \ldots, c_{n-1}$, for which it will prove knowledge of an offset r and randomizers $s_0, s_1, \ldots, s_{n-1}$ satisfying:

$$c_k = g^{\alpha_{k-r}} \widetilde{h}^{s_k}, \text{ for } k = 0, 1, \ldots, n-1. \tag{2}$$

Building Blocks. We will use a Σ-protocol DL-OR$(G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1})$ to prove that, given randomly selected challenges $\beta_0, \beta_1, \ldots, \beta_{n-1}$, $G = \prod_{j=0}^{n-1} c_j^{\beta_j}$ is a commitment to one of γ_0, γ_1, $\ldots, \gamma_{n-1}$ as defined in the protocol of Fig. 2.

Intuitively, the initial commitments $c_0, c_1, \ldots, c_{n-1}$ commit to a rotation of $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ if the prover knows the inner product of a random vector with one of the possible rotated versions of the vector with the α-values.

Theorem 1. *The protocol* PUB-ROT*of Fig. 2 is a special honest-verifier zero-knowledge proof of knowledge with witness-extended emulation that a prover knows an integer r, $0 \leq r < n$ and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that Eq. (2) holds.*

Proof. To show completeness, let commitments $c_0, c_1, \ldots, c_{n-1}$ be defined as in Eq. (2). For challenge $\beta_0, \beta_1, \ldots, \beta_{n-1}$ the values $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$ and the commitment G are then computed. Observe that G is a commitment to the value $\sum_{j=0}^{n-1} \alpha_{j-r}\beta_j$ due to the homomorphic properties, which is equal to γ_r. As G is a commitment to γ_r, the proof DL-OR will be accepted by the verifier.

To show SHVZK, we construct a simulator as follows. Given challenges $\beta_0, \beta_1, \ldots, \beta_{n-1}$ and λ, it follows that the values $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$ and G can be computed as specified by the protocol. From the SHVZK property of the Σ-protocol DL-OR$(G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1})$, there exists a simulator which produces an indistinguishable view when it is given challenge λ and all public information, namely $G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1}$.

We show knowledge soundness through witness-extended emulation. Witness extended emulation can be shown using standard techniques as done in [12,14,15]. The idea is to describe an emulator that runs in expected polynomial time producing a view indistinguishable from that of the protocol and at the same time gives a witness with the same probability as an adversarial prover produces an

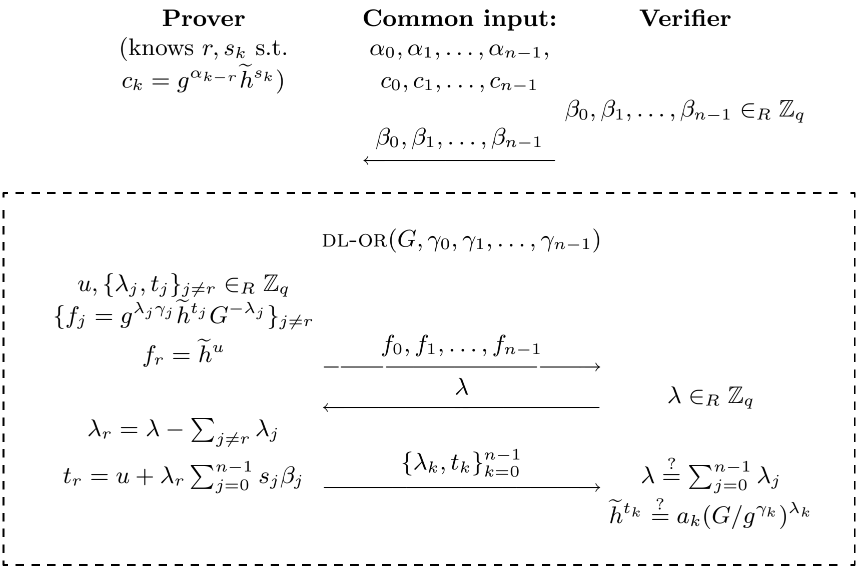

Prover
(knows r, s_k s.t.
$c_k = g^{\alpha_k - r}\widetilde{h}^{s_k}$)
Common input:
$\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$,
$c_0, c_1, \ldots, c_{n-1}$
Verifier
$\beta_0, \beta_1, \ldots, \beta_{n-1} \in_R \mathbb{Z}_q$
$\xleftarrow{\beta_0, \beta_1, \ldots, \beta_{n-1}}$
DL-OR$(G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1})$
$u, \{\lambda_j, t_j\}_{j \neq r} \in_R \mathbb{Z}_q$
$\{f_j = g^{\lambda_j \gamma_j}\widetilde{h}^{t_j} G^{-\lambda_j}\}_{j \neq r}$
$f_r = \widetilde{h}^u$
$\xrightarrow{f_0, f_1, \ldots, f_{n-1}}$
$\xleftarrow{\lambda}$
$\lambda \in_R \mathbb{Z}_q$
$\lambda_r = \lambda - \sum_{j \neq r} \lambda_j$
$t_r = u + \lambda_r \sum_{j=0}^{n-1} s_j \beta_j$
$\xrightarrow{\{\lambda_k, t_k\}_{k=0}^{n-1}}$
$\lambda \overset{?}{=} \sum_{j=0}^{n-1} \lambda_j$
$\widetilde{h}^{t_k} \overset{?}{=} a_k (G/g^{\gamma_k})^{\lambda_k}$

Fig. 2. Protocol PUB-ROT for proving a rotation of known committed values, where $\gamma_k = \sum_{j=0}^{n-1} \alpha_{j-k}\beta_j$, for $k = 0, 1, \ldots, n-1$, and $G = \prod_{j=0}^{n-1} c_j^{\beta_j}$

accepting conversation. This is achieved by first letting the prover run on random selected challenges, and if it is accepting, the witness is extracted using rewinding techniques until a sufficient number of transcripts have been obtained. As we use the Σ-protocol DL-OR as building block, we will make use of its witness-extended emulator, denoted as $E_{\text{DL-OR}}$.

The description of the emulator E is as follows. E picks random challenges $\overline{\beta}^{(1)} = \beta_0^{(1)}, \beta_1^{(1)}, \ldots, \beta_{n-1}^{(1)}$ and plays the witness-extended emulator for DL-OR, $E_{\text{DL-OR}}$, on $G^{(1)}$ and $\overline{\gamma}^{(1)}$ as defined in the protocol. This invocation will give a transcript of DL-OR along with a witness if the transcript is accepting. If the transcript is not accepting, E outputs no witness along with $(\overline{\beta}^{(1)}, view^{(1)})$ as the view of the protocol.

Otherwise, we have a valid transcript and a witness of DL-OR. Namely, an integer $r^{(1)}$ and a randomizer $t^{(1)}$ such that $G^{(1)} = \mathsf{C}(\gamma_{r^{(1)}}^{(1)}, t^{(1)})$.

The prover is rewound and fresh challenges $\beta_0^{(i)}, \beta_1^{(i)}, \ldots, \beta_{n-1}^{(i)}$ are chosen, for $i = 2, 3, \ldots, n$ until n valid transcripts and n witnesses for DL-OR are obtained, by subsequently invoking $E_{\text{DL-OR}}$. From all these n witnesses, E is able to compute the witness. After this is done, the output of E is the witness plus $\overline{\beta}^{(1)}$ attached to the first view obtained from $E_{\text{DL-OR}}$.

We first show how E manages to get a witness. Then, we show that this extractor runs in expected polynomial time and argue that E gives an accepting view plus a witness with essentially the same probability that the prover is able to produce accepting conversations.

From all of witnesses obtained, we get the following equalities, with $1 \leq i \leq n$.

$$G^{(i)} = \mathsf{C}(\gamma^{(i)}_{r^{(i)}}, t^{(i)}). \tag{3}$$

Also, as specified by the protocol, we have

$$G^{(i)} = \prod_{j=0}^{n-1} c_j^{\beta_j^{(i)}}. \tag{4}$$

By construction, the vectors $\overline{\beta}^{(i)}$ with $i = 1, 2 \ldots, n$ are linearly independent with overwhelming probability (for n polynomial in the size of q). The linear independence of the vectors $\overline{\beta}^{(i)}$ implies the existence of elements $d_{k,i}$ such that $\sum_{k=1}^{n} \overline{\beta}^{(i)} d_{k,i}$ is the $(k+1)$-st standard unit vector of $\mathbb{Z}_q^n$, for $k = 0, 1, \ldots, n-1$.

This implies that

$$c_k = \prod_{k=1}^{n} \big(\prod_{j=0}^{n-1} c_j^{\beta_j^{(i)}} \big)^{d_{k,i}}.$$

By Eq. (4), it in turn implies that

$$c_k = \prod_{k=1}^{n} (G^{(i)})^{d_{k,i}} = \prod_{k=1}^{n} \mathsf{C}(\gamma^{(i)}_{r^{(i)}}, t^{(i)})^{d_{k,i}} = \prod_{k=1}^{n} \mathsf{C}(d_{k,i}\gamma^{(i)}_{r^{(i)}}, d_{k,i}t^{(i)}) =$$
$$= \mathsf{C}(\sum_{k=1}^{n} d_{k,i}\gamma^{(i)}_{r^{(i)}}, \sum_{k=0}^{n-1} d_{k,i}t^{(i)})$$

Therefore, we find an opening of the commitment c_k. Let $\widetilde{\alpha}_k = \sum_{k=1}^{n} d_{k,i}\gamma^{(i)}_{r^{(i)}}$ and $s_k = \sum_{k=1}^{n} d_{k,i}t^{(i)}$.

We now prove that $\widetilde{\alpha}_k$ are a rotated version of the elements α_k. From Eqs. (3) and (4), and using the binding property of the commitment scheme, it follows that

$$\sum_{j=0}^{n-1} \beta_j^{(i)} \widetilde{\alpha}_j = \gamma^{(i)}_{r^{(i)}} = \sum_{j=0}^{n-1} \beta_j^{(i)} \alpha_{j-r^{(i)}},$$

for all $i = 1, 2, \ldots, n$.

As the $\beta_j^{(i)}$ are randomly chosen, we conclude, using the Schwartz-Zippel lemma, that with overwhelming probability $\widetilde{\alpha}_j = \alpha_{j-r^{(i)}}$ holds. This shows that indeed the committed values in c_k are a rotated list of the α_k. Note that this allows us to conclude that $r^{(i)} = r^{(j)}$ for all $i \neq j$.

In summary, we have found an integer r and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that $c_k = \mathsf{C}(\alpha_{k-r}, s_k)$ which is actually the witness for PUB-ROT.

We now argue that E runs in expected polynomial time. Let $\widetilde{\varepsilon}$ denote the probability that querying $E_{\text{DL-OR}}$ on independently random selected $\overline{\beta}$'s results in an accepting transcript. Then getting an accepting transcript and therefore a witness will take expected time $(1/\widetilde{\varepsilon})T$, where T is the expected running time of $E_{\text{DL-OR}}$. Therefore, the total expected running time of the repeated calls to $E_{\text{DL-OR}}$ made by E is $T + \widetilde{\varepsilon}(n-1)T/\widetilde{\varepsilon} = nT$.

Let ε denote the probability that a real prover gives an accepting proof of PUB-ROT. To check that the difference between ε and $\widetilde{\varepsilon}$ is negligible, we observe that ε can be seen as a weighted sum of probabilities $\sum_{\overline{\beta}}(1/q^n)\varepsilon_{\text{DL-OR}(\overline{\beta})}$, where $\varepsilon_{\text{DL-OR}(\overline{\beta})}$ is the success probability of the prover in DL-OR when it got challenges $\overline{\beta}$. On the other hand, the probability that we get an accepting answer in the first query to $E_{\text{DL-OR}}$ in E happens with probability $\sum_{\overline{\beta}}(1/q^n)\widetilde{\varepsilon}_{\text{DL-OR}(\overline{\beta})}$ where $\widetilde{\varepsilon}_{\text{DL-OR}(\overline{\beta})}$ denotes the probability that $E_{\text{DL-OR}}$ gives an accepting conversation on relation induced by challenges $\overline{\beta}$. We can conclude that $|\varepsilon - \widetilde{\varepsilon}|$ is negligible, using the fact that $|\varepsilon_{\text{DL-OR}(\overline{\beta})} - \widetilde{\varepsilon}_{\text{DL-OR}(\overline{\beta})}|$ is negligible which is true by definition of witness extended emulator for DL-OR. □

Efficiency. After the verifier submitted the challenges $\overline{\beta}$, the commitment G is computed using an n-way exponentiation, which roughly costs $n/2$ double exponentiations. Then the prover has to give the DL-OR proof based on G and the γ's. The latter requires n double exponentiations for the prover and the verifier separately. Therefore, we have $1.5n$ double exponentiations to produce the proof, and $1.5n$ to verify it.

4.2 General Rotation

Given two lists of encryptions, $X_0, X_1, \ldots, X_{n-1}$ and $Y_0, Y_1, \ldots, Y_{n-1}$, the prover shows that it knows an integer r, $0 \leq r \leq n-1$, and randomizers $s_0, s_1, \ldots, s_{n-1}$ satisfying:

$$Y_k = X_{k-r}(g^{s_k}, h^{s_k}) \text{ for } k = 0, 1, \ldots, n-1. \tag{5}$$

For this task, we propose the protocol presented in Fig. 3.

Building Blocks. In this protocol, besides of the proof for rotation of known contents, we use an additional proof of knowledge. The proof EQ-EXP(h, Y, A) allows the prover to show the knowledge of α, r, t such that $h = \mathsf{C}(\alpha, r)$ and $A = Y^{\alpha}\mathsf{E}(0, t)$, where Y is an encryption. This proof is actually a basic Σ-protocol.

Before going into the security analysis, we note that the protocol requires 6 rounds of interaction. This is because some rounds in the involved building blocks can be run in parallel. Namely, we can combine the first set of messages from EQ-EXP with the first answer from the prover in the protocol description of Fig. 3. Also the first round of PUB-ROT can be directly connected with the challenge coming from EQ-EXP. In the security analysis, however, we will use the modular description of the protocol.

Theorem 2. *The protocol* GEN-ROT *is a special honest-verifier zero-knowledge proof of knowledge with witness extended emulation that a prover knows an integer r, $0 \leq r < n$ and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that Eq. (5) holds.*

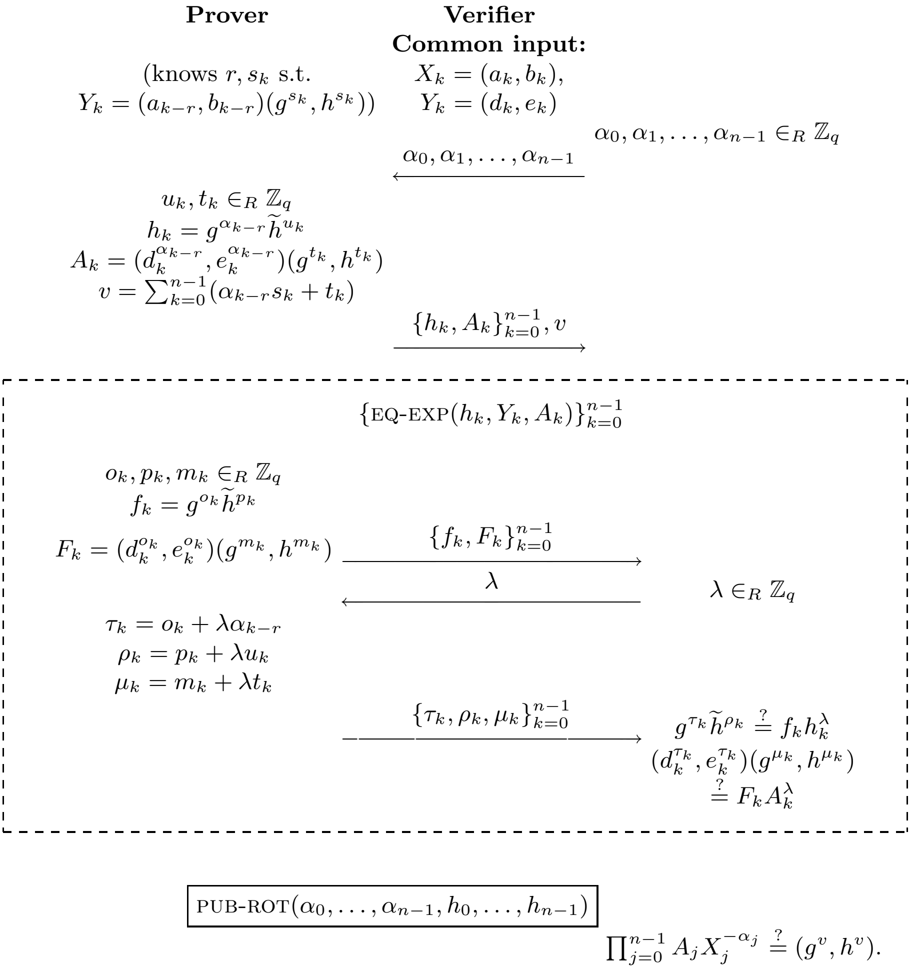

Prover
Verifier
Common input:
(knows r, s_k s.t.
$Y_k = (a_{k-r}, b_{k-r})(g^{s_k}, h^{s_k}))$
$X_k = (a_k, b_k)$,
$Y_k = (d_k, e_k)$
$\alpha_0, \alpha_1, \dots, \alpha_{n-1} \in_R \mathbb{Z}_q$
$\xleftarrow{\alpha_0, \alpha_1, \dots, \alpha_{n-1}}$
$u_k, t_k \in_R \mathbb{Z}_q$
$h_k = g^{\alpha_{k-r}} \widetilde{h}^{u_k}$
$A_k = (d_k^{\alpha_{k-r}}, e_k^{\alpha_{k-r}})(g^{t_k}, h^{t_k})$
$v = \sum_{k=0}^{n-1} (\alpha_{k-r} s_k + t_k)$
$\xrightarrow{\{h_k, A_k\}_{k=0}^{n-1}, v}$
$\{\text{EQ-EXP}(h_k, Y_k, A_k)\}_{k=0}^{n-1}$
$o_k, p_k, m_k \in_R \mathbb{Z}_q$
$f_k = g^{o_k} \widetilde{h}^{p_k}$
$F_k = (d_k^{o_k}, e_k^{o_k})(g^{m_k}, h^{m_k})$
$\xrightarrow{\{f_k, F_k\}_{k=0}^{n-1}}$
$\xleftarrow{\lambda}$
$\lambda \in_R \mathbb{Z}_q$
$\tau_k = o_k + \lambda \alpha_{k-r}$
$\rho_k = p_k + \lambda u_k$
$\mu_k = m_k + \lambda t_k$
$\xrightarrow{\{\tau_k, \rho_k, \mu_k\}_{k=0}^{n-1}}$
$g^{\tau_k} \widetilde{h}^{\rho_k} \stackrel{?}{=} f_k h_k^{\lambda}$
$(d_k^{\tau_k}, e_k^{\tau_k})(g^{\mu_k}, h^{\mu_k})$
$\stackrel{?}{=} F_k A_k^{\lambda}$
$\boxed{\text{PUB-ROT}(\alpha_0, \dots, \alpha_{n-1}, h_0, \dots, h_{n-1})}$
$\prod_{j=0}^{n-1} A_j X_j^{-\alpha_j} \stackrel{?}{=} (g^v, h^v)$.

Fig. 3. Protocol GEN-ROT for proving a rotation of ElGamal encryptions, where k runs from 0 to $n-1$

Proof. To prove that the protocol for general rotation is SHVZK, we will construct a simulator, given $X_0, X_1, \dots, X_{n-1}, Y_0, Y_1, \dots, Y_{n-1}$, and the random challenges $\alpha_0, \alpha_1, \dots, \alpha_{n-1}$ (along with all the challenges for PUB-ROT and the one for EQ-EXP).

The simulator runs as follows. First, it computes $A_k = X_k^{\alpha_k} \mathsf{E}(0, v_k)$ for random v_k. The assignment for A_k is somewhat arbitrary and there are various ways to do it, as long as it guarantees that the proof is accepting.

Now, the simulator creates commitments $h_0, h_1, \dots, h_{n-1}$ all of them to 0 (or any value). Despite this choice, the simulators for both PUB-ROT and EQ-EXP will still produce transcripts that are indistinguishable from real ones. Combining the obtained transcripts with the computed array of commitments h_k, encryptions

A_k, and the value v, the resulting view is indistinguishable to a real transcript of GEN-ROT.

For witness-extended emulation we construct the emulator E as follows. Given challenges $\overline{\alpha}^{(1)} = (\alpha_0^{(1)}, \alpha_1^{(1)}, \ldots, \alpha_{n-1}^{(1)})$, E runs the prover until it gets responses $\overline{h}^{(1)}, \overline{k}^{(1)}, v^{(1)}$, after this, the witness-extended emulators $E_{\text{EQ-EXP}}$ and $E_{\text{PUB-ROT}}$ are run. With some probability $\widetilde{\epsilon}$, they will produce an accepting transcript. If this is not the case, E outputs no witness along with the transcripts obtained from the witness-emulators and the prover. Else, E will rewind the prover and choose fresh random challenges $\overline{\alpha}^{(i)}$, until n accepting conversations are obtained in total. For each of them, let $\alpha_0^{(i)}, \alpha_1^{(i)}, \ldots, \alpha_{n-1}^{(i)}$ denote the challenges selected by E. For each of these cases, respective witness-extended emulators for PUB-ROT and EQ-EXP's are run in order to extract their witnesses.

This is the information necessary to compute the witness for GEN-ROT. We show now, how E is able to compute an integer r and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that Eq. (5) holds.

We have for a particular iteration of $i = 1, 2, \ldots, n$, and for every j with $0 \leq j \leq n-1$ that

$$\begin{aligned} A_j^{(i)} &= Y_j^{b_j^{(i)}} \mathsf{E}(0, t_j^{(i)}), \text{ and} \\ h_j^{(i)} &= \mathsf{C}(b_j^{(i)}, u_j^{(i)}). \end{aligned} \tag{6}$$

for some values $\overline{b}, \overline{t}$ and $\overline{u}$.

We have also witnesses for PUB-ROT. That is, there exist integers $r^{(i)}$ and randomizers $w_j^{(i)}$ such that

$$h_j^{(i)} = \mathsf{C}(\alpha_{j-r^{(i)}}^{(i)}, w_j^{(i)}). \tag{7}$$

From (6) and (7), due to the binding property of the commitment scheme, it follows that $b_j^{(i)} = \alpha_{j-r^{(i)}}^{(i)}$ and $u_j^{(i)} = w_j^{(i)}$. One of the consequences is that

$$A_j^{(i)} = Y_j^{\alpha_{j-r^{(i)}}^{(i)}} \mathsf{E}(0, t_j^{(i)}). \tag{8}$$

As all these equations are based on accepting conversations then the verifications of the proof passes, and therefore,

$$\frac{\prod_{j=0}^{n-1} A_k^{(i)}}{\prod_{j=0}^{n-1} X_j^{\alpha_j^{(i)}}} = \frac{\prod_{j=0}^{n-1} Y_j^{\alpha_{j-r^{(i)}}^{(i)}} \mathsf{E}(0, t_j^{(i)})}{\prod_{j=0}^{n-1} X_j^{\alpha_j^{(i)}}} = \mathsf{E}(0, v^{(i)}). \tag{9}$$

This implies that,

$$\prod_{j=0}^{n-1} \left(\frac{Y_{j-r^{(i)}}}{X_j} \right)^{\alpha_j^{(i)}} = \mathsf{E}(0, v^{(i)} - \sum_{j=0}^{n-1} t_j^{(i)})$$

From this last equality, we can conclude using the Schwartz-Zippel lemma that the X_k and Y_k encrypt the same elements up to a rotation of these elements, implying also that $r^{(i)} = r^{(j)}$ for all $i \neq j$. We denote that rotation offset as r.[3]

Let $Z_j = \frac{Y_{j-r}}{X_j}$ and $\widetilde{n}_k = v^{(i)} - \sum_{j=0}^{n-1} t_j^{(i)}$. The aim now is to extract the randomness used in Z_j which is by homomorphic properties the randomness used to re-blind the rotated list Y_j.

As the challenges throughout the extraction process are selected independently at random, it implies that the vectors $\overline{\alpha}^{(i)}$ are linearly independent. This enables the existence of elements $d_{k,i}$ such that $\sum_{j=0}^{n-1} d_{k,i}\overline{\alpha}^{(i)}$ is the $(k+1)$-st standard unit vector in $\mathbb{Z}_q^n$, for $k = 0, 1, \ldots, n-1$. Therefore,

$$Z_k = \prod_{k=1}^{n} \left(\prod_{j=0}^{n-1} Z_j^{\alpha_j^{(i)}} \right)^{d_{k,i}} = \mathsf{E}(0, \sum_{k=1}^{n} d_{k,i}\widetilde{n}_k),$$

that enables us to extract $s_k = \sum_{k=1}^{n} d_{k,i}\widetilde{n}_k$.

The way of arguing that E runs in expected polynomial time is similar to the protocol for known contents. □

Efficiency. First, we calculate the computational cost to prove one invocation of EQ-EXP. This requires 3 double exponentiations and 3 triple exponentiations to produce, and 3 triple exponentiations to verify the proof.

Now, the number of exponentiations required to compute GEN-ROT. The prover must compute the commitments h_k, costing n double exponentiations. The encryptions A_k need $2n$ more double exponentiations. Then, $3n$ for EQ-EXP and $1.5n$ for PUB-ROT, totalling $7.5n$ double exponentiations for the prover to give a proof.

The verifier needs to check the EQ-EXP and PUB-ROT which requires $3n$ triple exponentiations and $1.5n$ double exponentiations, respectively. The verification at the end of the proof requires 2 more n-way exponentiation. Then, the total cost for the verifier is $5.5n$ double exponentiations.

5 Comparison and Concluding Remarks

To the best of our knowledge, the protocol by Reiter and Wang [21] is the only one that presents protocols for proving a rotation. Their construction works in general given that appropriate homomorphic cryptosystem has been set up. In fact, they show a protocol to prove that a permutation has been applied making use of 4 invocations to the proof of shuffle.

Table 1 presents a comparison of the computational complexities for various protocols for rotation. In all cases, we assume that the protocols are based on homomorphic ElGamal and Pedersen commitments. For simplicity, we count

[3] If this is not the case, this must be because the plaintexts in X_k are all the same for all k. This is not a problem though, we just take $r = 0$ and the rest of the procedure will still be applicable.

Table 1. Performance figures for proving a rotation

Protocol	Prove	Verify	Rounds
Loop permutations [21]	$16n$	$10n$	30
Proposed: DFT-based	$5n$	$4n$	3
Proposed: General	$7.5n$	$5.5n$	6

the number of double exponentiations. Additionally, for the protocol in [21] we assume that they use the proof of shuffle by Groth [12] which is one of the most efficient.

Acknowledgements. We thank anonymous reviewers for their valuable comments. The fourth author is supported by the research program Sentinels (`http://www.sentinels.nl`). Sentinels is being financed by Technology Foundation STW, the Dutch Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

References

1. Atallah, M.J., Blanton, M., Frikken, K.B., Li, J.: Efficient correlated action selection. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 296–310. Springer, Heidelberg (2006)
2. Blake, I., Kolesnikov, V.: Strong conditional oblivious transfer and computing on intervals. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 515–529. Springer, Heidelberg (2004)
3. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
4. Cramer, R.: Modular Design of Secure yet Practical Cryptographic Protocols. PhD thesis, Universiteit van Amsterdam, Netherlands (1997)
5. Cramer, R., Damgård, I.: Zero-knowledge for finite field arithmetic. Or: Can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998)
6. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
7. Damgård, I., Fujisaki, E.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
8. Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for on-line auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007)
9. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory IT-31, 469–472 (1985)
10. Garay, J., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007)

11. Goldreich, O.: Foundations of Cryptography. Cambridge University Press, Cambridge (2001)
12. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002); http://eprint.iacr.org/2005/246
13. Groth, J.: Evaluating security of voting schemes in the universal composability framework. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 46–60. Springer, Heidelberg (2004)
14. Groth, J.: Honest Verifier Zero-Knowledge Arguments Applied. PhD thesis, University of Aarhus (2004)
15. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008)
16. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
17. Li, J., Atallah, M.: Secure and private collaborative linear programming. Collaborative Computing: Networking, Applications and Worksharing, 2006. CollaborateCom 2006, pp. 1–8 (2006)
18. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. Journal of Cryptology 16(3), 143–184 (2001)
19. Neff, C.: A verifiable secret shuffle and its application to e-voting. In: 8th ACM conference on Computer and Communications Security – CCS 2001, pp. 116–125. ACM Press, New York (2001)
20. Reistad, T., Toft, T.: Secret sharing comparison by transformation and rotation. In: Pre-Proceedings of the International Conference on Information Theoretic Security–ICITS 2007. LNCS. Springer, Heidelberg (2007) (to appear)
21. Reiter, M.K., Wang, X.: Fragile mixing. In: CCS 2004: Proceedings of the 11th ACM conference on Computer and communications security, pp. 227–235. ACM Press, New York (2004)
22. Ryan, P.: Prêt-à-Voter with Paillier encryption, Technical Report CS-TR No 965, School of Computing Science, Newcastle University (2006), http://www.cs.ncl.ac.uk/publications/trs/papers/965.pdf
23. Ryan, P., Schneider, F.: Prêt-à-Voter with re-encryption mixes. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 313–326. Springer, Heidelberg (2006)
24. Sako, K., Killian, J.: Receipt-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
25. Straus, E.: Addition chains of vectors (problem 5125). American Mathematical Monthly 71, 806–808 (1964)
26. Wen, R., Buckland, R.: Mix and test counting for the alternative vote electoral system (2008); presented at WISSec 2008
27. Yao, A.: How to generate and exchange secrets. In: 27th IEEE Symposium on Foundations of Computer Science, pp. 162–168 (1986)

A Practical Key Recovery Attack on Basic TCHo[⋆]

Mathias Herrmann[1] and Gregor Leander[2]

[1] Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr-University Bochum
Germany
mathias.herrmann@rub.de

[2] Department of Mathematics
Technical University of Denmark
Denmark
g.leander@mat.dtu.dk

Abstract. TCHo is a public key encryption scheme based on a stream cipher component, which is particular suitable for low cost devices like RFIDs. In its basic version, TCHo offers no IND-CCA2 security, but the authors suggest to use a generic hybrid construction to achieve this security level. The implementation of this method however, significantly increases the hardware complexity of TCHo and thus annihilates the advantage of being suitable for low cost devices. In this paper we show, that TCHo cannot be used without this construction. We present a chosen ciphertext attack on basic TCHo that recovers the secret key after approximately $d^{3/2}$ decryptions, where d is the number of bits of the secret key polynomial. The entropy of the secret key is $\log_2 \binom{d}{w}$, where w is the weight of the secret key polynomial, and w is usually small compared to d. In particular, we can break all of the parameters proposed for TCHo within hours on a standard PC.

Keywords: TCHo, chosen ciphertext attack, stream cipher.

1 Introduction

Since the invention of public key cryptography many different crypto systems have been presented. The most popular systems are either based on the hardness of factoring large integers or related problems (e.g. RSA) or computing discrete logarithms in various groups (e.g. DSA, ECDSA). While these schemes are an excellent and preferred choice in almost all applications, there is still a strong need for alternative systems based on other (supposedly) hard problems.

[⋆] The work described in this paper has been supported by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. Moreover, this research was partly supported by the German Research Foundation (DFG) as part of the project MA 2536/3-1.

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 411–424, 2009.

This is mainly due to the following two reasons. The first reason is that most of the standard schemes are not suitable for very constraint environments like RFID tags and sensor networks. This problem becomes even more pressing when looking at the next century's IT landscape, where a massive deployment of tiny computer devices is anticipated and thus the need for extremely low cost public key cryptography will increase significantly. The second — and quite unrelated — reason is that most popular public key crypto systems like RSA, DSA and ECDSA will be broken if quantum computers with sufficiently many qubits can be built (see [9]). Thus, it is important to search for public key crypto systems that have the potential to resist future quantum computers.

Those two reasons outlined above inspired Finiasz and Vaudenay to develop the crypto system TCHo [7]. The original version of TCHo has been revised by Aumasson, Finiasz, Meier and Vaudenay (see [3]). This revision was done mainly to improve the efficiency of the original scheme and we refer to TCHo as defined in [3].

TCHo is a public key encryption scheme based on a stream cipher component. TCHo uses mainly hardware friendly operations and is therefore suitable for low cost devices. Its security is based on the problem of finding a low weight multiple of a given polynomial in $\mathbb{F}_2[x]$ (LWPM for short). The public key of TCHo is a high degree polynomial $P \in \mathbb{F}_2[x]$ and the secret key K is a sparse, or low weight, multiple of P. The LWPM problem is of importance in syndrome decoding [5], stream cipher analysis and efficient finite field arithmetics [4]. In [2] El Aimani and von zur Gathen provide an algorithm to solve this problem based on lattice basis reduction and furthermore give an overview of other possible approaches to tackle LWPM. Yet, the suggested parameters of TCHo cannot be broken by any of those attacks.

In this paper we present a chosen ciphertext attack on TCHo that recovers the secret key after roughly $d^{3/2}$ decryptions, where d is the degree of the secret polynomial. In particular all proposed parameters of TCHo given in [3] can be broken within hours on a standard PC. Our attack recovers consecutively all bits of the secret key by decrypting pairs of ciphertexts with a carefully chosen difference. The choice of the difference as well as the choice of the ciphertext depend on all key bits recovered so far. This property of our attack is of independent interest, as it is one of the rare occasions where an attack on a public key crypto system is actually inherently adaptive with respect to the information gained so far. To clarify, we do not solve the problem of finding low weight multiples of a given polynomial efficiently, but rather provide an efficient method to extract this low weight polynomial given a decryption oracle.

It should be noted that the designers do not claim that TCHo is IND-CCA2 secure. On the contrary, as shown in [3] TCHo is clearly not IND-CCA2 secure since it is, just like RSA, trivially malleable. Given an encryption y for a message m and a second message m', it is easy to construct an encryption for $m \oplus m'$. However, as opposed to the trivial IND-CCA2 attack on TCHo that recovers the message our CCA1 attack recovers the secret key.

In [3] the authors propose to use the revised Fujisaki-Okamoto [8] construction from [1] to transform TCHo into a IND-CCA secure scheme. Clearly, this scheme is not affected by our attack. However, this transformation comes with an additional overhead in the ciphertext length as well as a non negligable overhead due to the fact that a practical implementation of the Fujisaki-Okamoto construction requires to implement at least one secure hash function. Following [6], the best known SHA-1 (resp. SHA-256) implementation requires approximately 8.000 GE (resp. 11.000 GE) which, based on the estimation in [3], would almost double the hardware implementation cost for TCHo . Moreover, this transformation is only efficient in the case where instead of using a truly random number generator for the encryption of TCHo a pseudo random number generator is used, which further increases the hardware complexity. Our result implies that TCHo cannot be used without the Fujisaki-Okamoto transformation. This in turn implies that the efficiency gain for low cost hardware devices compared to well established public key crypto systems like ECC vanishes.

One the positive side, our results can also be interpreted as an indication that breaking TCHo is equivalent to solving the low weight multiple problem.

Finally, our attack is based on a new technique that can be seen as an adaptive differential attack on public key systems. We believe that this technique can be useful for the cryptanalysis of other schemes as well.

The paper is organized as follows: In Section 2 we recall the encryption and decryption procedures for TCHo . In Section 3 we present our adaptive differential attack whose running time is discussed in detail in Section 4.

2 The TCHo Cipher

The encryption of a message using TCHo can be seen as transmitting a message over a noisy channel. Given the trapdoor, i.e. the secret key, allows to reduce the noise to a level where decoding of the encrypted message is possible.

The secret key of TCHo is a polynomial $K \in \mathbb{F}_2[x]$ of degree d. We denote its coefficients by k_0 up to k_d, i.e.

$$K = k_0 \oplus k_1 x \oplus k_2 x^2 \oplus \cdots \oplus k_d x^d.$$

For the key K it holds that $k_0 = k_d = 1$. Given the polynomial K we associate the following matrix M with ℓ columns and $\ell - d$ rows to it

$$M = \begin{pmatrix} k_0 & k_1 & \dots & k_d & 0 & 0 & \dots & 0 \\ 0 & k_0 & k_1 & \dots & k_d & 0 & \dots & 0 \\ & & \ddots & & & \ddots & & \\ 0 & 0 & \dots & 0 & k_0 & k_1 & \dots & k_d \end{pmatrix} \tag{1}$$

The weight of the secret polynomial, i.e. the number of non-zero coefficients is denoted by w_K. For TCHo this weight is small. The public key consists of a polynomial $P \in \mathbb{F}_2[x]$ whose degree is in a given interval $[d^P_{\min}, d^P_{\max}]$ and is chosen such that K is a multiple of P. The length k of the plaintext can be chosen arbitrarily, however following the proposed parameters in Table 1 we exemplarily choose the case where the plaintexts are 128 bit vectors. The length of the ciphertext is denoted by ℓ. Furthermore TCHo uses a random source with bias γ.

Table 1. Set of parameters proposed for TCHo (see [3])

	k	$d^P_{min} - d^P_{max}$	d	w_K	γ	ℓ
I65	128	5800 – 7000	25820	45	0.981	50000
II65	128	8500 – 12470	24730	67	0.987	68000
III	128	3010 – 4433	44677	25	$1 - \frac{3}{64}$	90000
IV	128	7150 – 8000	24500	51	0.98	56000
V	128	6000 – 8795	17600	81	$1 - \frac{3}{128}$	150000
VI	128	9000 – 13200	31500	65	$1 - \frac{1}{64}$	100000

For simplicity of the description we assume that $\ell - d$ is divisible by 128. Denote $N = \frac{\ell-d}{128}$. The attack has an identical complexity in the general case as can be seen from the experimental results in Section 4.3.

2.1 Encryption

TCHo encrypts a plaintext $m \in \mathbb{F}_2^{128}$ by repeating the message m contiguously and afterwards truncating it to ℓ bits. This results in a vector in $\mathbb{F}_2^{\ell}$. To this vector a random string $r \in \mathbb{F}_2^{\ell}$ is added. This random string is not balanced, but (highly) unbalanced in the sense that it contains far more zeros than ones. The bias is denoted by γ. In addition, the first ℓ bits, denoted by $p \in \mathbb{F}_2^{\ell}$, of the output of an (randomly initialized) LFSR with characteristic polynomial P is added. Thus the encryption of the message m is

$$c = R(m) \oplus r \oplus p$$

where $R(m) \in \mathbb{F}_2^{\ell}$ denotes the repeated and truncated version of m. The encryption process is shown in Figure 1.

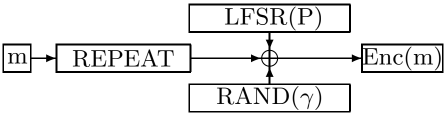


Fig. 1. Encryption with TCHo

2.2 Decryption

Given a ciphertext $c \in \mathbb{F}_2^{\ell}$ decryption works as follows.

1. The ciphertext c is multiplied by M, the matrix associated with the secret polynomial K given by (1). Let $t := Mc$ where $t \in \mathbb{F}_2^{\ell-d}$. In doing so, the contribution from the LFSR with characteristic polynomial P vanishes as a result of K being divisible by P. Thus t corresponds to an encoding of the original message m xored with a random bit string of bias approximately γ^{w_K} (see [3] for details). Now, as w_K is small, this bias is still large enough to recover the message in the next step with high probability.

2. A majority logic decoding is performed on t. More precisely for each $0 \leq j < 128$ the sum
$$s_j = \sum_{i=0}^{N-1} t_{128i+j}$$
is computed over the integers. Remember that $N = \frac{\ell - d}{128}$ and note that this is exactly the position where for simplicity of the description we require N to be an integer. When this sum is greater or equal to $N/2$ the result of the decoding is 1, otherwise it is 0. The result of this majority logic decoding is a vector $e \in \mathbb{F}_2^{128}$ where
$$e_j := \begin{cases} 1 & \text{if } s_j \geq \text{N/2} \\ 0 & \text{if } s_j < \text{N/2} \end{cases} \qquad j \in \{0, \ldots, 127\}$$
3. Finally the vector e is multiplied by an invertible 128×128 bit matrix T to recover the message $m := Te$. Note that this matrix T depends on K and is therefore unknown to the attacker.

2.3 Security Considerations

Given the public key P the problem to recover the secret key K is referred to as the *Low Weight Polynomial Multiple Problem*, i.e. given a polynomial P find a polynomial K that is divisible by P, has a bounded degree and low weight. This problem is supposed to be hard. In [7] several algorithms were presented to solve this problem. Additionally, an algorithm based on lattices was presented in [2]. None of these approaches is capable to break TCHo .

As mentioned above TCHo is clearly not IND-CCA2 secure. This is due to the fact that it is trivially malleable: Given a ciphertext c for a message m then $c \oplus R(m')$ is an encryption of $m \oplus m'$. Moreover, if given a ciphertext, changing only one bit is likely to be a valid ciphertext for the same message. In [3] the authors propose to use a generic hybrid construction to obtain a hybrid scheme that offers CCA2 security. However, this can only be applied efficiently in the case where the random source is actually a pseudo random source. Using a pseudo random source will increase the hardware complexity and is therefore a suboptimal solution. Furthermore, a secure hash function has to be implemented.

Below, we present a chosen ciphertext attack that recovers the secret key nearly in linear time. This attack shows that TCHo is not even CCA1 secure. Moreover, given a decryption oracle one can efficiently recover the secret key. From a practical perspective, such an attack is by far more important than an attack based on the malleability of the scheme.

3 The Attack

Our attack strategy is to decrypt pairs of ciphertexts with a carefully chosen difference. These differences are chosen such that the intermediate states in the decryption process, after multiplication with the secret matrix M, differ in one bit if and only if a certain key bit is set. With a high probability this difference will cause a difference in the output of the decryption oracle. Thus, after a few iterations of this approach we are able to decide with overwhelming probability if a certain key bit is set or not.

The reason why a difference after multiplication with the matrix M yields a difference after the majority logic decoding with good probability lies in the fact that randomly chosen vector is likely to be balanced. In this case the one bit difference between the two states will cause two different results after the majority logic decoding and therefore two different results after the last step in the decoding procedure, i.e. after multiplication with the invertible matrix T.

To illustrate our attack, we first demonstrate how to recover k_0 using the approach outlined above. Note that $k_0 = 1$ in any case, and thus there is no need to recover it, still this will clarify our attack strategy.

The following technical lemma will be used to estimate the success probability of our attack.

Lemma 1. *For $N \geq 1$ we have*

$$\frac{1}{\sqrt{2}}\frac{1}{\sqrt{\pi(N+1)}} < \frac{\binom{N}{\lceil N/2 \rceil}}{2^N} < 2\frac{1}{\sqrt{\pi N}}.$$

Proof. For even N the bounds obtained by Stirling's approximation state

$$\frac{1}{\sqrt{2}}\frac{1}{\sqrt{\pi N}} < \frac{\binom{N}{N/2}}{2^N} < 2\frac{1}{\sqrt{\pi N}}.$$

Looking at Pascals triangle, we get for odd N that $\binom{N}{(N+1)/2} = \frac{1}{2}\binom{N+1}{(N+1)/2}$, thus

$$\frac{1}{\sqrt{2}}\frac{1}{\sqrt{\pi(N+1)}} < \frac{\binom{N}{(N+1)/2}}{2^N} = \frac{\binom{N+1}{(N+1)/2}}{2^{N+1}} < 2\frac{1}{\sqrt{\pi(N+1)}}.$$

Combining both cases we have

$$\frac{1}{\sqrt{2}}\frac{1}{\sqrt{\pi(N+1)}} < \frac{\binom{N}{\lceil N/2 \rceil}}{2^N} < 2\frac{1}{\sqrt{\pi N}} \qquad \square$$

3.1 Recovering k_0

Recovering k_0 is very simple. First, one simply decrypts a randomly chosen bitstring. In a second step the first bit of the randomly chosen bitstring is flipped and is decrypted again. If the two decrypted messages differ then k_0 has to be equal to 1. This idea is explained in detail below.

Algorithm 1 *(Recover k_0)*

1. *Choose a random vector $c \in \mathbb{F}_2^\ell$ and let it be decrypted by the oracle. Let its decryption be m.*

2. *Compute the vector $c' = c \oplus \delta$ where $\delta = (1, 0, \dots, 0)$, i.e. flip the first bit of the ciphertext. Let c' be decrypted by the oracle and denote its decryption be m'.*
3. *If $m \neq m'$ we deduce that $k_0 = 1$.*
4. *Repeat these steps with a new random vector.*
5. *If after α repetitions no difference occurred, we deduce that $k_0 = 0$.*

The difference of the intermediate states, $t = Mc$ and $t' = Mc'$, after the first step in the decryption process (see Section 2.2) is

$$\Delta = M(c \oplus c') = M(1, 0, \dots, 0)^t = (k_0, 0, \dots, 0)^t.$$

Therefore t and t' differ if and only if k_0 is 1. Note that the attacker has no access to these values. However, if by coincidence this one bit difference causes a difference in the result of the majority logic decision during decryption, a difference will occur in the decrypted messages visible to the attacker. Let us denote by s_0 the value corresponding to the sum in the majority logic decoding step for t and s'_0 be the corresponding value for t', i.e.

$$s_0 = \sum_{i=0}^{N-1} t_{128i}, \quad s'_0 = \sum_{i=0}^{N-1} t'_{128i}$$

If s_0 is $\lceil N/2 - 1 \rceil$ (resp. $\lceil N/2 \rceil$) and the value of s'_0 is $\lceil N/2 \rceil$ (resp. $\lceil N/2 - 1 \rceil$) the values of e and e' after the majority logic decoding will differ in their first coordinates as

$$e_0 := \begin{cases} 1 & \text{if } s_0 \geq \text{N/2} \\ 0 & \text{if } s_0 < \text{N/2} \end{cases}$$

and

$$e'_0 := \begin{cases} 1 & \text{if } s'_0 \geq \text{N/2} \\ 0 & \text{if } s'_0 < \text{N/2} \end{cases}.$$

Therefore, in this case we can conclude that $k_0 = 1$ and moreover get

$$m \oplus m' = T(e \oplus e') = T(1, 0, \dots, 0)^t,$$

where T is the key dependent matrix used in the last step of the decryption procedure. The key point for the running time of the attack is that this happens with a non negligible probability. The fact that c was chosen randomly and M has maximal rank implies that t is a random vector in $\mathbb{F}_2^{\ell - d}$. Remember that s_0 equals $N/2$ if and only if exactly half of the bits t_{128j}, $j \in \{0, \dots, N-1\}$ are one and the other half is zero.

Thus, applying Lemma 1, we get

$$\mathcal{P}(s_0 = \lceil N/2 \rceil) = \frac{\binom{N}{\lceil N/2 \rceil}}{2^N} > \sqrt{\frac{1}{2\pi(N+1)}}.$$

Therefore, the probability to get a difference in m and m' is given by

$$\begin{aligned}\mathcal{P}(m \neq m' \mid k_0 = 1) &= \mathcal{P}(s_0 = \lceil N/2 \rceil \text{ and } s'_0 = \lceil N/2 - 1 \rceil) \\ &\quad + \mathcal{P}(s_0 = \lceil N/2 - 1 \rceil \text{ and } s'_0 = \lceil N/2 \rceil) \\ &= \frac{\binom{N}{\lceil N/2 \rceil}}{2^N} \mathcal{P}(t_0 = 0) + \frac{\binom{N}{\lceil N/2-1 \rceil}}{2^N} \mathcal{P}(t_0 = 1) \\ &> \sqrt{\frac{1}{8\pi(N+1)}}.\end{aligned}$$

Next we consider the error probability, i.e. the probability that, after running Algorithm 1 we deduce $k_0 = 0$ while it holds that $k_0 = 1$. This is given by the probability that, under the condition $k_0 = 1$, in none of the α tries a difference occurred. This probability can be upper bounded by

$$\mathcal{P}(\text{ error }) \leq \left(1 - \sqrt{\frac{1}{8\pi(N+1)}}\right)^{\alpha}$$

which is exponentially small in α. Note furthermore that following Algorithm 1 we will never erroneously deduce $k_0 = 1$ while it holds that $k_0 = 0$.

Example 1. For the parameter set (IV) in Table 1 we get for $\alpha = 100$ tries an error probability less than 0.006 and for $\alpha = 200$ an error probability less than 2^{-14}.

3.2 Recovering All Key Bits

The method to recover other key bits than k_0 generalizes the idea outlined above. Our goal is to construct two ciphertexts with a certain difference, such that with a high probability the majority decision flips one bit of the decrypted message if the keybit we are looking for is set. Below, we consider only the case where the attacker wants to recover k_n where $n < \ell - d$. This is the most challenging – and for all but the first proposed parameters the only– case that occurs. In the case where $n \geq \ell - d$ one can easily adopt the ideas described below to recover k_n with two decryptions only.

Choosing the difference. Let us assume that we already successfully recovered the bits k_0 up to k_{n-1} and want to recover k_n next. Denote the vector after multiplying the difference $\delta \in \mathbb{F}_2^\ell$ with the secret matrix M by Δ.

$$\Delta = \begin{pmatrix} k_0 & k_1 & \dots & k_d & 0 & 0 & \dots & 0 \\ 0 & k_0 & k_1 & \dots & k_d & 0 & \dots & 0 \\ & & \ddots & & & \ddots & & \\ 0 & 0 & \dots & 0 & k_0 & k_1 & \dots & k_d \end{pmatrix} \begin{pmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_\ell \end{pmatrix}$$

We wish to have the difference $\Delta = (1 \oplus k_n, 0, \dots, 0, 1, 0, \dots, 0)^t$ where the 1 is in the $(n+1)$-th position.

For the attack we have to distinguish two cases. First consider the case where n is not divisible by 128. In this case, for two ciphertexts c and c' with the difference δ the values s_0 and s'_0 will differ if k_n is not set. In the case where n is divisible by 128 the bit in the $(n+1)$-th position will contribute to the same sum s_0. In order to avoid a cancelation of these contributions special care has to be taken in the choice of the ciphertexts.

Given the knowledge about the key we already have we can compute a vector $\delta' \in \mathbb{F}_2^n$ such that

$$M'\delta' = (0, \ldots, 1)^t \tag{2}$$

where M' is the following $n \times n$ sub block of M:

$$M' = \begin{pmatrix} k_0 & k_1 & \ldots & k_{n-1} \\ 0 & k_0 & \ldots & k_{n-2} \\ 0 & & \ddots & \\ 0 & \ldots & 0 & k_0 \end{pmatrix}$$

Note that, as $k_0 = 1$ the matrix M' is bijective and thus the existence of δ' fulfilling (2) is guaranteed.

Now we can construct $\delta \in \mathbb{F}_2^\ell$ the following way: The first entry, δ_0, is computed as shown below, then the vector δ' is appended and finally δ is filled up with zeros, i.e.

$$\begin{array}{lll} \delta_0 = \sum_{i=0}^{n-2} \delta'_i k_{i+1} \oplus 1 & & \\ \delta_i = \quad \delta'_{i-1} & & \text{for } 1 \leq i \leq n \\ \delta_i = \quad 0 & & \text{for } n+1 \leq i < \ell. \end{array} \tag{3}$$

One verifies

$$\begin{aligned} \Delta = M\delta &= \left(k_0 (\sum_{i=0}^{n-2} \delta'_i k_{i+1} \oplus 1) \oplus \sum_{i=0}^{n-1} \delta'_i k_{i+1}, 0 \ldots, 0, 1, 0 \ldots, 0 \right)^t \\ &= \left(k_0 \oplus \delta'_{n-1} k_n, 0 \ldots, 0, 1, 0 \ldots, 0 \right)^t \\ &= (1 \oplus k_n, 0 \ldots, 0, 1, 0, \ldots, 0)^t, \end{aligned}$$

where the last equality follows as (2) implies $\delta'_{n-1} = 1$.

Choosing the ciphertext. Unlike in the case where we wanted to recover k_0 we will not use arbitrary random ciphertexts in this case, but restrict ourselves to certain types of vectors. As mentioned above, this will ensure that we can handle the case where we want to recover bits k_n where n is divisible by 128. Moreover, given the knowledge about the key we already have, carefully choosing the ciphertext will improve the probability of obtaining pairs such that a difference after multiplying with the secret matrix M will cause a difference at the output of the decryption oracle.

We choose the ciphertext c such that we obtain a vector $t = Mc$ with the properties that 1.) it starts with repeated blocks of 256 bits, where the first bit is one and the remaining 255 bits are zero, these repeated blocks are truncated to give an n bit string, 2.) has a zero in the $(n+1)$-th position, 3.) the remaining bits are randomly distributed. Since we know the key bits k_0 to k_{n-1}, we are able to compute the first part $\hat{c} \in \mathbb{F}_2^n$ of the ciphertext the same way we computed δ'. More precisely, using the matrix M' defined above, we are going to compute $\hat{c}$ such that

$$M'\hat{c} = b$$

where b consist of repeated blocks of 256 bits of the form $(1, 0^{(255)})$, i.e.

$$b = (1, 0^{(255)}, 1, 0^{(255)}, \dots) \in \mathbb{F}_2^n .$$

To get the zero entry at the $(n+1)$-th position of t, we will set $d+1$ consecutive bits of c equal to zero and finally the remaining bits of c are chosen uniformly at random. The ciphertext then has the structure

$$c = (\hat{c}, 0^{(d+1)}, r) \in \mathbb{F}_2^\ell \tag{4}$$

where $r \in \mathbb{F}_2^{\ell-n-(d+1)}$ is randomly chosen.

The one at the first position together with the zero at the $(n+1)$-th position of t will ensure that we can handle the case n divisible by 128. The repeated blocks of a one followed by 255 zeros at the beginning of t will increase the probability to get a difference in the decryptions provided that k_n is zero. This is explained in detail in Section 4.

Algorithm 2 *(Recover k_n)*

1. *Choose a random vector $r \in \mathbb{F}_2^{\ell-n-d-1}$ and compute c as described by (4). Let c be decrypted by the oracle. Let its decryption be m.*
2. *Compute the vector $c' = c \oplus \delta$ where δ was computed following (3) and let it be decrypted by the oracle. Let its decryption be m'.*
3. *If $m \oplus m' = T(1, 0, \dots, 0)^t$ we deduce that:*
   (a) *$k_n = 0$ in the case where $n \neq 0 \bmod 128$*
   (b) *$k_n = 1$ in the case where $n = 0 \bmod 128$*
4. *Repeat the steps with a new random vector.*
5. *If after α repetitions no difference equal to $T(1, 0, \dots, 0)^t$ occurred, we deduce that*
   (a) *$k_n = 1$ in the case where $n \neq 0 \bmod 128$*
   (b) *$k_n = 0$ in the case where $n = 0 \bmod 128$*

Note that, after performing the algorithm to recover k_0 the value $T(1, 0, \dots, 0)^t$ is known to the attacker.

4 Analysis of the Attack

We now analyze the success probability and running time of Algorithm 2. We distinguish two cases depending on $n \bmod 128$.

4.1 $n \neq 0 \bmod 128$

In this case the vectors $t = Mc$ and $t' = Mc'$ differ by

$$\Delta = t \oplus t = M(c \oplus c') = M\delta,$$

where

$$\Delta = (1 \oplus k_n, 0, \dots, 0, 1, 0, \dots, 0)^t$$

i.e. the vectors differ in their first coordinate if and only if $k_n = 0$ and in their $(n+1)$-th coordinate. Assume that $k_n = 0$. Due to (4) it holds that $t_0 = 1$ and $t'_0 = 0$ and thus the sums used for the majority logic decoding step are related by $s_0 = s'_0 + 1$. Analogously we have $s_n + 1 = s'_n$

Now, lets assume that the vector t is such that the sum used for the majority logic decoding step is $s_0 = \lceil N/2 \rceil$ (and thus $s'_0 = \lceil N/2 - 1 \rceil$). If this happens and additionally s_n and s'_n are either both less than $\lceil N/2 \rceil$ or both greater or equal to $\lceil N/2 \rceil$ then the corresponding vectors after the majority logic decoding differ in their first coordinate exactly. Thus Algorithm 2 will successfully deduce $k_n = 0$. Note that, due to the relation $s_n + 1 = s'_n$ the condition that either both values s_n and s'_n are smaller or both greater or equal to $\lceil N/2 \rceil$ is equivalent to $s'_n \neq \lceil N/2 \rceil$.

Remember that $t = Mc$ is of the form

$$t = (b, 0, r)$$

where $b \in \mathbb{F}_2^n$ is a vector consisting of repeated blocks of 256 bits, where the first bit is one and the remaining 255 bits are zero, and $r \in \mathbb{F}_2^{\ell-d-1-n}$ is a randomly chosen vector. Considering the bits t_{128j} that contribute to s_0, we see that the first $\lfloor n/128 \rfloor$ bits are balanced. Thus, $s_0 = \lceil N/2 \rceil$ if and only if half of the bits of r contributing to s_0 equal zero and the other half equals one. Therefore, the probability that $s_0 = \lceil N/2 \rceil$ equals

$$\mathcal{P}(s_0 = \lceil N/2 \rceil) = \frac{\binom{N'}{\lceil N'/2 \rceil}}{2^{N'}},$$

where $N' = \lceil \frac{\ell-d-1-n}{128} \rceil$. As n is not divisible by 128 the first $\lfloor n/128 \rfloor$ bits contributing to s'_n are all zero. Thus

$$\mathcal{P}(s'_n = \lceil N/2 \rceil) = \frac{\binom{N'}{\lceil N/2 \rceil}}{2^{N'}}.$$

It follows that the success probability

$$p := \mathcal{P}(m \oplus m' = T(1, 0, \dots, 0)^t \mid k_n = 0)$$

can be upper bounded by

$$\begin{aligned} p &= \mathcal{P}(s_0 = \lceil N/2 \rceil)\,\mathcal{P}(s'_n \neq \lceil N/2 \rceil) \\ &= \frac{\binom{N'}{\lceil N'/2 \rceil}}{2^{N'}} \left(1 - \frac{\binom{N'}{\lceil N/2 \rceil}}{2^{N'}}\right) \\ &\geq \frac{\binom{N'}{\lceil N'/2 \rceil}}{2^{N'}} \left(1 - \frac{\binom{N}{\lceil N/2 \rceil}}{2^{N}}\right) \\ &> \sqrt{\frac{1}{2\pi(N'+1)}} \left(1 - 2\sqrt{\frac{1}{\pi N}}\right). \end{aligned}$$

Next, let us consider the probability that, after running Algorithm 2 we deduce $k_n = 1$ while it holds that $k_n = 0$. This is given by the probability that, under the condition that $k_n = 0$, in none of the α tries a difference equal to $T(1, 0, \dots, 0)$ occurred. It can be upper bounded by

$$\mathcal{P}(\text{ error }) \leq (1 - p)^{\alpha}$$

which is exponentially small in α. Note that in the case where $k_n = 0$ the expected running time is $1/p$. As the weight of K is small, this is the running time for most of the cases.

4.2 $n = 0$ mod 128

Like before the vectors $t = Mc$ and $t' = Mc'$ differ by

$$\Delta = t \oplus t = M(c \oplus c') = M\delta,$$

i.e. the vectors differ in their first coordinate if and only if $k_n = 0$ and in their $(n+1)$-th coordinate. Due to (4) we have $t_0 = 1$ and $t'_0 = k_n$ and $t_n = 0$ and $t'_n = 1$. Now, as n is divisible by 128, the first and the $(n+1)$-th coordinate both contribute to the value of s_0 (resp. s'_0). Hence, we get $s'_0 = s_0 + k_n$.

Now if $k_n = 1$ and $s_0 = \lceil N/2 - 1 \rceil$ (and thus $s'_0 = \lceil N/2 \rceil$) the corresponding vectors e and e' after the majority logic decoding differ in their first coordinate exactly. Thus Algorithm 2 will successfully deduce $k_n = 1$.

Again the special choice of c ensures that the first $\lfloor n/128 \rfloor$ bits of t'_{128j} are balanced. Therefore the probability of s'_0 being $\lceil N/2 \rceil$ can be upper bounded by

$$\begin{aligned} \mathcal{P}(m \oplus m' = T(1, 0, \dots, 0)^t \mid k_n = 1) &= \mathcal{P}(s'_0 = \lceil N/2 \rceil) \\ &= \frac{\binom{N'}{\lceil N'/2 \rceil}}{2^{N'}} \\ &> \sqrt{\frac{1}{2\pi(N'+1)}} \end{aligned}$$

where again $N' = \lceil \frac{\ell - d - 1 - n}{128} \rceil$. Finally, let us consider the probability that, after running Algorithm 2 we deduce $k_n = 0$ while it holds that $k_n = 1$. This is givenby the probability that, under the condition that $k_n = 1$, in none of the α tries a difference equal to $T(1,0,\ldots,0)^t$ occurred. It can be upper bounded by

$$\mathcal{P}(\text{ error }) \leq \left(1 - \sqrt{\frac{1}{2\pi(N'+1)}}\right)^{\alpha}$$

which is exponentially small in α.

4.3 Experimental Results

We implemented the described attack against TCHo in C/C++ using Shoup's NTL library. We were able to derive the secret key for each proposed parameter set of [3] on a Core2 Duo 2.2 GHz laptop in less than 20 hours. The individual timings are given in Table 2.

Table 2. Time to recover the secret key

	k	d	w_K	ℓ	time in h
I65	128	25820	45	50000	2
II65	128	24730	67	68000	4.5
III	128	44677	25	90000	7
IV	128	24500	51	56000	3
V	128	17600	81	150000	20
VI	128	31500	65	100000	13

One implementation detail that is worth mentioning, is the computation of δ' (resp. $\hat{c}$). A straightforward approach might be to compute the inverse of the matrix M' of known bits. This has however an utterly bad performance, so that it is not even possible to consider matrices of dimension 5000, which is a rather small example compared to the size of secret polynomial. The best idea to compute δ' and $\hat{c}$ is to solve the corresponding system of equations. We started by using the method provided by NTL, but its performance was still unsatisfactory. Because of the extreme sparsity of the matrix M' and the additionally a priori given triangular form, it is obvious that solving such a system of equations over $\mathbb{F}_2$ should not require much computation resources. Therefore we implemented a simple backwards substitution using an array to store the known one-bits and a obtained very efficient method to compute the required values δ' and $\hat{c}$.

The value of α can be chosen rather large to get the probability of an error close to zero, since the expected number of encryptions to find a zero-bit does not depend on α and the number of one-bits is very small compared to the size of the key.

Also notice that it is possible to run an arbitrary number of instances in parallel to find the correct differences at the end of the decryption process.

There are several possibilities for further improvements of the actual attack. One could guess blocks of zeros, make use of the ability to detect missing one-keybits or reuse *good* ciphertext pairs. These improvements would allow to speed up the attack by some (small) factors.

Acknowledgement

We like to thank the authors of [3] for providing us with their sample implementation of TCHo as well as for helpful comments about the cipher.

References

1. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005)
2. El Aimani, L., von zur Gathen, J.: Finding low weight polynomial multiples using lattices. Cryptology ePrint Archive, Report 2007/423 (2007), `http://eprint.iacr.org/`
3. Aumasson, J.-P., Finiasz, M., Meier, W., Vaudenay, S.: TCHo : A hardware-oriented trapdoor cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 184–199. Springer, Heidelberg (2007)
4. Brent, R.P., Zimmermann, P.: Algorithms for finding almost irreducible and almost primitive trinomials. In: Primes and Misdemeanours: Lectures in Honour of the Sixtieth Birthday of Hugh Cowie Williams. The Fields Institute, Toronto, p. 212 (2003)
5. Canteaut, A., Chabaud, F.: A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. IEEE Transactions on Information Theory 44(1), 367–378 (1998)
6. Feldhofer, M., Rechberger, C.: A case against currently used hash functions in RFID protocols. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4277, pp. 372–381. Springer, Heidelberg (2006)
7. Finiasz, M., Vaudenay, S.: When stream cipher analysis meets public-key cryptography. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 266–284. Springer, Heidelberg (2007)
8. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
9. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: IEEE Symposium on Foundations of Computer Science, pp. 124–134 (1994)

An Algebraic Surface Cryptosystem

Koichiro Akiyama[1], Yasuhiro Goto[2], and Hideyuki Miyake[1]

[1] Computer & Network Systems Laboratory, Corporate Research & Development Center, Toshiba Corp., 1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, Kanagawa 212-8582, Japan
koichiro.akiyama@toshiba.co.jp, hideyuki.miyake@toshiba.co.jp
[2] Department of Mathematics, Hokkaido University of Education at Hakodate, 1-2 Hachiman-cho, Hakodate, Hokkaido 040-8567, Japan
ygoto@hak.hokkyodai.ac.jp

Abstract. We construct a public-key cryptosystem based on an NP-complete problem in algebraic geometry. It is a problem of finding sections on fibered algebraic surfaces; in other words, we use a solution to a system of multivariate equations of high degrees. Our cryptosystem is a revised version of the algebraic surface cryptosystem (ASC) we constructed earlier (cf. [AG04, AG06]). We revise its encryption algorithm to avoid known attacks. Further, we show that the key size of our cryptosystem is one of the shortest among those of post-quantum public-key cryptosystems known at present.

Keywords: Public-key Cryptosystem, Algebraic Surface, Section.

1 Introduction

In 1994, Shor showed that the factorization problem and the discrete logarithm problem can be solved efficiently by a quantum computer [Shr]. This implies that the RSA cryptosystem and Elliptic Curve cryptosystems will no longer be secure, once a quantum computer is built. We are thus in search for a public-key cryptosystem that does not rely on these problems and possibly can be implemented even on our present machines.

In this paper, we propose a new public-key cryptosystem whose security is based on an NP-complete problem in algebraic geometry. It is a problem of finding sections on algebraic surfaces fibered on an affine line. We shall call it a *section finding problem* (SFP) on algebraic surfaces. The SFP can be viewed as a problem of solving multivariate equation systems (of high degrees) over a finite field $\mathbb{F}_p$ with an arbitrary prime p. As this problem is known to be NP-complete, our cryptosystem is expected to have resistance against quantum computers. In what follows, we call our cryptosystem an *algebraic surface cryptosystem* (ASC).

The first version of the ASC was announced in [AG04]. It was then attacked by Uchiyama-Tokunaga [UT] and Voloch [Vol] in two different methods. The former attack uses a reduction-by-polynomial method that works in some special cases, while the latter employs a trace map of algebraic extensions of function fields that works in any case. (Eventually, Iwami [Iw08] found an unconditional reduction method generalizing the result of Uchiyama-Tokunaga.) The weakness

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 425–442, 2009.

of the original ASC lay in the use of a one-variable polynomial in the encryption algorithm. We have therefore changed it to a three-variable polynomial and revised the entire cryptosystem to avoid the attacks we mentioned above.

The new ASC was announced in [AG07] and soon, Voloch [Vol] came up with an idea of attacking this new system. Fortunately, however, it did not really break our system as we explain in Sect. 5.5 later. In the present paper, we reproduce the cryptosystem described in [AG07], and fill in various details and update the toy example.

One of the advantages of our new ASC is the small key size. To the best of our knowledge, it can offer one of the shortest keys of the known post-quantum public-key cryptosystems. For instance, a multivariate public-key cryptosystem is a candidate for a post-quantum cryptosystem. Its private key size is in the same order as ASC, but the public key size tends to be very large. The following table describes a rough comparison of the key sizes for several post-quantum cryptosystems, where n is a security parameter.

Table 1. Key size of several post-quantum public-key cryptosystems

Cryptosystem	Lattice-based	Multivariate	Knapsack	ASC
Public key	$O(n \log n)$	$O(n^3)$	$O(n^2)$	$O(n)$
Private key	$O(n \log n)$	$O(n)$	$O(n)$	$O(n)$

Both multivariate public-key cryptosystems and our ASC are associated with a system of multivariate algebraic equations $f_1 = \cdots = f_m = 0$ over a finite field $\mathbb{F}_p$. Typically, the public key of a multivariate public-key cryptosystem is constructed directly from polynomials $f_1, \cdots, f_m$. The public key of the ASC, on the other hand, is a single equation $X(x, y) = 0$ over a polynomial ring $\mathbb{F}_p[t]$ whose solution $(u_x(t), u_y(t))$ is a pair of polynomials over $\mathbb{F}_p$ related with $f_1, \cdots, f_m$ (precisely, the coefficients of $u_x(t)$ and $u_y(t)$ are solutions to some equation system $f_1 = \cdots = f_m = 0$). In this way, we can elude the direct use of $f_1, \cdots, f_m$ and save the public-key size drastically.

This paper is organized as follows. Section 2 collects some important facts about algebraic surfaces and recalls our original cryptosystem (ASC04). Section 3 describes the attacks on the ASC04 by Uchiyama-Tokunaga and Iwami, and also by Voloch. We discuss how to avoid their attacks. Section 4 presents our new algebraic surface cryptosystem which has resistance against various types of attacks. That resistance is discussed in Sect. 5. Lastly, the key size of the ASC is evaluated in Sect. 6. In Appendix, we give a toy example to illustrate our algorithm concretely.

2 Preliminaries

2.1 Algebraic Surfaces and the Section Finding Problem

Let $k := \mathbb{F}_p$ be a finite prime field of p elements. An algebraic surface over k is the set of solutions of algebraic equations over k that has two dimensional freedom over k. In order to construct our cryptosystem, we use an affine algebraic surface, X, in affine 3-space $\mathbb{A}_k^3$ defined by a single equation

$$f(x, y, t) = 0 \tag{1}$$

over k. It does not matter whether X is smooth or singular, but $f(x, y, t)$ should be irreducible.

There are many curves and points on X. For example, if we take another surface Y, then the intersection $X \cap Y$ is often a curve on X. These curves are easy to find, but finding *all* curves on X is a difficult problem; in fact, there is no effective algorithm to do so in general.

There is a special kind of curves on X which are generally very difficult to find explicitly. They are parameterized curves on X written in such a form as

$$(x, y, t) = (u_x(t), u_y(t), t) \ ,$$

where $u_x(t)$ and $u_y(t)$ are polynomials in t over k. If we define a map $\sigma : X \longrightarrow \mathbb{A}^1$ by $\sigma(x, y, t) = t$, then this parameterized curve induces an inverse map $\tau : \mathbb{A}^1 \longrightarrow X$ such that $\sigma \circ \tau = \mathrm{id}_{\mathbb{A}^1}$. The map σ is called a *fibration* of X on $\mathbb{A}^1$ and τ is called a *section* of σ. A section may be explained also as follows: rewriting $f(x, y, t)$ as a polynomial over $k[t]$, we can view X as a curve over the field $k(t)$ (or over the ring $k[t]$). Then a section is a $k(t)$-rational point on this curve. Finding such rational points is a Hilbert's 10th problem over a function field and is a hard mathematical problem. In our case, there is an exponential-time algorithm to solve this problem (cf. [Szp]), but no polynomial-time algorithm exists to find sections in general.

Definition 1. (Section Finding Problem) If $X(x, y, t) = 0$ is a surface over k, then the problem of finding a parameterized curve $(x, y, t) = (u_x(t), u_y(t), t)$ on X is called a *section finding problem* on X.

A general (but computationally inefficient) method of solving this problem, known at present, is as follows: express the defining equation for X as

$$X(x, y, t) = \sum_{(i,j,k) \in \Gamma_X} \eta_{i,j,k} x^i y^j t^k = 0 \ ,$$

where Γ_f denotes the set of indices (i, j, k) that appear in a polynomial $f(x, y, t)$. Choose r_x and r_y that satisfy $\deg u_x(t) < r_x$ and $\deg u_y(t) < r_y$ and write

$$\begin{aligned} u_x(t) &= \alpha_0 + \alpha_1 t + \cdots + \alpha_{r_x - 1} t^{r_x - 1} \ , \\ u_y(t) &= \beta_0 + \beta_1 t + \cdots + \beta_{r_y - 1} t^{r_y - 1} \ . \end{aligned}$$

The substitution of these into $X(x, y, t)$ gives

$$X(u_x(t), u_y(t), t) = \sum_{(i,j,k) \in \Gamma_X} \eta_{i,j,k} u_x(t)^i u_y(t)^j t^k =: \sum_i c_i t^i \ ,$$

where c_i are polynomials in α_i and β_j. If we write $r = \max\{i \deg u_x(t) + j \deg u_y(t) + k \mid (i,j,k) \in \mathit{\Gamma}_X\}$, then we find a system of equations

$$\begin{cases} c_0(\alpha_0, \cdots, \alpha_{r_x-1}, \beta_0, \cdots, \beta_{r_y-1}) = 0 \ , \\ \cdots \\ c_r(\alpha_0, \cdots, \alpha_{r_x-1}, \beta_0, \cdots, \beta_{r_y-1}) = 0 \ . \end{cases}$$

A solution to this system is a section of X. In this sense, our section finding problem can be reduced to solving a multivariate equation system of large degrees and such a problem is known to be NP-complete (cf. [GJ]).

2.2 Original Version (ASC04)

We briefly explain the first version of our cryptosystem (ASC04) that was announced in 2004. See [AG04] for the details.

Keys. Following are important system parameters:

1. Size of the ground field: p
2. Maximum degree of sections: d
3. Number of blocks in a plaintext: l (assume $d < l$)

[Public keys and secret keys]

1. The secret key is a pair of two sections

$$D_1 : (x,y,t) = (u_x(t), u_y(t), t), \qquad D_2 : (x,y,t) = (v_x(t), v_y(t), t)$$

with

$$d = \max\{\deg u_x(t), \deg u_y(t), \deg v_x(t), \deg v_y(t)\} \ . \tag{2}$$

2. The public key is a surface X that contains D_i as sections

$$X(x,y,t) = \sum_{(i,j) \in \mathit{\Lambda}_X} c_{ij}(t) x^i y^j = 0 \ ,$$

where $\mathit{\Lambda}_X := \{(i,j) \in \mathbb{N}^2 | c_{ij}(t) \neq 0\} \ni (0,0), (1,0)$.

Key Generation. First we choose polynomials $D_1 = (u_x(t), u_y(t), t)$ and $D_2 = (v_x(t), v_y(t), t)$, and then construct a surface $X(x,y,t)$ that contains D_1 and D_2 as sections. This can be done, for instance, by letting the polynomials satisfy $(u_x(t) - v_x(t)) | (u_y(t) - v_y(t))$.

Encryption Algorithm. Divide a plaintext m into l blocks as $m = m_0 || \cdots || m_{l-1}$ and embed m into a polynomial in t by

$$m(t) = m_{l-1} t^{l-1} + \cdots + m_1 t + m_0 \quad (0 \leq m_i < p, \ i = 0, \cdots, l-1) \ .$$

1. Choose an irreducible polynomial $f(t)$ of degree l.

2. Choose a random polynomial
$$r(x, y, t) = \textstyle\sum_{(i,j)\in\Lambda_r} r_{ij}(t)x^i y^j \tag{3}$$
and write
$$X(x, y, t)r(x, y, t) = \textstyle\sum_{(i,j)\in\Lambda_{Xr}} a_{ij}(t)x^i y^j \tag{4}$$
where $\Lambda_{Xr} := \{(i, j) \in \mathbb{N}^2 | a_{ij}(t) \neq 0\}$.
3. Randomly choose
$$s(x, y, t) = \sum_{(i,j)\in\Lambda_{Xr}} s_{ij}(t)x^i y^j \tag{5}$$
with $\deg s_{ij}(t) = \deg a_{ij}(t) - l$. This makes fs and Xr have the same form as polynomials in x and y over $k[t]$.
4. Set the cipher polynomial $F(x, y, t)$ to be
$$F(x, y, t) = m(t) + f(t)s(x, y, t) + X(x, y, t)r(x, y, t)\ . \tag{6}$$

Decryption Algorithm. First we substitute sections D_i into $F(x, y, t)$ and let
$$\begin{aligned} h_1(t) =&F(u_x(t), u_y(t), t) = m(t) + f(t)s(u_x(t), u_y(t), t)\ , \\ h_2(t) =&F(v_x(t), v_y(t), t) = m(t) + f(t)s(v_x(t), v_y(t), t)\ . \end{aligned}$$

1. Compute $h_1(t) - h_2(t)$ to find $f(t)\{s(u_x(t), u_y(t), t) - s(v_x(t), v_y(t), t)\}$.
2. Factor $h_1(t) - h_2(t)$ and obtain $f(t)$ as an irreducible polynomial of degree l.
3. Find $m(t)$ as the remainder in division of $h_1(t)$ by $f(t)$ and recover the plaintext m from $m(t)$.

3 Attacks on ASC04

There have been announced two attacks on the ASC04. We sketch the ideas of these attacks and analyze how to avoid them.

3.1 Reduction Attack by Uchiyama and Tokunaga

Uchiyama and Tokunaga announced an attack on the ASC04 in 2007 (cf. [UT]). Their algorithm is as follows.

1. Given a cipher text $F(x, y, t)$ as in (6), compute the remainder
$$R(x, y, t) = \sum_{(i,j)\in\Lambda_R} g_{ij}(t)x^i y^j \tag{7}$$
in division of $F(x, y, t)$ by a public key $X(x, y, t)$.
2. Let G be the set of all irreducible factors of $g_{ij}(t)$ of degree $\geq l$.

3. For each $f_i(t) \in G$, find the remainder $m_i(t)$ in division by $g_{00}(t)$. Then one of the $m_i(t)$'s coincides with the plaintext $m(t)$.

To make this algorithm work, it is necessary that G contains $f(t)$ and that g_{00} has the form $g_{00}(t) = m(t) + f(t)s(t)$ for some $s(t)$. In [UT], it is proven that this condition is satisfied if the leading term $LT(X)$ of $X(x, y, t)$ in a monomial order is of the form $LT(X) = cx^\alpha y^\beta$ with $c \in \mathbb{F}_p$.

The algorithm of Uchiyama and Tokunaga can be generalized if there exists a monomial order for x, y and t with which the remainder of $F(x, y, t)$ in division by $X(x, y, t)$ coincides with some part of

$$m(t) + f(t)s(x, y, t) = m(t) + f(t) \sum_{(i,j)\in\Lambda} s_{ij}(t)x^i y^j \ . \tag{8}$$

3.2 A Refinement by Iwami

The Uchiyama and Tokunaga attack had an assumption that the leading term $LT(X)$ of $X(x, y, t)$ in a monomial order is of the form $LT(X) = cx^\alpha y^\beta$ with $c \in \mathbb{F}_p$. In [Iw08], Iwami found a way to get rid of this assumption. The main idea is to consider $X(x, y, t)$ as a polynomial in two variables x and y over the field $\mathbb{F}_p(t)$ rather than as a polynomial in three variables over $\mathbb{F}_p$. Then by dividing through by the coefficient of the leading term, one can always have the situation $LT(X) = x^\alpha y^\beta$. Now apply the reduction algorithm to $X(x, y, t)$ over $\mathbb{F}_p(t)$ and clear the denominators of the coefficients. The same method of Uchiyama and Tokunaga on the numerators of the coefficients reveals the polynomial $f(t)$.

3.3 Conditions to Avoid the Reduction Attack

One way to avoid the reduction attack is to modify the ASC04 so that no monomial order will be effective to extract sufficient information of $m(t)$ and $f(t)$ when $F(x, y, t)$ is divided by $X(x, y, t)$.

Let $>$ be a monomial order on $k[x_1, \cdots, x_n]$ and write x^α for $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. For a non-zero polynomial $f = \sum_\alpha a_\alpha x^\alpha \in k[x_1, \cdots, x_n]$, let $multideg(f)$ denote the multidegree of f and $LT(f)$ the leading term of f. It is known that every polynomial $f \in k[x_1, \cdots, x_n]$ can be expressed as

$$f = aX + r$$

for some $a, r \in k[x_1, \cdots, x_n]$ satisfying $r = 0$ or r is a linear combination of monomials that are not divisible by $LT(X)$. Furthermore, if $aX \neq 0$, then $multideg(f) \geq multideg(aX)$. As r is not divisible by $LT(X)$ when $r \neq 0$, we can avoid the reduction attack if $LT(X)$ divide some monomials in $m(t)$ and $f(t)$ in every monomial order. Since there are an infinite number of monomial orders, almost all monomials in $X(x, y, t)$ can be a leading term. Therefore we are led to change $m(t)$ and $f(t)$ to polynomials $m(x, y, t)$ and $f(x, y, t)$ in 3 variables and pose the following condition:

(Condition). $m(x, y, t)$ and $f(x, y, t)$ contain some monomials that are divisible by all monomials in $X(x, y, t)$.

3.4 An Attack by Voloch

Another attack was suggested by Voloch [Vol]. His idea is to consider an extension of $\mathbb{F}_p(t)$ and use the trace map T. Let $F(x, y, t)$ be a ciphertext.

1. Substitute some polynomial $c(t)$ into y so that $X(x, c(t), t)$ becomes irreducible.
2. Let α be a solution to $X(x, c(t), t) = 0$ over $\mathbb{F}_p(t)$ and find $\beta \in \mathbb{F}_p(t)(\alpha)$ such that $T_{\mathbb{F}_p(t)(\alpha)/\mathbb{F}_p(t)}(\beta) = 0$.
3. Compute $T(\beta F(\alpha, c(t), t))$ and have
$$T(\beta F(\alpha, c(t), t)) = T(\beta m(t) + \beta f(t)s(\alpha, c(t), t)) = f(t)T(\beta s(\alpha, c(t), t)) \ .$$
4. Factor $T(\beta F(\alpha, c(t), t))$ and obtain $f(t)$.
5. Find $\beta_1 \in \mathbb{F}_p(t)(\alpha)$ such that $T_{F_p(t)(\alpha)/F_p(t)}(\beta_1) \in \mathbb{F}_p^{\times}$ and compute:
$$T(\beta_1 F(\alpha, c(t), t)) = m(t)T(\beta_1) + f(t)T(\beta_1 s(\alpha, c(t), t)) \ .$$
6. Divide $T(\beta_1 F(\alpha, c(t), t))$ by $f(t)$ to find $m(t)T(\beta_1)$ and then $m(t)$.

3.5 Ideas to Avoid Voloch's Attack

There may be two ways to avoid Voloch's attack: make the trace computation extremely time-consuming or change the form of $m(t)$ and $f(t)$. Both ideas can be realized simultaneously by letting $m(t)$ and $f(t)$ multi-variable.

For instance, replace $m(t)$ by $m(x, t)$ and $f(t)$ by $f(y, t)$. Choose $x = c(t)$. Let $y = \alpha$ be a solution to $X(c(t), y, t) = 0$. Compute $T(\beta_0 F(c(t), \alpha, t))$ with an element β_0 satisfying $T(\beta_0) = 0$. We can find $T(\beta_0 f(\alpha, t)s(c(t), \alpha, t))$. But, this does not yield $f(\alpha, t)$. Neither in the attempt by $y = c(t)$ can we obtain enough information for $f(y, t)$ or $m(x, t)$. Therefore the Voloch attack does not work in this case.

(On the other hand, if we replace $m(t)$ by $m(x, t)$ and keep $f(t)$ as is, then $f(t)$ can be obtained in the same way as in Sect. 3.4 and $m(x, t)$ can be found by taking various $y = c(t)$. Hence the case $(m(x, t),\ f(t))$ is insecure.)

Considering all cases, we conclude in particular the following:

(Safe case). The case where $m(t)$ and $f(t)$ are replaced by three-variable polynomials $m(x, y, t)$ and $f(x, y, t)$, respectively, is safe.

4 New Algorithm (Algebraic Surface Cryptosystem)

This section presents an improved algebraic surface public-key cryptosystem (ASC) which has resistance against the attacks described in Sect. 3. The discussions in Sect. 3.5 and 3.3 suggest that $m(t)$ and $f(t)$ should be 3-variable polynomials $m(x, y, t)$ and $f(x, y, t)$.

Although this idea is effective to avoid the attacks, a problem arises now in decryption steps 1 and 2 of ASC04 as $m(u_x(t), u_y(t), t) \neq m(v_x(t), v_y(t), t)$ and

$f(u_x(t), u_y(t), t) \neq f(v_x(t), v_y(t), t)$. Our solution to overcome this drawback is to employ an algebraic surface X with one section and use two cipher polynomials instead.

We assume that algebraic surfaces are defined over a prime field $\mathbb{F}_p$. (p is a prime small enough to calculate, such as primes within the word size.)

4.1 Keys

1. Secret key
   $D : (x, y, t) = (u_x(t), u_y(t), t)$: a section of X
2. Public key
   (a) $X(x, y, t) = 0$: a defining equation of a surface X with fibration.
   (b) $m(x, y, t) = \sum_{(i,j)\in\Lambda_m} m_{ij}(t)x^i y^j$: a plaintext polynomial where Λ_m and $\deg m_{ij}(t)$ are fixed.
   (c) $f(x, y, t) = \sum_{(i,j)\in\Lambda_f} f_{ij}(t)x^i y^j$: a divisor polynomial where Λ_f and $\deg f_{ij}(t)$ are fixed.

   Here Λ_A denotes the set of exponents of nonzero $x^i y^j$ terms in $A(x, y, t)$. We choose $m(x, y, t)$ and $f(x, y, t)$ so that they satisfy

   $$\Lambda_m \subset \Lambda_f \Lambda_X \tag{9}$$

   where $\Lambda_A\Lambda_B = \{(i_a + i_b, j_a + j_b)|(i_a, j_a) \in \Lambda_A, (i_b, j_b) \in \Lambda_B\}$.
   The decryption process requires that these keys satisfy the following condition:

   $$\begin{array}{l}\left\{\begin{array}{l}\deg_x X(x,y,t) < \deg_x m(x,y,t) < \deg_x f(x,y,t)\\ \deg_y X(x,y,t) < \deg_y m(x,y,t) < \deg_y f(x,y,t)\\ \deg_t X(x,y,t) < \deg_t m(x,y,t) < \deg_t f(x,y,t)\end{array}\right.\\ \text{and}\\ (\deg_x m(x,y,t), \deg_y m(x,y,t), \deg_t m(x,y,t)) \in \Gamma_m \ ,\\ (\deg_x f(x,y,t), \deg_y f(x,y,t), \deg_t f(x,y,t)) \in \Gamma_f \ ,\end{array} \tag{10}$$

   where $\Gamma_m = \{(i, j, k) \in \mathbb{N}^3 | c_{ijk} \neq 0\}$ denotes the set of exponents of nonzero $x^i y^j t^k$ terms in $m(x, y, t)$, so that $m(x, y, t) = \sum_{(i,j,k)\in\Gamma_m} c_{ijk}x^i y^j t^k$.

Condition (10) implies the following inequality:

$$\deg(m(u_x(t), u_y(t), t)) < \deg(f(u_x(t), u_y(t), t)) \ . \tag{11}$$

Also, we see that $m(x, y, t)$ and $f(x, y, t)$ have at least one term divisible by any terms of $X(x, y, t)$.

First we define a set of polynomials D (i.e. secret key) and then construct X containing D as a section. (Details are explained in Sect. 4.3.) For security reasons, we assume that the general fiber of X is not a rational curve. This can be realized, for instance, by letting $\deg_x X(x, y, t) > 2$ and $\deg_y X(x, y, t) > 2$.

4.2 Encryption/Decryption

Encryption. Let m be a plaintext, and divide m into small blocks as $m = m_{00}||\cdots||m_{ij}||\cdots||m_{IJ}$ where

$$\forall(i,j) \in \Lambda_m, \quad |m_{ij}| \leq (|p|-1)(\deg m_{ij}(t)+1) \ .$$

Further, write $\ell_{ij} := \deg m_{ij}(t)$ and divide m_{ij} into $\ell_{ij}+1$ blocks each of which is of $(|p|-1)$ bits:

$$m_{ij} = m_{ij0}||m_{ij1}||\cdots||m_{ij\ell_{ij}} \ .$$

1. Embed m into a plaintext polynomial as

$$m(x,y,t) = \sum_{(i,j)\in\Lambda_m} m_{ij}(t)x^i y^j$$

   where $m_{ij}(t)$ is given as

$$m_{ij}(t) = \sum_{k=0}^{\deg m_{ij}(t)} m_{ijk}t^k \ .$$

2. Choose a random divisor polynomial $f(x,y,t)$ in accordance with the condition of $f(x,y,t)$.
3. Choose random polynomials $r_0(x,y,t)$ and $r_1(x,y,t)$ that have the same form as $f(x,y,t)$; i.e. they have $\Lambda_r = \Lambda_f$ and $\deg r_{ij}(t) = \deg f_{ij}(t)$ for $(i,j) \in \Lambda_f$ as polynomials in x and y over $k[t]$.
4. Choose random polynomials $s_0(x,y,t)$ and $s_1(x,y,t)$ that have the same form as $X(x,y,t)$; i.e. they have $\Lambda_s = \Lambda_X$ and $\deg s_{ij}(t) = \deg c_{ij}(t)$ for $(i,j) \in \Lambda_X$ as polynomials in x and y over $k[t]$.
5. Construct the cipher polynomial $F(x,y,t)$ by

$$\begin{aligned} F_0(x,y,t) &= m(x,y,t) + f(x,y,t)s_0(x,y,t) + X(x,y,t)r_0(x,y,t) \ , \\ F_1(x,y,t) &= m(x,y,t) + f(x,y,t)s_1(x,y,t) + X(x,y,t)r_1(x,y,t) \ . \end{aligned} \tag{12}$$

Decryption. Note that the section D satisfies $X(u_x(t), u_y(t), t) = 0$ as they are on the surface X.

1. Substitute D into $F_i(x,y,t)$:

$$\begin{aligned} h_0(t) &= F_0(u_x(t), u_y(t), t) \\ &= m(u_x(t), u_y(t), t) + f(u_x(t), u_y(t), t)s_0(u_x(t), u_y(t), t) \ , \\ h_1(t) &= F_1(u_x(t), u_y(t), t) \\ &= m(u_x(t), u_y(t), t) + f(u_x(t), u_y(t), t)s_1(u_x(t), u_y(t), t) \ . \end{aligned}$$

2. Compute $h_0(t) - h_1(t)$:

$$h_0(t) - h_1(t) = f(u_x(t), u_y(t), t)\{s_0(u_x(t), u_y(t), t) - s_1(u_x(t), u_y(t), t)\} \ . \tag{13}$$

3. Factor $h_0(t) - h_1(t)$.

4. Find a factor of $h_0(t) - h_1(t)$ whose degree matches $\deg f(u_x(t), u_y(t), t)$. (This degree can be calculated from the initial setting of $f(x, y, t)$ and $D = (u_x(t), u_y(t), t)$.)
5. Compute $h_0(t) \equiv m(u_x(t), u_y(t), t) \pmod{f(u_x(t), u_y(t), t)}$ (cf. (11))
6. Extract the coefficient $m_{ij}(t)$ from $m(x, y, t)$ by solving linear equations.
   Let $m(x, y, t) = \sum_{(i,j,k)\in\Gamma_m} m_{ijk}x^iy^jt^k$, where m_{ijk}'s are variables. Construct linear equations by comparing the coefficients of t in

$$m(u_x(t), u_y(t), t) = \sum_{(i,j,k)\in\Gamma_m} m_{ijk}u_x(t)^iu_y(t)^jt^k \ .$$

   The left-hand side is given in Step 5.
7. Extract m from $m_{ij}(t)$ and authenticate the MAC of m. We can make certain of the plaintext m, if MAC is authenticated. Otherwise, return to Step 4.

In Step 4, we may not always extract $f(u_x(t), u_y(t), t)$ exactly, since the factor of degree equal to $\deg f(u_x(t), u_y(t), t)$ is not always unique. If this happens, then we repeat Steps 4 to 7 until MAC is authenticated.

We note that $\deg m(u_x(t), u_y(t), t)$ and $\deg f(u_x(t), u_y(t), t)$ are fixed. If the difference $\deg(f(u_x(t), u_y(t), t)) - \deg(m(u_x(t), u_y(t), t))$ in (11) is set large, then we have a good chance to find $f(u_x(t), u_y(t), t)$ immediately.

Remark 1. In the decryption process, some factorizations of polynomials in t can be rather time-consuming and Step 4 involves a knapsack problem. But, as we noted above, it is not an arbitrary knapsack problem, and so we can keep the entire algorithm practical. (The exact complexity of the decryption algorithm is under evaluation now and will be discussed elsewhere.)

4.3 Key Generation

Generation of Algebraic Surfaces. Let $X(x, y, t) = 0$ be a surface given by

$$X(x, y, t) = \sum_{(i,j)\in\Lambda_X} c_{ij}(t)x^iy^j \ .$$

1. Randomly choose a set of polynomials $(u_x(t), u_y(t))$ as a section.
2. Randomly choose polynomials $c_{ij}(t)$ with $(i, j) \neq (0, 0)$ and calculate $c_{00}(t)$ by

$$c_{00}(t) = - \sum_{(i,j)\in\Lambda_X\setminus\{(0,0)\}} c_{ij}(t)u_x(t)^iu_y(t)^j \ .$$

The Form of $\boldsymbol{m(x, y, t)}$ and $\boldsymbol{f(x, y, t)}$. We describe a method of determining $\deg f_{ij}(t)$ and $\deg m_{ij}(t)$. The form of $f(x, y, t)$ which satisfies (10) can be defined easily from the information of $X(x, y, t)$. Λ_m can be determined as a subset of $\Lambda_X\Lambda_f$ in (10). To find a plaintext efficiently, linear equations established in decryption step 6 should have a unique solution. In Step 6, we construct equations as follows

$$A \begin{pmatrix} m_{000} \\ m_{001} \\ m_{002} \\ \vdots \\ m_{ijk} \\ \vdots \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_K \end{pmatrix} , \tag{14}$$

where $c_0, \cdots, c_K$ are coefficients of

$$m(u_x(t), u_y(t), t) = \sum_{\tau=0}^{K} c_\tau t^\tau \ .$$

The equations (14) have a unique solution, if and only if rank $(A) = n$, where n denotes the number of variable m_{ijk}'s. Hence we must return to the determining process of the form of $f(x, y, t)$, if rank $(A) < n$.

5 Security Analysis

In this section, we will discuss about the resistance of our system against various types of attacks.

5.1 Reduction to a Multivariate Equation System

When we solve $F_0(x, y, t) - F_1(x, y, t) = f(x, y, t)s(x, y, t) + X(x, y, t)r(x, y, t)$, an obvious way is to let

$$\begin{aligned} f(x, y, t) &= \textstyle\sum_{(i,j,k)\in \Gamma_f} a_{ijk} x^i y^j t^i \ , \\ s(x, y, t) &= \textstyle\sum_{(i,j,k)\in \Gamma_X} b_{ijk} x^i y^j t^k \ , \\ r(x, y, t) &= \textstyle\sum_{(i,j,k)\in \Gamma_f} c_{ijk} x^i y^j t^k \ , \end{aligned}$$

and consider a multivariate equation system in a_{ijk}, b_{ijk} and c_{ijk}. If there exists a solution to this system, then we can obtain exact $f(x, y, t)$. Then we may have $m(x, y, t)$ by using ideal (f, X). But, as $\#\Gamma_f$ and $\#\Gamma_X$ increase, finding a solution to this system becomes considerably difficult, even if c_{ijk}'s are eliminated by substituting rational points of $X(x, y, t)$. For instance, if $\#\Gamma_f > 50$ and $\#\Gamma_X > 50$, then the system contains more than 100 variables. Hence it becomes computationally intractable when we choose a sufficiently large $\#\Gamma_f$ and $\#\Gamma_X$.

5.2 Reduction by the Defining Equation

One can try to divide $F_0(x, y, t) - F_1(x, y, t)$ by $X(x, y, t)$ to find a common divisor $f(x, y, t)$ in the possible remainders. But $f(x, y, t)$ does not appear in these remainders since $m(x, y, t)$ and $f(x, y, t)$ have at least one term divisible by any terms of $X(x, y, t)$; this is due to (10).

5.3 Reduction by Substituting Various Curves

Among the affine curves in $\mathbb{A}^3$, there are many rational curves parameterized in such a way as

$$(x, y, t) = (u_x(\omega), u_y(\omega), u_t(\omega)) \ .$$

If one can find such curves on X, then he can use them for the sections of X and decode the ciphertext in the same way as we decipher it using sections. We show, however, that finding such curves on X is as difficult as finding sections on X. We explain this according to $\deg u_t(\omega)$.

(i) Case $\deg u_t(\omega) \geq 2$

This is part of the divisor finding problem on X. As we assume the difficulty of it, such parameterized curves cannot be found easily.

(ii) Case $\deg u_t(\omega) = 1$

This is equivalent to finding sections on X, and hence it is difficult to find such curves on X.

(iii) Case $\deg u_t(\omega) = 0$

This means that t is set to be some constant value and we try to find a parameterized curve in ω. As we assume that the general fibers of X are non-rational (cf. Sect. 4.1), only singular fibers may contain rational curves. Hence we look for a singular fiber containing a rational curve.

One can find singular fibers by solving a system of equations $\partial X/\partial x = \partial X/\partial y = 0$ consisting of partial derivatives of $X(x, y, t) = 0$ with respect to x and y. But, as we raise the degree of X, this becomes considerably difficult. Also, no efficient algorithm is known for determining whether or not a singular fiber contains a rational curve. Even if it contains a rational curve, finding a parameterization by ω is a divisor finding problem and is known to be difficult. Therefore the attack by substituting rational curves does not seem to be effective.

5.4 Reduction to a Function Field $\mathbb{F}_p(t)$ by the Trace Map

As we explained in Sect. 3.5, Voloch's attack by the trace map (at least in the original form) does not work on ASC.

5.5 Voloch's New Attack

Previously, our ASC was announced in SCIS 2008 (cf. [AG07]) and soon after, Voloch communicated to us with a new attack that uses rational points on surfaces over finite fields; see [Vol]. The attacking procedure is described as follows:

1. Let $F(x, y, t) = F_1(x, y, t) - F_2(x, y, t)$; i.e.

   $$F(x,y,t) = f(x,y,t)(s_1(x,y,t) - s_2(x,y,t)) + X(x,y,t)(r_1(x,y,t) - r_2(x,y,t)).$$

2. Let $g(x, y, t) = f(x, y, t)(s_1(x, y, t) - s_2(x, y, t))$ and write

   $$g(x, y, t) = \sum_{(i,j) \in \Gamma_g} g_{ijk} x^i y^j t^k \ .$$

3. Find a large number of rational points (x_ℓ, y_ℓ, t_ℓ) on $X(x,y,t) = 0$ and substitute them into $F(x,y,t)$ to obtain a system of linear equations in $g_{ijk} \in \mathbb{F}_p$:
$$g(x_\ell, y_\ell, t_\ell) = F(x_\ell, y_\ell, t_\ell) \quad (\ell = 1, \cdots, n) \ . \tag{15}$$
4. Solve this system for g_{ijk} and factor $g(x,y,t)$ to find $f(x,y,t)$.
5. Finally, substitute rational points of $X(x,y,t) = 0$ into
$$F_1(x,y,t) = m(x,y,t) + f(x,y,t)s_1(x,y,t) + X(x,y,t)r_1(x,y,t)$$
to construct a system of linear equations in the coefficients of $m(x,y,t)$ and $s_1(x,y,t)$. A solution to this system gives $m(x,y,t)$.

Effectiveness of Voloch's Rational Point Attack. The above new attack requires many rational points on $X(x,y,t) = 0$, which can be obtained by raising the field of definition for $X(x,y,t) = 0$. However, we claim that no matter how many rational points we use, the polynomials $f(x,y,t)$ and $m(x,y,t)$ cannot be determined uniquely. In fact, if $g_0(x,y,t)$ is a solution to (15), then for any polynomial $r(x,y,t)$, $g_0(x,y,t) + X(x,y,t)r(x,y,t)$ also serves as a solution. Hence by raising the number of monomials in $r(x,y,t)$ (which is the same as in $f(x,y,t)$), we have too many candidates for $g(x,y,t)$ in the decryption process. More precisely, if γ denotes the number of monomials in $r(x,y,t)$, then there are p^γ candidates for $g(x,y,t)$. Therefore, for instance, by choosing γ that satisfies (16), we may avoid Voloch's rational point attack:
$$p^\gamma > 2^{100} \tag{16}$$
It is not difficult to create the situation with (16).

6 Key Size Estimation

Finally, we discuss the public and secret key sizes to keep the ASC sufficiently secure. ASC has four parameters d (maximal degree of the polynomials defining a section), $w = \deg_{xy} X(x,y,t)$, k (number of terms in $X(x,y,t)$ respect to x and y) and p (size of finite fields). Now we assume $p = 2$ to compare with the case of HFE.

In the case of $w \leq 4$, surfaces X are very likely to be elliptic or rational surfaces whose sections are known well. So w must be greater than or equal to 5 to avoid this case. Also, d must be greater than or equal to 50 to avoid the attack by Faugère et al. in [FJ03].

These observations suggest that the secret key size must be larger than 100 bits. A public key $X(x,y,t)$ contains coefficients $a_1(t), \cdots, a_k(t)$. The degrees of $c_{00}(t)$ can be set equal to dw by the key generation algorithm, if the coefficient of $x^{\deg_x X} y^{\deg_y X}$ is constant. So the public-key size can be set less than or equal to $(k-1)dw$ in size. The lower bound of k is 3, since the key generation algorithm requires a constant term. So the public-key size is presented in the linear from of d. Hence a lower bound for the public-key size is 500 bits, which is much smaller than HFE.

7 Conclusion

This paper has proposed a new type of public-key cryptosystem whose security is based on a section finding problem on algebraic surfaces. The section finding problem has no known efficient algorithm to solve other than finding roots of a multivariable equation system that is NP-complete in general. We show that our system requires only $O(n)$ bit key size that is much smaller than other post-quantum cryptosystems.

Acknowledgments

We thank Felipe Voloch for communicating us with his attacks on our cryptosystems at earlier stages. We also thank Shinji Miura, Shigenori Uchiyama and Hiroo Tokunaga for useful comments and discussions. We are grateful to Jintai Ding and Tatsuaki Okamoto for their constant encouragement. Many thanks are due to the referees for helpful comments and suggestions.

References

[AG04] Akiyama, K., Goto, Y.: An Algebraic Surface Public-key Cryptosystem. IEICE Tech. Report, vol. 104(421), pp. 13–20 (2004)

[AG06] Akiyama, K., Goto, Y.: A Public-key Cryptosystem using Algebraic Surfaces. In: Proc. of PQCrypto 2006, pp. 119–138 (2006)

[AG07] Akiyama, K., Goto, Y.: An improvement of the algebraic surface public-key cryptosystem. In: Proc. of SCIS 2008, CD-ROM 1F1-2 (2008)

[FJ03] Faugère, J.-C., Joux, A.: Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)

[Iw08] Iwami, M.: A Reduction Attack on Algebraic Surface Public-Key Cryptosystems. In: Kapur, D. (ed.) ASCM 2007. LNCS, vol. 5081, pp. 323–332. Springer, Heidelberg (2008)

[Kob98] Koblitz, N.: Algebraic Aspects of Cryptography. Springer, Heidelberg (1998)

[Shr] Shor, P.W.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, pp. 124–134 (1994)

[UT] Uchiyama, S., Tokunaga, H.: On the Security of the Algebraic Surface Public-key Cryptosystems (in Japanese). In: Proc. of SCIS 2007, CD-ROM 2C1-2 (2007)

[Vol] Voloch, F.: Breaking the Akiyama-Goto algebraic surface cryptosystem. Arithmetic, Geometry, Cryptography and Coding Theory, CIRM meeting (2007)

[GJ] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)

[Szp] Szpiro, L.: Seminaire sur les pinceaux de courbes de genre au moins deux. Asterisque 86(3), 44–78 (1981)

A Toy Example

We give an example of the algebraic surface cryptosystem described in this paper. This example is intended mainly to demonstrate our algorithm explicitly. The security of the system is not fully guaranteed; in practice, we should use more complicated polynomials.

Set $k = \mathbb{F}_{17}$ (i.e. $p = 17$).

A.1 Key Generation

Choose $u_x(t) = 14\,t^3 + 12\,t^2 + 5\,t + 1$ and $u_y(t) = 11\,t^3 + 3\,t^2 + 5\,t + 4$. Let D be a parameterized curve

$$D : (u_x(t), u_y(t), t) = (14\,t^3 + 12\,t^2 + 5\,t + 1, 11\,t^3 + 3\,t^2 + 5\,t + 4, t) \ . \qquad (17)$$

An algebraic surface X having D as a section is constructed as follows:

$$\begin{aligned} X(x,y,t) = {} & (t{+}10)x^3y^2{+}(16\,t^2{+}7\,t{+}4)xy^2{+}3\,t^{16}{+}8\,t^{15}{+}13\,t^{14}{+}8\,t^{13}{+}3\,t^{12}{+} \\ & 12\,t^{11}{+}4\,t^{10}{+}8\,t^9{+}7\,t^8{+}4\,t^7{+}13\,t^6{+}2\,t^5{+}5\,t^4{+}4\,t^3{+}14\,t^2{+}9\,t{+}14. \end{aligned}$$

We fix the form of a plain-text polynomial $m(x, y, t)$ and a divisor polynomial $f(x, y, t)$ as follows, where $\mathfrak{F}_m = \{(0, 0) = 17, (4, 4) = 17\}$ for instance means that the index set of $m(x, y, t)$ is $\Lambda_m = \{(0, 0), (4, 4)\}$ and $\deg m_{00}(t) = 17$ and $\deg m_{44}(t) = 17$:

$$\begin{aligned} \mathfrak{F}_X &= \{(0,0) = 16, (1,2) = 2, (3,2) = 1\}, \\ \mathfrak{F}_f &= \{(0,0) = 13, (1,2) = 11, (5,5) = 18\}, \\ \mathfrak{F}_m &= \{(0,0) = 17, (4,4) = 17\}. \end{aligned}$$

The polynomials $X(x, y, t)$, $f(x, y, t)$ and $m(x, y, t)$ thus constructed satisfy (10) and (16).

A.2 Encryption

For example, a plain text $m = 0xb3f25a22d683a10b362bc3e17a6b832794f5$ can be embedded into a polynomial $m(x, y, t)$ as

$$\begin{aligned} m(x,y,t) = {} & (5\,t^{17} + 15\,t^{16} + 4\,t^{15} + 9\,t^{14} + 7\,t^{13} + 2\,t^{12} + 3\,t^{11} + 8\,t^{10} + 11\,t^9 + \\ & 6\,t^8 + 10\,t^7 + 7\,t^6 + t^5 + 14\,t^4 + 3\,t^3 + 12\,t^2 + 11\,t + 2)x^4y^4 + 6\,t^{17} + \\ & 3\,t^{16} + 11\,t^{15} + t^{13} + 10\,t^{12} + 3\,t^{11} + 8\,t^{10} + 6\,t^9 + 13\,t^8 + 2\,t^7 + \\ & 2\,t^6 + 10\,t^5 + 5\,t^4 + 2\,t^3 + 15\,t^2 + 3\,t + 11. \end{aligned}$$

Then choose randomly an irreducible polynomial $f(x, y, t)$ as

$$\begin{aligned} f(x,y,t) = {} & (t^{18} + 8\,t^{17} + 8\,t^{16} + 6\,t^{15} + 3\,t^{14} + 11\,t^{13} + 12\,t^{12} + 9\,t^{11} + 14\,t^{10} + \\ & 8\,t^9 + 11\,t^8 + 10\,t^7 + 7\,t^6 + 8\,t^5 + 16\,t^4 + 10\,t^3 + 12\,t^2 + 7\,t + 16)x^5y^5 + \\ & (7\,t^{11} + 2\,t^{10} + 16\,t^9 + 16\,t^8 + 2\,t^7 + 4\,t^6 + 4\,t^5 + 9\,t^4 + 9\,t^3 + t^2 + \\ & 7\,t + 14)xy^2 + 8\,t^{13} + 12\,t^{12} + 15\,t^{11} + 5\,t^9 + 12\,t^8 + 13\,t^7 + 6\,t^6 + \\ & 6\,t^5 + 2\,t^4 + 13\,t^3 + 14\,t^2 + 14\,t + 11. \end{aligned}$$

Also, we may choose $s_i(x,y,t)$ and $r_i(x,y,t)$ as

$$\begin{aligned}
s_1(x,y,t) = \ & (4\,t+2)x^3y^2+(16\,t^2+9\,t+4)xy^2+8\,t^{16}+4\,t^{15}+11\,t^{14}+7\,t^{13}+t^{12}+\\
& 11\,t^{10}+8\,t^9+13\,t^8+12\,t^7+14\,t^6+16\,t^5+8\,t^4+13\,t^3+16\,t^2+14\,t+4,
\end{aligned}$$

$$\begin{aligned}
s_2(x,y,t) = \ & (7\,t+11)x^3y^2+(11\,t^2+3\,t+3)xy^2+t^{16}+3\,t^{15}+13\,t^{14}+t^{13}+\\
& 3\,t^{12}+16\,t^{11}+9\,t^{10}+4\,t^9+12\,t^7+t^6+7\,t^5+t^4+4\,t^3+2\,t+1,
\end{aligned}$$

$$\begin{aligned}
r_1(x,y,t) = \ & (10\,t^{18}+3\,t^{17}+7\,t^{16}+t^{15}+10\,t^{14}+10\,t^{13}+5\,t^{12}+7\,t^{11}+15\,t^{10}+\\
& 10\,t^9+8\,t^8+2\,t^7+16\,t^6+4\,t^4+t^3+3\,t^2+16\,t+2)x^5y^5+(t^{11}+\\
& 10\,t^{10}+14\,t^9+10\,t^8+2\,t^7+4\,t^6+13\,t^5+6\,t^4+10\,t^3+10\,t^2+\\
& 4\,t+15)xy^2+5\,t^{13}+16\,t^{12}+t^{11}+8\,t^{10}+8\,t^9+3\,t^8+3\,t^7+5\,t^6+\\
& 3\,t^5+3\,t^4+9\,t^3+7\,t^2+t+15,
\end{aligned}$$

$$\begin{aligned}
r_2(x,y,t) = \ & (12\,t^{18}+2\,t^{17}+7\,t^{16}+6\,t^{15}+8\,t^{14}+9\,t^{13}+16\,t^{12}+4\,t^{11}+8\,t^8+\\
& 8\,t^7+10\,t^6+13\,t^5+12\,t^4+11\,t^3+8\,t^2+4\,t+16)x^5y^5+(t^{11}+\\
& 8\,t^{10}+2\,t^9+t^8+4\,t^7+2\,t^6+8\,t^5+4\,t^4+13\,t^3+15\,t^2+2\,t+\\
& 8)xy^2+16\,t^{13}+6\,t^{12}+t^{11}+11\,t^{10}+16\,t^9+4\,t^8+2\,t^7+14\,t^6+\\
& 3\,t^5+7\,t^4+13\,t^3+13\,t^2+8\,t+16,
\end{aligned}$$

where $r_i(x,y,t)$ satisfies the condition of a divisor polynomial, $s_i(x,y,t)$ is in the same form as $X(x,y,t)$. Expanding $F_i(x,y,t)$, we obtain the following polynomial:

$$\begin{aligned}
F_0(x,y,t) = \ & (14\,t^{19}+t^{18}+9\,t^{16}+10\,t^{15}+7\,t^{14}+5\,t^{13}+15\,t^{12}+6\,t^{11}+16\,t^{10}+\\
& 15\,t^9+8\,t^8+16\,t^7+2\,t^6+16\,t^5+11\,t^4+13\,t^3+13\,t^2+2\,t+1)x^8y^7+\\
& (6\,t^{20}+3\,t^{18}+5\,t^{17}+6\,t^{16}+2\,t^{15}+7\,t^{13}+16\,t^{12}+5\,t^{11}+t^{10}+11\,t^9+\\
& 4\,t^8+11\,t^7+8\,t^6+6\,t^5+9\,t^4+14\,t^3+13\,t^2+12\,t+4)x^6y^7+(4\,t^{34}+\\
& 4\,t^{33}+10\,t^{32}+13\,t^{31}+2\,t^{30}+11\,t^{29}+3\,t^{28}+15\,t^{27}+7\,t^{25}+13\,t^{24}+\\
& 4\,t^{23}+6\,t^{21}+4\,t^{20}+t^{18}+15\,t^{17}+6\,t^{16}+16\,t^{15}+15\,t^{14}+7\,t^{13}+\\
& 14\,t^{11}+12\,t^{10}+8\,t^9+9\,t^8+6\,t^7+6\,t^6+10\,t^5+14\,t^4+2\,t^3+4\,t^2+t+\\
& 7)x^5y^5+(5\,t^{17}+15\,t^{16}+4\,t^{15}+9\,t^{14}+7\,t^{13}+14\,t^{12}+11\,t^{11}+3\,t^{10}+\\
& 2\,t^9+12\,t^8+3\,t^7+16\,t^6+11\,t^5+2\,t^4+16\,t^3+10\,t^2+10)x^4y^4+\\
& (3\,t^{14}+11\,t^{13}+7\,t^{12}+14\,t^{11}+6\,t^{10}+5\,t^9+7\,t^8+4\,t^6+2\,t^5+10\,t^4+\\
& 9\,t^3+2\,t^2+12\,t+2)x^3y^2+(9\,t^{13}+7\,t^{12}+5\,t^{11}+9\,t^{10}+7\,t^9+9\,t^8+\\
& 12\,t^7+8\,t^6+2\,t^5+13\,t^4+8\,t^3+4\,t^2+3\,t+14)x^2y^4+(8\,t^{27}+14\,t^{26}+\\
& 8\,t^{25}+16\,t^{24}+16\,t^{23}+13\,t^{22}+6\,t^{21}+13\,t^{20}+10\,t^{19}+4\,t^{18}+10\,t^{17}+\\
& 10\,t^{16}+13\,t^{15}+11\,t^{14}+14\,t^{13}+14\,t^{12}+15\,t^{11}+4\,t^{10}+11\,t^9+\\
& 13\,t^8+5\,t^7+4\,t^6+10\,t^5+13\,t^4+3\,t^3+2\,t^2+16\,t+13)xy^2+11\,t^{29}+\\
& 12\,t^{28}+10\,t^{27}+t^{26}+14\,t^{25}+16\,t^{24}+12\,t^{23}+14\,t^{22}+14\,t^{21}+11\,t^{20}+\\
& 7\,t^{19}+15\,t^{18}+6\,t^{17}+16\,t^{16}+15\,t^{15}+10\,t^{14}+4\,t^{13}+7\,t^{12}+16\,t^{11}+\\
& 11\,t^{10}+8\,t^9+2\,t^8+16\,t^7+t^6+12\,t^5+3\,t^4+13\,t^3+12\,t^2+5\,t+10,
\end{aligned}$$

$$\begin{aligned}
F_1(x,y,t) = \ & (2\,t^{19}+2\,t^{18}+t^{17}+2\,t^{16}+2\,t^{15}+12\,t^{14}+5\,t^{13}+2\,t^{12}+16\,t^{11}+\\
& 6\,t^{10}+3\,t^9+7\,t^8+11\,t^7+8\,t^6+2\,t^5+3\,t^4+6\,t^3+10\,t^2+7\,t+\\
& 13)x^8y^7+(16\,t^{20}+3\,t^{19}+12\,t^{17}+t^{16}+15\,t^{15}+15\,t^{14}+6\,t^{13}+\\
& 3\,t^{12}+3\,t^{11}+9\,t^{10}+11\,t^9+14\,t^8+7\,t^7+t^5+4\,t^4+t^3+5\,t^2+10\,t+\\
& 10)x^6y^7+(3\,t^{34}+11\,t^{33}+8\,t^{31}+11\,t^{30}+11\,t^{29}+4\,t^{28}+5\,t^{27}+t^{26}+
\end{aligned}$$

$$\begin{aligned}
&4\,t^{25}+3\,t^{24}+9\,t^{23}+5\,t^{22}+7\,t^{21}+16\,t^{20}+4\,t^{19}+10\,t^{18}+7\,t^{17}+\\
&9\,t^{16}+15\,t^{15}+13\,t^{14}+8\,t^{13}+9\,t^{12}+10\,t^{11}+10\,t^{10}+3\,t^{9}+14\,t^{7}+\\
&15\,t^{6}+4\,t^{5}+11\,t^{4}+2\,t^{3}+7\,t^{2}+t+2)x^5y^5+(5\,t^{17}+15\,t^{16}+4\,t^{15}+\\
&9\,t^{14}+7\,t^{13}+t^{12}+10\,t^{11}+3\,t^{10}+14\,t^{9}+6\,t^{8}+5\,t^{6}+5\,t^{5}+8\,t^{4}+\\
&16\,t^{3}+3\,t^{2}+10\,t+15)x^4y^4+(4\,t^{14}+15\,t^{13}+9\,t^{12}+16\,t^{11}+8\,t^{10}+\\
&14\,t^{9}+10\,t^{8}+15\,t^{7}+13\,t^{6}+15\,t^{5}+9\,t^{4}+10\,t^{3}+16\,t^{2}+4\,t+9)x^3y^2+\\
&(8\,t^{13}+8\,t^{12}+6\,t^{11}+3\,t^{10}+10\,t^{9}+9\,t^{8}+16\,t^{7}+13\,t^{6}+15\,t^{5}+\\
&4\,t^{4}+7\,t^{3}+6\,t^{2}+8\,t+6)x^2y^4+(10\,t^{27}+4\,t^{26}+9\,t^{25}+7\,t^{24}+3\,t^{23}+\\
&13\,t^{22}+16\,t^{21}+14\,t^{20}+t^{19}+t^{17}+6\,t^{16}+11\,t^{15}+9\,t^{14}+2\,t^{13}+\\
&16\,t^{12}+9\,t^{11}+16\,t^{10}+13\,t^{9}+2\,t^{7}+2\,t^{6}+14\,t^{5}+6\,t^{4}+15\,t^{3}+6\,t^{2}+\\
&14\,t+2)xy^2+5\,t^{29}+12\,t^{28}+6\,t^{27}+14\,t^{26}+5\,t^{25}+10\,t^{24}+12\,t^{23}+\\
&t^{22}+8\,t^{21}+2\,t^{20}+15\,t^{19}+3\,t^{18}+5\,t^{17}+14\,t^{15}+7\,t^{14}+5\,t^{13}+2\,t^{12}+\\
&9\,t^{11}+7\,t^{10}+11\,t^{9}+3\,t^{8}+10\,t^{7}+7\,t^{6}+14\,t^{4}+t^{3}+8\,t^{2}+6\,t+8.
\end{aligned}$$

A.3 Decryption

Substituting the section defined in (17) into $F_i(x,y,t)$ $(i=0,1)$, we obtain

$$\begin{aligned}
h_0(t) &= F_0(u_x(t),u_y(t),t)\\
&= 13\,t^{64}+8\,t^{63}+8\,t^{62}+13\,t^{61}+7\,t^{60}+16\,t^{58}+10\,t^{57}+13\,t^{56}+6\,t^{55}+3\,t^{54}+\\
&\quad 15\,t^{53}+3\,t^{52}+t^{51}+4\,t^{50}+2\,t^{49}+5\,t^{48}+12\,t^{47}+3\,t^{46}+8\,t^{44}+14\,t^{43}+\\
&\quad 9\,t^{42}+13\,t^{41}+14\,t^{40}+10\,t^{39}+8\,t^{38}+11\,t^{37}+12\,t^{36}+9\,t^{35}+7\,t^{33}+\\
&\quad 14\,t^{32}+12\,t^{31}+8\,t^{30}+4\,t^{28}+9\,t^{27}+15\,t^{26}+t^{25}+4\,t^{24}+8\,t^{23}+5\,t^{22}+\\
&\quad 14\,t^{21}+3\,t^{20}+7\,t^{19}+6\,t^{18}+7\,t^{17}+16\,t^{16}+9\,t^{15}+6\,t^{13}+3\,t^{12}+8\,t^{11}+\\
&\quad 11\,t^{10}+11\,t^{9}+14\,t^{8}+11\,t^{7}+15\,t^{6}+14\,t^{5}+2\,t^{4}+10\,t^{3}+10\,t^{2}+t+10,
\end{aligned}$$

$$\begin{aligned}
h_1(t) &= F_1(u_x(t),u_y(t),t)\\
&= 14\,t^{64}+6\,t^{63}+6\,t^{62}+8\,t^{61}+7\,t^{60}+t^{59}+4\,t^{58}+t^{57}+7\,t^{56}+11\,t^{55}+10\,t^{54}+\\
&\quad 2\,t^{53}+13\,t^{52}+16\,t^{51}+14\,t^{50}+15\,t^{49}+3\,t^{48}+3\,t^{46}+t^{45}+11\,t^{44}+10\,t^{43}+\\
&\quad 13\,t^{42}+8\,t^{41}+6\,t^{40}+9\,t^{39}+4\,t^{38}+13\,t^{37}+16\,t^{36}+13\,t^{35}+12\,t^{34}+\\
&\quad t^{33}+t^{32}+6\,t^{31}+15\,t^{30}+15\,t^{29}+16\,t^{28}+14\,t^{27}+2\,t^{26}+13\,t^{25}+16\,t^{24}+\\
&\quad 16\,t^{23}+3\,t^{22}+13\,t^{21}+4\,t^{20}+5\,t^{19}+15\,t^{18}+5\,t^{17}+4\,t^{16}+t^{15}+10\,t^{14}+\\
&\quad 15\,t^{13}+t^{11}+8\,t^{10}+6\,t^{9}+13\,t^{8}+15\,t^{6}+10\,t^{5}+4\,t^{4}+8\,t^{3}+11\,t^{2}+12\,t+2.
\end{aligned}$$

Factor $h_1(t)-h_2(t)$. We have

$$\begin{aligned}
h_1(t)-h_2(t) = {}& 16\,(t^3+3\,t^2+13\,t+3)(t^4+11\,t^3+15\,t^2+14\,t+13)(t^9+8\,t^8+\\
& 11\,t^7+3\,t^5+4\,t^4+6\,t^3+14\,t^2+12\,t+13)(t^{17}+2\,t^{16}+14\,t^{15}+\\
& 5\,t^{14}+5\,t^{13}+8\,t^{12}+9\,t^{11}+11\,t^{10}+3\,t^9+13\,t^8+10\,t^7+8\,t^6+\\
& 15\,t^5+7\,t^4+12\,t^3+10\,t^2+3\,t+2)(t^5+13\,t^4+4\,t^3+2\,t^2+\\
& 4\,t+13)(t^{16}+4\,t^{15}+11\,t^{14}+t^{13}+4\,t^{12}+13\,t^{11}+t^{10}+2\,t^9+\\
& t^8+2\,t^7+t^6+2\,t^4+15\,t^3+5\,t^2+11\,t+6)(t^6+4\,t^5+3\,t^4+\\
& 10\,t^3+14\,t^2+2\,t+5)(t^4+4\,t^3+5\,t^2+16\,t+10).
\end{aligned}$$

Then we find 4 candidates for $f(u_x(t),u_y(t),t)$ as it should have degree 48. Furthermore, by comparison of the degrees of $m(u_x(t),u_y(t),t)$ and $h_1(t) \pmod{f(t)}$, we can single out the correct $f(u_x(t),u_y(t),t)$ as

$$\begin{aligned} f(u_x(t),u_y(t),t) = {} & (t^3+3\,t^2+13\,t+3)(t^4+11\,t^3+15\,t^2+14\,t+13)(t^5+13\,t^4+ \\ & 4\,t^3+2\,t^2+4\,t+13)(t^6+4\,t^5+3\,t^4+10\,t^3+14\,t^2+2\,t+ \\ & 5)(t^9+8\,t^8+11\,t^7+3\,t^5+4\,t^4+6\,t^3+14\,t^2+12\,t+13)(t^{17}+ \\ & 2\,t^{16}+14\,t^{15}+5\,t^{14}+5\,t^{13}+8\,t^{12}+9\,t^{11}+11\,t^{10}+3\,t^9+ \\ & 13\,t^8+10\,t^7+8\,t^6+15\,t^5+7\,t^4+12\,t^3+10\,t^2+3\,t+2)(t^4+ \\ & 4\,t^3+5\,t^2+16\,t+10) \end{aligned}$$

and we obtain

$$\begin{aligned} m(u_x(t),u_y(t),t) = {} & 5\,t^{41}+10\,t^{40}+9\,t^{38}+9\,t^{36}+5\,t^{35}+12\,t^{34}+14\,t^{33}+9\,t^{31}+ \\ & 6\,t^{30}+t^{29}+t^{27}+7\,t^{26}+10\,t^{25}+3\,t^{24}+10\,t^{23}+13\,t^{22}+ \\ & 4\,t^{21}+10\,t^{20}+11\,t^{19}+6\,t^{18}+4\,t^{17}+5\,t^{16}+7\,t^{15}+14\,t^{14}+ \\ & t^{13}+7\,t^{12}+11\,t^{11}+5\,t^{10}+2\,t^9+8\,t^8+14\,t^7+13\,t^6+12\,t^5+ \\ & 16\,t^4+13\,t^3+9\,t^2+13\,t+13. \end{aligned}$$

Now recall that $m(u_x(t),u_y(t),t)$ has the form

$$\begin{aligned} m(u_x(t),u_y(t),t) = {} & (m_{4,4,17}t^{17} + m_{4,4,16}t^{16} + \cdots + m_{4,4,1}t + \\ & m_{4,4,0})u_x(t)^4u_y(t)^4 + m_{0,0,17}t^{17} + m_{0,0,16}t^{16} + \cdots + \\ & m_{0,0,1}t + m_{0,0,0}. \end{aligned} \tag{18}$$

By comparing two expressions of $m(u_x(t),u_y(t),t)$ above, we create a system of linear equations with variables $m_{i,j,t}$ as follows:

$$\begin{cases} m_{4,4,17} & = 5 \\ m_{4,4,0}+m_{0,0,0} & = 13 \\ 16\,m_{4,4,17}+m_{4,4,16} & = 10 \\ m_{0,0,1}+m_{4,4,1}+8\,m_{4,4,0} & = 13 \\ \quad\vdots & \\ m_{4,4,15} + 13\,m_{4,4,2} + 14\,m_{4,4,12} + 2\,m_{4,4,8} + 12\,m_{4,4,6} + & \\ 16\,m_{4,4,3}+10\,m_{4,4,13}+13\,m_{4,4,5}+5\,m_{4,4,0}+3\,m_{4,4,1}+m_{4,4,17}+ & \\ m_{4,4,14}+8\,m_{4,4,16}+m_{0,0,17} & = 4. \end{cases}$$

Solving this, we recover the plaintext polynomial

$$\begin{aligned} m(x,y,t) = {} & (5\,t^{17}+15\,t^{16}+4\,t^{15}+9\,t^{14}+7\,t^{13}+2\,t^{12}+3\,t^{11}+8\,t^{10}+11\,t^9+ \\ & 6\,t^8+10\,t^7+7\,t^6+t^5+14\,t^4+3\,t^3+12\,t^2+11\,t+2)x^4y^4+6\,t^{17}+ \\ & 3\,t^{16}+11\,t^{15}+t^{13}+10\,t^{12}+3\,t^{11}+8\,t^{10}+6\,t^9+13\,t^8+2\,t^7+ \\ & 2\,t^6+10\,t^5+5\,t^4+2\,t^3+15\,t^2+3\,t+11. \end{aligned}$$

Fast Multibase Methods and Other Several Optimizations for Elliptic Curve Scalar Multiplication

Patrick Longa, and Catherine Gebotys

Department of Electrical and Computer Engineering,
University of Waterloo, Canada
{plonga,cgebotys}@uwaterloo.ca

Abstract. Recently, the new Multibase Non-Adjacent Form (*mb*NAF) method was introduced and shown to speed up the execution of the scalar multiplication with an efficient use of multiple bases to represent the scalar. In this work, we first optimize the previous method using fractional windows, and then introduce further improvements to achieve additional cost reductions. Moreover, we present new improvements in the point operation formulae. Specifically, we reduce further the cost of composite operations such as quintupling and septupling of a point, which are relevant for the speed up of multibase methods in general. Remarkably, our tests show that, in the case of standard elliptic curves, the refined *mb*NAF method can be as efficient as Window-*w* NAF using an optimal fractional window size. Thus, this is the first published method that does not require precomputations to achieve comparable efficiency to the standard window-based NAF method using precomputations. On other highly efficient curves as Jacobi quartics and Edwards curves, our tests show that the refined *mb*NAF currently attains the highest performance for both scenarios using precomputations and those without precomputations.

Keywords: Elliptic curve cryptosystem, scalar multiplication, multibase non-adjacent form, double base number system, fractional window.

1 Introduction

Scalar multiplication, denoted by kP (where k is a scalar and P a point on the elliptic curve), is the most time consuming operation in Elliptic Curve Cryptosystems (ECC). Although several algorithms to compute kP using efficient representations of k have been proposed and extensively studied in past years, it is still a challenge to improve the performance of this operation for further deployment in embedded systems.

In that effort, a strategy that has gained lots of attention in recent years is the use of representations based on double- and multi-base chains. The use of the so-called Double Base Number System (DBNS) for cryptographic applications was first proposed by Dimitrov et al. in [7]. In the setting of ECC, double-base

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 443–462, 2009.

chains were first applied to the computation of scalar multiplication by Dimitrov et al. [8], and later extended to multibase chains by [17] and [9].

Although it was empirically shown that double-base chains reduce the cost of the scalar multiplication, the main drawbacks of the initial approaches using this strategy were their memory penalty and difficulty to analyze performance theoretically [8,11,1]. To solve these problems, an improved multibase representation was introduced by Longa in [17]. One of the key features of this representation is the use of the non-adjacency property (as found in NAF), which makes the conversion process (from binary to multibase) simple, efficient and with no memory impact. This new representation is called Multibase NAF (mbNAF). Its window-based version using an extended set of precomputations appears as a natural extension and is referred to as Window-w Multibase NAF (wmbNAF)[1].

Nevertheless, although Multibase NAF is simple and offers high performance, it is still possible to find shorter and more efficient multibase chains. In that direction, this work proposes new algorithms that are able to find highly efficient multibase chains and thus reduce scalar multiplication costs even further.

In addition, we also propose other several optimizations that aim at improving the efficiency of multibase methods (and standard methods in some cases). The contributions of this paper can be summarized as follows[2]:

- New reductions in the cost of composite (point) operations. We present improved formulas for quintupling and septupling in Jacobian coordinates.
- Improved (w)mbNAF-based algorithms (hereinafter referred to as Refined mbNAF methods) that "smartly" trade doublings for triplings/quintuplings and find shorter chains, reducing further the cost of the scalar multiplication.
- Window-based methods, namely wmbNAF and its refined version, are optimized by using fractional windows.
- The theoretical analysis demonstrating mathematically the performance of the Multibase NAF methods (and variants) is presented.
- Finally, we carry out a more comprehensive comparison taking into account most efficient curve shapes and point formulas found in the literature, and recent and most efficient methods to compute the evaluation and precomputation stages in the scalar multiplication.

Note that we focus here on methods that are efficient if the point P used to compute kP is not known in advance. In such a context, we analyze the performance of the proposed methods and compare to that of traditional binary methods such as NAF and wNAF, and another approaches using double-base chains (methods using a "Greedy" algorithm will be generically referred to as DB [8,11,1]). Our analysis includes the standard form of an elliptic curve over prime fields, and other very efficient curve forms such as Jacobi quartics [5] and Edwards curves [12]. It will turn out that the Refined mbNAF is currently the

[1] In some way, Multibase NAF can be seen as a generalization of the ternary/binary algorithm by Ciet et al. [6]. The original intention of the author [17], however, was to insert the concept of double-base chains proposed by [8] into the NAF algorithm.

[2] This paper presents formally and expands the results of the technical report [19].

most cost efficient method for both scenarios with and without precomputations and for all the studied curve forms.

Our work is organized as follows. In Section 2, we detail some background about ECC over prime fields, summarizing the state-of-the-art point formulae for Jacobian coord., Jacobi quartics and Edwards curves. Improvements to these operations are discussed in this section. In the following section, we briefly describe the original $(w)mb$NAF methods and discuss their theoretical performance. In Section 4, we optimize the performance of multibase algorithms by using fractional windows and present a detailed theoretical analysis. We then describe new improvements to Multibase NAF in Section 5 and propose the Refined mbNAF method, highlighting its advantages and high performance for computing scalar multiplication. In Section 6, the performance of various methods for scalar multiplication is evaluated through extensive tests. Finally, in Section 7 we present some conclusions summarizing the contributions of this work.

2 Elliptic Curve Cryptography

A brief introduction to ECC is presented in this section. The reader is referred to [14] for extended details. An elliptic curve E over a prime field $\mathbb{F}_p$ (denoted by $E(\mathbb{F}_p)$) can be defined by the simplified Weierstrass equation E: $y^2 = x^3 + ax + b$ (referred to as the standard EC form in the remainder), where $a, b \in \mathbb{F}_p$. The points on the curve E and the point at infinity, denoted by $\mathcal{O}$, form an additive group on top of which the cryptosystem works. Two basic operations exist to perform point computations: doubling $(2P)$ and addition $(P + Q)$ of points.

The representation of points on the curve E using (x, y), known as affine coordinates, introduces expensive field inversions (I) in the computation of point operations. Hence, most efficient implementations use representations of the form $(X : Y : Z)$, known as projective coordinates. For example, an efficient case of the latter is given by Jacobian coordinates, where each projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(X_i/Z_i^2, Y_i/Z_i^3)$. The reader is referred to [18] for complete details about most efficient formulae in this system.

Recently, other curve forms with faster group laws have appeared in the literature. We focus here on two of them: Jacobi quartics and Edwards curves, whose explicit formulas are highly efficient. We briefly described them in the following.

Jacobi quartic curve. It is defined by the projective curve $Y^2 = X^4 + 2aX^2Z^2 + Z^4$, where $a \in \mathbb{F}_p$ and $a^2 \neq 1$. A given projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(X_i/Z_i, Y_i/Z_i^2)$. The most efficient formulae in these curves have been developed by Hisil et al. [15,16] using an extended coordinate system of the form $(X_i : Y_i : Z_i : X_i^2 : Z_i^2)$.

Edwards curve. The projective curve in this setting is given by $(X^2 + Y^2)\, Z^2 = Z^4 + dX^2Y^2$. However, the most efficient explicit formulas in this case correspond to a new coordinate system known as inverted Edwards coord. [4], for which the curve equation takes the form $(X^2 + Y^2)\, Z^2 = X^2Y^2 + dZ^4$ and each projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(Z_i/X_i, Z_i/Y_i)$.

Note that the basic doubling and addition operations are sufficient to implement traditional methods relying on (signed) binary representations as NAF. However, new double- and multi-base methods require specialized operations such as tripling and/or quintupling of a point for their efficient realization.

In this work, we present optimized formulas for quintupling and septupling using Jacobian coord. (refer to Appendix A for complete details). Furthermore, certain computations such as those at the beginning of the evaluation stage can benefit from having specialized formulas with *mixed* coordinates that accept the input in affine and output the result in some projective system. Because of page constrains, formulas using mixed coordinates on standard and Edwards curves have not been included (the interested reader is referred to [19]). For mixed Jacobi quartic-affine coord., formulas can be easily derived from doubling, tripling, quintupling and septupling formulas due to [15,16] by setting $Z_1 = 1$.

In Table 1, we summarize the costs of the state-of-the-art point formulae, including the ones described above, for our three curves of interest: standard elliptic curves using Jacobian coordinates (*Jacobian*, parameter $a = -3$ in equation E), Jacobi quartics using the extended coordinate system (*JQuartic*) and Edwards curves using inverted Edwards coord. (*InvEdw*). For the remainder, doubling ($2P$), tripling ($3P$), quintupling ($5P$), septupling ($7P$), addition ($P+Q$) and doubling-addition ($2P+Q$) are denoted by D, T, Q, S, A and DA, respectively. Operations using *mixed* coordinates are denoted by mD, mT, mQ, mS, mA and mDA, corresponding to each of the aforementioned point operations. For addition, the case in which both inputs are in affine is denoted by mmA. Costs are expressed in terms of field multiplications (M) and squarings (S), disregarding field addition/subtractions (A) and multiplication/divisions by small constants for simplification purposes. We also assume that $1S = 0.8M$.

In some cases, it is possible to reduce the cost of certain operations if some values are precalculated in advance. That is the case of addition and doubling-addition (DA) with stored values in Jacobian coordinates (see Table 1). If, for instance, values Z_i^2 and Z_i^3 are precalculated for each precomputed point in windowed methods the costs of these point operations can be reduced by $1M + 1S$.

We use the efficient operations discussed in this section for our comparisons and cost analyses of scalar multiplication methods in Section 6.

Table 1. Cost of elliptic curve point operations

Curve	D/mD	T/mT	Q/mQ	S/mS	A	mA/mmA	DA/mDA
Jacobian	$3M+5S/$ $1M+5S$	$7M+7S/$ $5M+7S$ (1)	$10M+12S$ (1)/ $8M+12S$ (1)	$14M+15S$ (1)/ $12M+15S$ (1)	$10M+4S$ (2) $11M+5S$	$7M+4S/$ $4M+2S$	$13M+8S$ (2) $14M+9S/$ $11M+7S$
InvEdw	$3M+4S/$ $3M+3S$	$9M+4S/$ $7M+3S$ (1)	-	-	$9M+1S$	$8M+1S/$ $7M$	-
JQuartic	$2M+5S/$ $6S$ (1)	$8M+4S/$ $5M+5S$ (1)	$14M+4S/$ $11M+5S$ (1)	$16M+8S/$ $13M+9S$ (1)	$7M+3S$ (2) $7M+4S$	$6M+3S/$ $4M+3S$	-

(1) Introduced in this work (see Appendix A and [19]); (2) cost of operation with stored values.

3 Multibase Non-adjacent Form Methods

Following, we briefly describe the original Multibase NAF methods introduced in Longa [17]. Our main contribution in this section is to provide the theoretical analysis of the average density of these methods when using bases {2,3} and {2,3,5} that was deferred in [17].

3.1 Multibase NAF (*mb*NAF) and Window-*w* Multibase NAF (*wmb*NAF)

Determining and finding the "optimal" multibase chain in the setting of ECC seems to be a hard problem, mainly due to the fact that an "optimal" multibase chain is not necessarily the shortest, but the one that requires the "right" balance in the number of additions and all the other point operations. Although finding such an "optimal" multibase chain remains an open problem, Longa [17] proposed a representation that allows a better control of the appearance of point operations in the scalar expansion, and consequently, gets closer to the optimal. Such a generic multibase representation, known as *mb*NAF, has the form:

$$k = \sum_{i=1}^{m} s_i \prod_{j=1}^{J} a_j^{c_i(j)} \tag{1}$$

where $a_1 \neq \ldots \neq a_J$ are prime integers from a set of bases $\mathcal{A}=\{a_1,\ldots,a_J\}$ (a_1: main base),
m is the length of the expansion,
s_i are signed digits from a given set $D\backslash\{0\}$, i.e., $|s_i| \geq 1$ and $s_i \in D\backslash\{0\}$,
$c_i(j)$ are decreasing exponents, s.t. $c_1(j) \geq c_2(j) \geq \ldots \geq c_m(j) \geq 0$ for each j from 2 to J, and
$c_i(1)$ are decreasing exponents for the main base a_1(i.e., $j=1$), s.t. $c_i(1) \geq c_{i+1}(1) + 2 \geq 2$ for $1 \geq i \geq m-1$.

The last two conditions above guarantee that an expansion of the form (1) is efficiently executed by a scalar multiplication using Horner's method as follows:

$$kP=\prod_{j=1}^{J} a_j^{d_m(j)} \left(\prod_{j=1}^{J} a_j^{d_{m-1}(j)} \left(\cdots \left(\prod_{j=1}^{J} a_j^{d_1(j)} (s_1P) + s_2P \right) + \ldots + s_{m-1}P \right) + s_mP \right)$$

where $d_m(1) \geq 0$, and $d_i(1) \geq 2$ for $1 \geq i \geq m-1$. The latter is equivalent to the last condition in (1) and incorporates the non-adjacency property in the multibase representation. Basically, it fixes the minimal number of consecutive operations with the main base (i.e., a_1) between any two additions to 2. Note that an operation with the main base refers to a doubling if $a_1 = 2$ (this will be the case for most scenarios where doubling is the most efficient point operation).

If we relax the previous condition and allow larger window sizes (i.e., allowing 3, 4, or more, consecutive operations with the main base between any two additions) we can reduce further the average number of nonzero terms in the scalar representation at the expense of a larger digit set D and, consequently, a larger precomputed table. The previous technique is known as *wmb*NAF.

The *mb*NAF and *wmb*NAF representations require the following digit set [17]

$$D = \left\{0, \pm1, \pm2, \ldots, \pm \left\lfloor \frac{a_1^w - 1}{2} \right\rfloor \right\} \setminus \left\{\pm 1a_1, \pm 2a_1, \ldots, \pm \left\lfloor \frac{a_1^{w-1} - 1}{2} \right\rfloor a_1 \right\} \quad (2)$$

where $w \geq 2 \in \mathbb{Z}^+$ ($w = 2$ for *mb*NAF). Without considering $\{\mathcal{O}, P\}$, the digit set (2) involves precomputing $d_i P$, where $d_i \in D^+ \setminus \{0, 1\}$ (note that only positive values $d_i P$ need to be stored in the table as the inverse of points can be computed on the fly). Thus, the precomputed table consists of $(a_1^w - a_1^{w-1} - 2)/2$ points. Note that if $w = 2$ (*mb*NAF case), the requirement of precomputations is minimal. For instance, in the case $a_1 = 2$ we need to store *nil* points besides $\{\mathcal{O}, P\}$.

It is important to remark that, obviously, (1) does not involve unique representations. In [17], Longa provided algorithms (see Alg. 3.1) that efficiently find a multibase chain of the form (1) and, given a window width and set of bases, is unique for each integer. Note that Algorithm 3.1 integrates *mb*NAF and *wmb*NAF.

Algorithm 3.1. Computing the *mb*NAF (*wmb*NAF) of a positive integer

```
INPUT: scalar $k$, bases $\mathcal{A} = \{a_1, \ldots, a_J\}$, where $a_j \in \mathbb{Z}^+$ are primes for $1 \geq j \geq J$,
        window $w = 2$ for mbNAF, and window $w > 2$ for wmbNAF, where $w \in \mathbb{Z}^+$
OUTPUT: the $(a_1, a_2, \ldots, a_J)\mathrm{NAF}_w(k) = (\ldots, k_2^{(a_j)}, k_1^{(a_j)})$
    1. $i = 1$
    2. While $k > 0$ do
        2.1. If $k \bmod a_1 = 0$ or $k \bmod a_2 = 0$ or ... or $k \bmod a_J = 0$, then $k_i = 0$
        2.2. Else:
            2.2.1. $k_i = k \text{ mods } a_1^w$
            2.2.2. $k = k - k_i$
        2.3. If $k \bmod a_1 = 0$, then $k = k/a_1, k_i = k_i^{(a_1)}$
        2.4. Elseif $k \bmod a_2 = 0$, then $k = k/a_2, k_i = k_i^{(a_2)}$
                ⋮
        2.(J+2). Elseif $k \bmod a_J = 0$, then $k = k/a_J, k_i = k_i^{(a_J)}$
        2.(J+3). $i = i + 1$
    3. Return $(\ldots, k_2^{(a_j)}, k_1^{(a_j)})$
```

$k_i^{(a_j)}$ in Algorithm 3.1 represents the digits in the multibase NAF representation, where $k_i \in D$ (see (2)) and the superscript (a_j) represents the base $a_j \in \mathcal{A}$ associated to the digit in position i. The function *mods* represents the following

$$\begin{cases} \text{If } k \bmod a_1^w \geq a_1^w/2, \text{then } k_i = (k \bmod a_1^w) - a_1^w \\ \text{Else, } k_i = k \bmod a_1^w \end{cases}$$

Let us illustrate the method using Alg. 3.1 with the following example.

Example 1. The *mb*NAF representation of 9750 according to Algorithm 3.1 is $(2,3)\mathrm{NAF}_2(9750) = 1^{(2)}\,0^{(2)}\,0^{(2)}\,0^{(2)}\,1^{(2)}\,0^{(3)}\,0^{(2)}\,\text{-}1^{(2)}\,0^{(2)}\,0^{(2)}\,1^{(2)}\,0^{(3)}\,0^{(2)}$, which would allow to compute $9750P$ as $2 \times 3\ \ (2^3\,(2^2 \times 3(2^4P + P) - P) + P)$. The latter involves 1mD+9D+2T+3mA. For instance, using Table 1 (JQuartic, $1S = 0.8M$), $9750P$ would cost $107.2M$. Compare this to the cost using NAF, i.e., 1mD+12D+5mA = $119.6M$.

Zero and Nonzero Density of Multibase NAF Methods. One of the attractive properties of Multibase NAF methods is that the average number of operations can be precisely determined by using Markov chains. The following theorems are presented on this regard (please, refer to Appendix B for the proofs).

Theorem 1. *The average densities of additions, doublings and triplings for the (w)mbNAF using bases $\mathcal{A} = \{2,3\}$ are approximately*

$$\delta_x = \frac{2^w}{3(2^{w-2}-s)+2^w(w+1)}\,, \delta_{0^2} = \frac{2^w(w+1)}{3(2^{w-2}-s)+2^w(w+1)} \ \ \textit{and}\ \ \delta_{0^3} = \frac{3(2^{w-2}-s)}{3(2^{w-2}-s)+2^w(w+1)},$$

respectively, where $s = \lfloor (2^{w-2}+1)/3 \rfloor$ and $w \geq 2 \in \mathbb{Z}^+$ ($w = 2$ for mbNAF).

Theorem 2. *The average densities of additions, doublings, triplings and quintuplings for the (w)mbNAF using bases $\mathcal{A} = \{2,3,5\}$ are approximately*

$$\delta_x = \frac{2^{w+3}}{17{\cdot}2^{w-1}-5r-24s-5t+2^{w+3}(w+1)}\,, \delta_{0^2} = \frac{2^{w+3}(w+1)}{17{\cdot}2^{w-1}-5r-24s-5t+2^{w+3}(w+1)}\,,$$
$$\delta_{0^3} = \frac{24(2^{w-2}-s)}{17{\cdot}2^{w-1}-5r-24s-5t+2^{w+3}(w+1)} \ \ \textit{and}\ \ \delta_{0^5} = \frac{5(2^{w-1}-r-t)}{17{\cdot}2^{w-1}-5r-24s-5t+2^{w+3}(w+1)},$$

respect., where $r = \lfloor (2^{w-2}+2)/5 \rfloor$, $s = \lfloor (2^{w-2}+1)/3 \rfloor$ and $t = \lfloor (2^{w-2}+7)/15 \rfloor$.

Let us determine the average number of operations when using the Multibase NAF method. First, it is known that the expected number of doublings, triplings and additions is given by $\#D = \delta_{0^2} \cdot digits$, $\#T = \delta_{0^3} \cdot digits$ and $\#A = \delta_x \cdot digits$, where *digits* represents the total number of digits in the expansion (note that a nonzero digit involves one doubling and one addition). Also, we can assume that $2^{\#D} \cdot 3^{\#T} \approx 2^{n-1}$, where n represents the average bitlength of the scalar k. Thus, $\#D \cdot \log 2 + T \cdot \log 3 \approx (n-1)\log 2$, and replacing #D and #T, we can estimate *digits* with the following

$$digits \approx \frac{(n-1)\log 2}{\delta_{0^2} \cdot \log 2 + \delta_{0^3} \cdot \log 3} \tag{3}$$

which allow us to determine #D, #T and #A using the expressions above. A similar procedure easily follows for the case of bases {2,3,5}.

For instance, in the case of *mb*NAF, bases $\mathcal{A} = \{2,3\}$, $w = 2$ and $n = 160$ bits, the average densities for doublings, triplings and additions derived from Theorem 1 are 4/5, 1/5 and 4/15. Using (3), we determine that $digits = 142.35$. Then, the average cost of a scalar multiplication using Table 1 (JQuartic,$1S = 0.8M$) is approx. 113.88D+28.47T+37.96mA = $1321M$. Similarly, if we use bases

$\mathcal{A} = \{2,3,5\}$, the average cost can be estimated as approximately 97.06D+24.27T+10.11Q+32.35mA = $1299.82M$. Compare the previous costs to that offered by NAF: 159D+53mA = $1399.2M$ (in this case, $\delta_{NAF} = 1/3$). Hence, theoretically, it is determined that (2,3)NAF and (2,3,5)NAF surpasses NAF (case with no precomputations) by about 5.6% and 7.1%, respectively.

Despite these results, it is still possible to find more efficient multibase chains at the expense of some increment in the complexity of the basic Multibase NAF. The improved multibase algorithms will be discussed in Section 5. Following, we optimize the basic multibase methods using a recoding based on fractional windows.

4 The Fractional Window-*w* Multibase Non-adjacent Form (Frac-*wmb*NAF)

In this section, we apply the concept of "fractional" windows [24] to the multibase NAF method to allow a flexible number of points in the precomputed table. The new representation is called Fractional *wmb*NAF (denoted by Frac-*wmb*NAF).

For the remainder, we will assume that the main base a_1 is 2 as this value is expected to achieve the lowest costs with most efficient ECC curve forms. First, let us establish our ideal table with unrestricted number of points d_iP, where $d_i \in D^+ \backslash \{0,1\} = \{3, 5, \ldots, m\}$, and $m \geq 3 \in \mathbb{Z}^+$ is an odd integer. If we define m in terms of the standard windows w, it would be expressed as

$$m = 2^{w-2} + s, \tag{4}$$

where $2^{w-2} < m < 2^{w-1}$ and $s \geq 1 \in \mathbb{Z}^+$ is odd.

We can now define the rules of our recoding scheme for bases $\mathcal{A} = \{a_1, a_2, \ldots, a_J\}$ in the following:

1. If (k mod $2 = 0$ or k mod $a_2 = 0$ ...or k mod $a_J = 0$), then $k_i = 0$
2. Elseif $0 < r \leq m$, then $k_i = r$
3. Elseif $m < r < (3m - 4s)$, then $k_i = r - 2^{w-1}$
4. Elseif $(3m - 4s) \leq r < 2^w$, then $k_i = r - 2^w$
5. $k = k - k_i$

where $r = k$ mod 2^w. Basically, the proposed recoding first detects if k is divisible by one of the bases. Else, it establishes a window w and checks if k can be approximated to the closest extreme of the window using any of the digits d_i available. It can be verified that the latter will be accomplished if steps 2 or 4 are satisfied. Otherwise, the established window is too large and, hence, it is "reduced" to the immediately preceding window size to which k can be approximated (condition in step 3).

An algorithm to convert any integer to Frac-*wmb*NAF representation can be easily derived by replacing steps 1-5 above by steps 2.1 and 2.2 in Algorithm 3.1. In this case, we will denote the Frac-*wmb*NAF of an integer k by $(2, a_2, \ldots, a_J)\text{NAF}_{w,t}(k) = (\ldots, k_2^{(a_j)}, k_1^{(a_j)})$, where t represents the number of precomputed points, i.e., $m = 2t + 1$.

Let us illustrate the new recoding with the following example.

Example 2. If $k = 9750$ and $m = 5$, then $d_i \in D^{+}\backslash\{0,1\} = \{3,5\}$, and $w = 4$ and $s = 1$ by means of (4). Then, the Frac-*wmb*NAF of 2950 is given by (2,3)NAF$_{4,2}$(9750) $= 1^{(2)}0^{(2)}0^{(2)}0^{(2)}$-$3^{(2)}0^{(2)}0^{(2)}0^{(2)}$-$5^{(2)}0^{(2)}0^{(2)}1^{(2)}0^{(3)}0^{(2)}$, and the conversion process can be visualized as $\frac{9750}{2} \rightarrow \frac{4875}{3} \rightarrow 1625 - 1 \rightarrow \frac{1624}{8} \rightarrow 203 + 5 \rightarrow \frac{208}{16} \rightarrow 13 + 3 \rightarrow \frac{16}{16} \rightarrow 1$.

Observe that, when 1625 is obtained, it requires an addition with 7 to reach 1632 (which is the closest number $\equiv$ (0 mod 2^4), as required by a standard window $w = 4$). However, 7 is not part of our precomputed table, so the window size is reduced accordingly to $w = 3$ and the value 1625 is approximated to the closest value in the new window (i.e., 1624) using an addition with -1.

We now present the following theorem regarding the average density of this method for the case $\mathcal{A} = \{2,3\}$.

Theorem 3. *The average densities of nonzero terms, doublings and triplings of the Frac-wmbNAF using bases $\mathcal{A} = \{2,3\}$, window size w and t available points (represented by (2,3)NAF$_{w,t}$) are approximately*

$$\frac{2^w}{8(t+1)-3(u+v)+2^{w-2}(4w-1)}\,,\ \frac{8(t+1)+2^w(w-1)}{8(t+1)-3(u+v)+2^{w-2}(4w-1)}\,,\ \frac{3(2^{w-2}-(u+v))}{8(t+1)-3(u+v)+2^{w-2}(4w-1)}$$

respectively, where $u = \lfloor (t+2)/3 \rfloor$ and $v = \lfloor (2^{w-2}-t)/3 \rfloor$.

The reader is referred to Appendix C for a proof. With Theorem 3, it is possible to theoretically estimate the expected number of doublings, triplings and additions using this method. For instance, following the procedure detailed in Section 3.1, we can determine the cost of a scalar multiplication (without including precomputation) for $n = 160$ bits using $t = 2$ points ($w = 4$) as 132.7D+16.6T+29.5mA = 1229.9M (JQuartic). Compare to the cost achieved by Frac-wNAF, namely 159D+35.3mA = 1250.5M ($\delta_{Frac-wNAF} = 1/4.5$ when using $m = 5$; see [25]). Further cost reductions are observed for the case of $\mathcal{A} = \{2,3,5\}$.

5 The Refined Multibase Non-adjacent Form (Refined *mb*NAF)

In this section we present a new methodology to derive algorithms able to find more efficient multibase chains. Similarly to the original Multibase NAF methods, we base our approach on the key observation that point operations such as doublings and triplings have different costs and that any multibase algorithm with application to scalar multiplication should not only try to reduce the length of the expansion but also (and more importantly) find the right balance between the number of all the point operations involved.

Following the previous criteria, we modify the original Multibase NAF using the next conditional statements (again, we restrict our analysis to the most efficient case $a_1 = 2$. Also, q, r are odd integers and $\{2, a_2, \ldots, a_J\} \nmid q,\, r$):

1. CONDITION1: before every nonzero term, approximate the current partial value k to the closest value in the established window with $k - k_i = 2^w \cdot a_2^{w_2} \cdot$

$\ldots \cdot a_J^{w_J} \cdot q$, where $k_i \in D\backslash\{0\} = \{\pm 1, \pm 3, \pm 5, \ldots, \pm m\}$, as usually performed, if and only if there does not exist some value $k - d_i = 2^{w'_1} \cdot a_2^{w'_2} \cdot \ \ldots \ \cdot a_J^{w'_J} \cdot r$ in the established window, with $w_j, w'_j \geq 0$ for each base from the set $\mathcal{A} = \{a_1, a_2, \ldots, a_J\}$ and $d_i \in D\backslash\{0\} \neq k_i$, such that the zero digit sequence to follow is ”greater” than that guaranteed by the window w (i.e., for practical purposes, $2^{w'_1} \cdot \ldots \cdot a_J^{w'_J} > 2^{w} \cdot \ldots \cdot a_J^{w_J} + e$), in which case (*CONDITION1 = true*) the approximation $k - d_i$ is applied instead of $k - k_i$.

2. CONDITION2: before each zero term different than $0^{(2)}$, we test if there is a nonzero digit $d_i \in D\backslash\{0\}$ which would allow an approximation $k - d_i = 2^{w'} \cdot a_2^{w'_2} \cdot \ \ldots \ \cdot a_J^{w'_J} \cdot r$ such that $2^{w'} \cdot \ldots \cdot a_J^{w'_J} > a_2^{w_2} \cdot \ldots \cdot a_J^{w_J} + e'$, where $k = a_2^{w_2} \cdot \ \ldots \ \cdot a_J^{w_J} \cdot q$ is a partial scalar value and $w_j, w'_j \geq 0$ for each base from the set $\mathcal{A}$. If the latter happens (*CONDITION2 = true*), the approximation $k - d_i$ replaces the testing and dividing by extra bases.

Algorithm 5.1. Computing the Refined *mb*NAF of a positive integer

```
INPUT: scalar $k$, bases $\{2,3\}$ or $\{2,3,5\}$, digit set $D\backslash\{0\}=\{\pm 1, \pm 3, \ldots, \pm(m=2t+1)\}$
       $w \geq 2 \in \mathbb{Z}^+; m = 2^{w-2} + s$ and $2^{w-2} < m < 2^{w-1}$, where $m \geq 3$ and $s \geq 1$
       are odd integers ($m = 1, s = 0$ for case without precomputations)
OUTPUT: the Refined $(2,3)\mathrm{NAF}_{w,t}(k)$ or $(2,3,5)\mathrm{NAF}_{w,t}(k) = (\ldots, k_2^{(a_j)}, k_1^{(a_j)})$
1. $i = 1$, $exception = 0$
2. While $k > 0$ do
    2.1. If $exception = 0$ and ($k \bmod 2 = 0$ or ... or $k \bmod a_J = 0$), then $k_i = 0$
    2.2. Else:
        2.2.1. $r = k \bmod 2^w$
        2.2.2. If $0 < r \leq m$, then $k_i = r$
        2.2.3. Elseif $m < r < (3m - 4s)$, then $k_i = r - 2^{w-1}$
        2.2.4. Elseif $(3m - 4s) \leq r < 2^w$, then $k_i = r - 2^w$
        2.2.5. If $CONDITION1 = true$, then $k_i = d_i$
        2.2.6. $k = k - k_i$, $exception = 0$
    2.3. If $k \bmod 2 = 0$, then $k = k/2, k_i = k_i^{(2)}$
    2.4. Elseif $k_i = 0$
        2.4.1. If $CONDITION2 = true$, then $exception = 1$
        2.4.2. Elseif $k \bmod 3 = 0$, then $k = k/3, k_i = k_i^{(3)}$
          ⋮
        2.4.J. Elseif $k \bmod a_J = 0$, then $k = k/a_J, k_i = k_i^{(a_J)}$
    2.5. $i = i + 1$
3. Return $(\ldots, k_2^{(a_j)}, k_1^{(a_j)})$
```

CONDITION1 aims at fulfilling our first criteria, namely, reducing the length of the expansion. In this case, parameter e guarantees that the new approximation will yield a much shorter chain such that is justifiable to use more expensive point operations instead of the usual sequence of doublings after each nonzero term. Similarly, *CONDITION2* is responsible for fixing a good balance between the different point operations. In particular, it will "smartly" insert more doublings, as these are the most efficient operations in most common ECC settings. In this case, e' is a security parameter that guarantees that the algorithm in fact trades expensive point operations by a large enough sequence of doublings such that is justifiable to introduce an extra nonzero term. Both parameters, e and e', vary according to the relative cost among point operations and even with the value of the scalar. Despite this complexity, we have been able to determine parameter values that are efficient for most cases by performing extensive tests with random numbers.

We have inserted the modifications above to the Frac-wmbNAF algorithm (see Section 4), since this representation generalizes the $(w)mb$NAF methods, and derived Algorithm 5.1 for bases {2,3} and {2,3,5}.

Notice the addition of *CONDITION1* and *2* in steps 2.2.5. and 2.4.1. As can be seen from the descriptions above, these techniques are quite general and give a high degree of freedom to adjust the algorithm to different settings with different constrains in the complexity level. In this work, we have focused on selecting parameters that achieve high performance without increasing excessively the complexity of the Multibase NAF algorithms.

The recommended conditional statements for Alg. 5.1 are detailed in Tables 2 and 3 for bases {2,3} and {2,3,5}, respectively. We remark that these are only recommended parameters, and that *CONDITION1* and *2* can be modified to suit the complexity constrains of a specific implementation, leading to different performance levels.

It is clear from Tables 2 and 3 that the original conditions of the Multibase NAF regarding non-adjacency (see (1)) have been relaxed. In particular, according to *CONDITION1*, it can be the case that fewer consecutive doublings are inserted for a particular window size.

Let us illustrate the proposed method with the following example.

Example 3. Using Algorithm 5.1 and Table 2, we find that the Refined mbNAF chain for computing $9750P$ using bases {2,3}, $w = 4, m = 5$, is $9750 = 5 \times 2^3 \times 3^5 + 5 \times 2 \times 3$, which has been derived using the sequence $\frac{9750}{2} \rightarrow \frac{4875}{3} \rightarrow 1625 - 5 \rightarrow \frac{1620}{2} \rightarrow \frac{810}{2} \rightarrow \frac{405}{3} \rightarrow \frac{135}{3} \rightarrow \frac{45}{3} \rightarrow \frac{15}{3} \rightarrow 5$.

Notice that the partial value 1625 is conveniently approximated to 1620, by means of *CONDITION1*, instead of 1624 (see Example 2), allowing the efficient insertion of several triplings to reduce the length of the expansion. If we compare the performance of the refined method when computing $9750P$ against the basic Multibase NAF approach using the same fractional window size (see Example 2), we can observe that the cost reduces significantly from 12D+1T+3mA = $108.4M$ to only 3D+5T+1mA = $82.4M$ (JQuartic, $1S = 0.8M$).

As can be observed, the gain in performance with this method is obtained by increasing the complexity in the conversion step. This may or may not be a limiting factor depending on the characteristics of a particular implementation and the chosen platform. In cases where the conversion to multibase becomes non-negligible, the method would still remain practical for settings where the same scalar k is reused several times.

Table 2. Recommended parameters for *CONDITION1* and *2*, bases $\mathcal{A} = \{2, 3\}, d_i \in D \backslash \{0, k_i\}$

Window w	*CONDITION1*	*CONDITION2*
2 $m = 1$	If ($\mathcal{D} \bmod 9 = 0$ and $\mathcal{K} \bmod 8 \neq 0$) or ($\mathcal{D} \bmod 27 = 0$ and $\mathcal{K} \bmod 16 \neq 0$) or ($\mathcal{D} \bmod 81 = 0$ and $\mathcal{K} \bmod 32 \neq 0$)	If ($\mathcal{D} \bmod 16 = 0$ and $k \bmod 9 \neq 0$) or ($\mathcal{D} \bmod 32 = 0$ and $k \bmod 27 \neq 0$) or ($\mathcal{D} \bmod 64 = 0$ and $k \bmod 81 \neq 0$)
3 $m = 3$	If ($\mathcal{D} \bmod 27 = 0$ and $\mathcal{K} \bmod 16 \neq 0$) or ($\mathcal{D} \bmod 81 = 0$ and $\mathcal{K} \bmod 32 \neq 0$)	If ($\mathcal{D} \bmod 2^{w+1} = 0$ and $k \bmod 9 \neq 0$) or ($\mathcal{D} \bmod 2^{w+2} = 0$ and $k \bmod 27 \neq 0$) or ($\mathcal{D} \bmod 2^{w+3} = 0$ and $k \bmod 81 \neq 0$)
4 $m = 5, 7$	If ($\mathcal{D} \bmod 216 = 0$ and $\mathcal{K} \bmod 32 \neq 0$) or ($\mathcal{D} \bmod 324 = 0$ and $\mathcal{K} \bmod 64 \neq 0$)	
5 $m = 9, \ldots, 15$	If ($\mathcal{D} \bmod 144 = 0$ and $\mathcal{K} \bmod 64 \neq 0$) or ($\mathcal{D} \bmod 432 = 0$ and $\mathcal{K} \bmod 128 \neq 0$)	
6 $m = 17, \ldots, 31$	If ($\mathcal{D} \bmod 288 = 0$ and $\mathcal{K} \bmod 128 \neq 0$) or ($\mathcal{D} \bmod 864 = 0$ and $\mathcal{K} \bmod 256 \neq 0$)	

(*) $\mathcal{D} = k - d_i$; $\mathcal{K} = k - k_i$

Table 3. Recommended parameters for *CONDITION1* and *2*, bases $\mathcal{A} = \{2, 3, 5\}, d_i \in D \backslash \{0, k_i\}$

Window w	*CONDITION1*	*CONDITION2*
2 $m = 1$	If ($\mathcal{K} \bmod 5 \neq 0$ and (($\mathcal{D} \bmod 9 = 0$ and $\mathcal{K} \bmod 8 \neq 0$) or ($\mathcal{D} \bmod 27 = 0$ and $\mathcal{K} \bmod 16 \neq 0$) or ($\mathcal{D} \bmod 81 = 0$ and $\mathcal{K} \bmod 32 \neq 0$))) or ($\mathcal{D} \bmod 15 = 0$ and $\mathcal{K} \bmod 8 \neq 0$)	If $k \bmod 3 \neq 0$ If ($\mathcal{D} \bmod 2^{w+2} = 0$ and $k \bmod 25 \neq 0$) or ($\mathcal{D} \bmod 2^{w+3} = 0$ and $k \bmod 125 \neq 0$) or ($\mathcal{D} \bmod 2^{w+4} = 0$ and $k \bmod 625 \neq 0$) else If $\mathcal{K} \bmod 25 \neq 0$ and (or ($\mathcal{D} \bmod 2^{w+2} = 0$ and $k \bmod 9 \neq 0$) or ($\mathcal{D} \bmod 2^{w+3} = 0$ and $k \bmod 27 \neq 0$) or ($\mathcal{D} \bmod 2^{w+4} = 0$ and $k \bmod 81 \neq 0$))
3 $m = 3$	If ($\mathcal{K} \bmod 5 \neq 0$ and (($\mathcal{D} \bmod 27 = 0$ and $\mathcal{K} \bmod 16 \neq 0$) or ($\mathcal{D} \bmod 81 = 0$ and $\mathcal{K} \bmod 32 \neq 0$))) or ($\mathcal{D} \bmod 45 = 0$ and $\mathcal{K} \bmod 32 \neq 0$)	
4 $m = 5, 7$	If ($\mathcal{K} \bmod 5 \neq 0$ and (($\mathcal{D} \bmod 108 = 0$ and $\mathcal{K} \bmod 32 \neq 0$) or ($\mathcal{D} \bmod 324 = 0$ and $\mathcal{K} \bmod 64 \neq 0$)))	
5 $m = 9, \ldots, 15$	If ($\mathcal{K} \bmod 5 \neq 0$ and (($\mathcal{D} \bmod 144 = 0$ and $\mathcal{K} \bmod 64 \neq 0$) or ($\mathcal{D} \bmod 432 = 0$ and $\mathcal{K} \bmod 128 \neq 0$)))	
6 $m = 17, \ldots, 31$	If ($\mathcal{K} \bmod 5 \neq 0$ and (($\mathcal{D} \bmod 288 = 0$ and $\mathcal{K} \bmod 128 \neq 0$) or ($\mathcal{D} \bmod 864 = 0$ and $\mathcal{K} \bmod 256 \neq 0$)))	

(*) $\mathcal{D} = k - d_i$; $\mathcal{K} = k - k_i$

To evaluate the performance of this refined methodology for scalar multiplication, we implemented the method and carried out several tests. The results are summarized in the following section.

6 Performance Comparison

We have carried out several tests to demonstrate the high performance of the Multibase NAF methods discussed in this work when applied on standard, Jacobi quartic and Edwards curves. We implemented the traditional Frac-wNAF and the (Refined) Frac-wmbNAF (Algorithm 5.1) and ran the algorithms with different window sizes for 1000 160-bit scalars chosen randomly. In the case of Multibase NAF, we evaluated the methods when using the following sets of bases $\mathcal{A} : \{2, 3\}$ and $\{2, 3, 5\}$.

To estimate costs for each method, we first counted the required number of point operations per scalar, averaged the results and then calculated the cost using Table 1. Also, for windows $w > 2$ we included in the overall cost the cost of calculating the precomputed points. For computing these points, we consider two cases: points are left in projective coordinates (referred to as *case 1*), and points are converted to affine using *one* inversion (referred to as *case 2*). As expected, case 2 is advantageous using Jacobian coordinates, where mDA is significantly more efficient than the general DA version (see Table 1). Ultimately, the particular I/M ratio of an implementation will decide which case is more effective on a standard curve. In the case of JQuartic and InvEdw, we only consider case 1 as this scheme should be largely preferred because of the minimal difference of costs between general and mixed additions. Specifically, for Jacobian coordinates, we use the efficient scheme proposed in [22], and for JQuartic and InvEdw we apply the recently proposed scheme by the authors [20].

The costs using the various methods are summarized in Table 4. Costs with the label **Optimized** correspond to methods that have been slightly optimized by saving some initial computations. This technique is similar to that proposed in [13, Section 4.2.2] plus some additional savings gained with the use of composite operations (i.e., tripling, quintupling).

The "basic" operation count (without the aforementioned optimization) is detailed per method. In the case of windowed methods, the count is given separately for 7 and 6 precomputations (the latter case always corresponds to Jacobian coordinates only). Also, note that for Jacobian coord. we use doubling-addition (DA) operations instead of traditional additions. Hence, in this case, the total number of doublings is obtained by subtracting the number of doublings listed by that of additions. Finally, for case 2 (Jacobian), the total number of mDA operations is obtained by adding numbers listed in mDA and DA, as precomputed points are in affine and all the additions involve mixed coordinates.

As can be seen, in scenarios without precomputations, the basic Multibase NAF using bases {2,3} and {2,3,5} achieve better performance than the original DB method based on the "Greedy" algorithm [8]. That is in addition to the attractive features of Multibase NAF such as simplicity and memory efficiency. More recently, Doche et al. [10] introduced a new method that also finds double-base chains without using the "Greedy" algorithm, although using a somewhat more complex search-based approach in comparison with the basic Multibase NAF. This method's cost is comparable to (2,3)NAF, but slightly higher than that achieved by (2,3,5)NAF. More importantly, the proposed Refined Multibase NAF method presents even lower costs in all the cases, bases {2,3} and {2,3,5}. The improvement is especially significant in the case without precomputations,

Table 4. Comparison of scalar multiplication methods ($n = 160$ bits; $1S = 0.8M$)

Method	# pts	Operation count					JQuartic		InvEdw		Jacobian	
		D	T	Q	mA (mDA)	A (DA)	Precomp	Total	Precomp	Total	Precomp	Total
Refined (2,3,5)NAF (this work)	0	109.9	21.6	6.2	31.0	-	0	**1267.5M**	-	-	0	**1459.6M**
Refined (2,3)NAF (this work)	0	120.6	23.9	-	35.6	-	0	1288▷4M	0	**1350.5M**	0	1484▷8M
(2,3,5)NAF, Longa [17]	0	96.9	24.3	10.0	32.0	-	0	1292▷6M	-	-	0	1485▷9M
(2,3)NAF, Longa [17]	0	113.6	28.4	-	37.8	-	0	1314▷9M	0	1381▷0M	0	1513▷3M
Tree-based DB, Doche et al. [10]	0	91.1	42.9	-	32.8	-	0	1302▷6M	0	1376▷8M	0	1493▷1M
DB (Greedy), Dimitrov et al. [8]	0	94.2	40.9	-	36.3	-	0	1328▷2M	0	1402▷5M	0	1523▷2M
NAF	0	158.7	-	-	52.9	-	0	1396▷6M	0	1449▷5M	0	1618▷7M
Optimized	7	-					59▷6M	**1214.1M**	-	-	55▷4M	**1425.8M**
	6 (Jac)	-					-	-	-	-	$1I$+72▷4M	**1I+1393.5M**
Refined (2,3,5)NAF (this work)	7	132.4	10.0	3.8	2.9	18.2	59▷6M	1226▷5M	-	-	55▷4M	1435▷2M
	6 (Jac)	129.8	10.9	4.3	5.0	16.3	-	-	-	-	$1I$+72▷4M	$1I$+1406▷5M
Optimized	7	-					59▷6M	1216▷0M	61▷2M	**1265.6M**	55▷4M	1434▷1M
	6 (Jac)	-					-	-	-	-	$1I$+72▷4M	$1I$+1396▷4M
Refined (2,3)NAF (this work)	7	143.9	8.3	-	3.0	20.0	59▷6M	1229▷2M	61▷2M	1277▷1M	55▷4M	1445▷1M
	6 (Jac)	142.2	9.4	-	5.1	18.3	-	-	-	-	$1I$+72▷4M	$1I$+1410▷9M
(2,3,5)NAF, Longa [17]	7	129.5	10.5	5.1	2.7	18.8	59▷6M	1240▷9M	-	-	55▷4M	1450▷4M
	6 (Jac)	124.0	12.6	6.0	5.5	16.0	-	-	-	-	$1I$ + 72▷4M	$1I$+1422▷4M
(2,3)NAF, Longa [17]	7	140.2	11.0	-	2.9	20.4	59▷6M	1240▷5M	61▷2M	1290▷5M	55▷4M	1456▷0M
	6 (Jac)	136.1	13.6	-	5.8	17.8	-	-	-	-	$1I$+72▷4M	$1I$+1423▷3M
Optimized (this work)	7	-					59▷6M	1233▷1M	61▷2M	1276▷2M	55▷4M	1463▷1M
	6 (Jac)	-					-	-	-	-	$1I$+72▷4M	$1I$+1419▷5M
Frac-wNAF	7	157.3	-	-	3.3	22.8	59▷6M	1245▷9M	61▷2M	1289▷5M	55▷4M	1476▷4M
	6 (Jac)	157.4	-	-	7.0	20.3	-	-	-	-	$1I$+72▷4M	$1I$+1436▷4M
DB (Greedy), Bernstein et al. [1]	7	N/A					N/A	1311▷0M	N/A	1290▷3M	N/A	1504▷3M
NAF-based, Bernstein et al. [2]	7	N/A					N/A	1308▷5M	N/A	1287▷8M	N/A	1511▷9M
	8	N/A					-	-	-	-	-	$1I$+1434▷1M

which makes our method especially interesting for applications on constrained devices. With Jacobi quartics, the advantage of the Refined *mb*NAF using bases {2,3,5} is as large as 9.3% over the traditional NAF. In Jacobian coord., that advantage rises to 9.8%.

Interestingly enough, in the case of windowed methods, we observe that the refined multibase algorithms surpass the performance of traditional binary methods for all the curve shapes analyzed, contradicting conclusions by [1] and [10]. Most remarkably, if we consider the "basic" operation count, the Refined (2,3,5)NAF with no precomputations is comparable and/or surpasses the performance of the fastest NAF method using an optimal window with 7 and 6 precomputed points for Jacobi quartics and Jacobian coordinates, respectively. For the latter, the multibase method is superior always that $1I > 23M$. Even if we consider the optimized version of the NAF method, the multibase method achieves higher performance always that $1I > 40M$.

For the record, we also include results by [2] and [1]. These works use highly optimized radix-2 and double-base (DB) scalar multiplications. We can see that both the basic Multibase NAF using precomputations and the refined version offer lower computing costs for the cases when precomputations include one or nil field inversions. Moreover, our optimized implementations of *w*NAF and Frac-*w*NAF are also superior in performance to these works. The latter is due to a combination of improved precomputation schemes, more efficient point formulas and the inclusion of the technique to save initial computations.

In particular, the Refined *mb*NAF using 6 and 7 precomputed points achieves the highest performance using bases {2,3,5} in the case of standard curves and Jacobi quartics. In the case of Edwards curves using inverted Edwards coordinates the lowest cost is achieved by the same method using bases {2,3} and 7 points. (The lowest costs per curve are highlighted in **bold**.) Also, note that for Jacobian the highest speed up is achieved with a table of the form {3,5,...,13} (6 points; 160 bits), which corresponds to a fractional window and, thus, highlights the importance of this recoding for Jacobian coordinates.

7 Conclusion

We have introduced a refined multibase method and other several optimizations, including improved point operation formulas, that have been efficiently applied to speed up (multibase) methods for scalar multiplication. In particular, we have applied the concept of "fractional" windows to the multibase scenario, generalizing Multibase NAF methods to any number of precomputations. Also, we have presented a more comprehensive analysis of scalar multiplications methods and tested their performance in comparison with Multibase NAF methods using different elliptic curve shapes. The conclusion is that currently the proposed Refined *mb*NAF achieves the lowest costs found in the literature among methods without precomputations, independently of the curve selected. Using bases {2,3,5} and {2,3} we can perform a scalar multiplication with costs of only $1459M$ (field multiplications) and $1350M$ in Jacobian and inverted Edwards coordinates (respect.). With Jacobi quartics, that cost can be as low as $1267M$ using bases {2,3,5}. Similar results are attained by the same method when using precomputations. In this case, we present the lowest costs reported in the literature: $1425M$ or $1I + 1393M$ in Jacobian, $1265M$ in inverted Edwards and $1214M$ in extended Jacobi quartic coordinates.

Finally, we have included the theoretical analysis of Multibase NAF and its different variants, detailing the average zero and nonzero density characterizing these representations. This analysis has been confirmed with our extensive tests.

Acknowledgments. We would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Centres of Excellence (OCE) for partially supporting this work. We would also like to thank the reviewers for their useful comments.

References

1. Bernstein, D., Birkner, P., Lange, T., Peters, C.: Optimizing Double-Base Elliptic-Curve Single-Scalar Multiplication. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 167–182. Springer, Heidelberg (2007)
2. Bernstein, D., Lange, T.: Analysis and Optimization of Elliptic-Curve Single-Scalar Multiplication. Cryptology ePrint Archive, Report 2007/455 (2007)
3. Bernstein, D., Lange, T.: Faster Addition and Doubling on Elliptic Curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
4. Bernstein, D., Lange, T.: Inverted Edwards Coordinates. In: Boztaş, S., Lu, H.-F(F.) (eds.) AAECC 2007. LNCS, vol. 4851, pp. 20–27. Springer, Heidelberg (2007)
5. Billet, O., Joye, M.: The Jacobi Model of an Elliptic Curve and Side-Channel Analysis. In: Fossorier, M.P.C., Høholdt, T., Poli, A. (eds.) AAECC 2003. LNCS, vol. 2643, pp. 34–42. Springer, Heidelberg (2003)
6. Ciet, M., Joye, M., Lauter, K., Montgomery, P.L.: Trading Inversions for Multiplications in Elliptic Curve Cryptography. Designs, Codes and Cryptography 39(2), 189–206 (2006)
7. Dimitrov, V., Jullien, G., Miller, W.: Theory and Applications for a Double-Base Number System. ARITH 1997, p. 44 (1997)
8. Dimitrov, V., Imbert, L., Mishra, P.K.: Efficient and Secure Elliptic Curve Point Multiplication using Double-Base Chains. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 59–78. Springer, Heidelberg (2005)
9. Dimitrov, V., Mishra, P.K.: Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication using Multibase Number Representation. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 390–406. Springer, Heidelberg (2007)
10. Doche, C., Habsieger, L.: A Tree-Base Approach for Computing Double-Base Chains. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 433–446. Springer, Heidelberg (2008)
11. Doche, C., Imbert, L.: Extended Double-Base Number System with Applications to Elliptic Curve Cryptography. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 335–348. Springer, Heidelberg (2006)

12. Edwards, H.: A Normal Form for Elliptic Curves. Bulletin of the American Mathematical Society 44, 393–422 (2007)
13. Elmegaard-Fessel, L.: Efficient Scalar Multiplication and Security against Power Analysis in Cryptosystems based on the NIST Elliptic Curves over Prime Fields. Master Thesis, University of Copenhagen (2006)
14. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004)
15. Hisil, H., Wong, K., Carter, G., Dawson, E.: Faster Group Operations on Elliptic Curves. Cryptology ePrint Archive, Report 2007/441 (2007)
16. Hisil, H., Wong, K., Carter, G., Dawson, E.: An Intersection Form for Jacobi-Quartic Curves. Personal communication (2008)
17. Longa, P.: Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields. Master Thesis, University of Ottawa (2007), `http://patricklonga.bravehost.com/publications.html`
18. Longa, P.: ECC Point Arithmetic Formulae (EPAF), `http://patricklonga.bravehost.com/jacobian.html`
19. Longa, P., Gebotys, C.: Setting Speed Records with the (Fractional) Multibase Non-Adjacent Form Method for Efficient Elliptic Curve Scalar Multiplication. CACR Technical Report, CACR 2008-06, University of Waterloo (2008)
20. Longa, P., Gebotys, C.: Novel Precomputation Schemes for Elliptic Curve Cryptosystems. Cryptology ePrint Archive, Report 2008/526 (2008)
21. Longa, P., Miri, A.: Fast and Flexible Elliptic Curve Point Arithmetic over Prime Fields. IEEE Trans. Comp. 57(3), 289–302 (2008)
22. Longa, P., Miri, A.: New Composite Operations and Precomputation Scheme for Elliptic Curve Cryptosystems over Prime Fields. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 229–247. Springer, Heidelberg (2008)
23. Meloni, N.: New Point Addition Formulae for ECC Applications. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 189–201. Springer, Heidelberg (2007)
24. Möller, B.: Improved Techniques for Fast Exponentiation. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
25. Möller, B.: Fractional windows revisited:Improved signed-digit representations for efficient exponentiation. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 137–153. Springer, Heidelberg (2005)

A Derivation of Composite Operations of Form *dP*

Consider the following formula due to [23] to add two points $P = (X_1, Y_1, Z)$ and $Q = (X_2, Y_2, Z)$ with the same coordinate Z in Jacobian coordinates:

$$\begin{aligned} X_3 &= (Y_2 - Y_1)^2 - (X_2 - X_1)^3 - 2X_1(X_2 - X_1)^2,\ Z_3 = Z(X_2 - X_1) \\ Y_3 &= (Y_2 - Y_1)(X_1(X_2 - X_1)^2 - X_3) - Y_1(X_2 - X_1)^3\ . \end{aligned} \tag{5}$$

To derive composite operations of the form dP, where $d > 2$ is a small prime, we follow the next scheme using (5) to perform additions:

$$dP = (2P + (\dots (2P + (2P + P)) \dots))\ . \tag{6}$$

According to (6), we first compute $2P$ with the following [21]:

$$X_2 = [3(X_1 + Z_1^2)(X_1 - Z_1^2)]^2 - 8X_1Y_1^2,\ Z_2 = 2Y_1Z_1 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2,$$
$$Y_2 = [3(X_1 + Z_1^2)(X_1 - Z_1^2)](4X_1Y_1^2 - X_2) - 8Y_1^4 .$$

And then, we perform the addition $3P = 2P + P = (X_2, Y_2, Z_2) + (X_1^{(1)}, Y_1^{(1)}, Z_1^{(1)}) = (X_3, Y_3, Z_3)$ using formula (5), where $(X_1^{(1)}, Y_1^{(1)}, Z_1^{(1)}) = (X_1(4Y_1^2), Y_1(8Y_1^3), Z_1(2Y_1)) \equiv (X_1, Y_1, Z_1)$, as follows:

$$X_3 = (Y_1^{(1)} - Y_2)^2 - (X_1^{(1)} - X_2)^3 - 2X_2(X_1^{(1)} - X_2)^2,$$
$$Y_3 = (Y_1^{(1)} - Y_2)[X_2(X_1^{(1)} - X_2)^2 - X_3] - Y_2(X_1^{(1)} - X_2)^3,\ Z_3 = Z_2(X_1^{(1)} - X_2) .$$

After scaling and replacement of some multiplications by squarings, computation of $3P$ takes the form:

$$X_3 = \omega^2 - 4\theta^3 - 8X_2\theta^2,\ Y_3 = \omega[4X_2\theta^2 - X_3] - 8Y_2\theta^3,\ Z_3 = 2Z_1^{(1)}\theta, \tag{7}$$

where $\alpha = 3(X_1 + Z_1^2)(X_1 - Z_1^2)$, $\theta = 4X_1Y_1^2 - X_2$, $\omega = 16Y_1^4 - 2Y_2$, $X_2 = \alpha^2 - 8X_1Y_1^2$, $2Y_2 = (\alpha + \theta)^2 - \alpha^2 - \theta^2 - 16Y_1^4$, $Z_1^{(1)} = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2$. Following the same approach for the next addition in (6), it is easy to derive the formula for the quintupling of a point $5P = (X_5, Y_5, Z_5)$ in Jacobian coordinates (special case $a = -3$). The new formula is given by:

$$\begin{gathered} X_5 = \gamma^2 - 4\phi^3 - 8X_2^{(1)}\phi^2,\ Y_5 = \gamma[4X_2^{(1)}\phi^2 - X_5] - 8Y_2^{(1)}\phi^3, \\ Z_5 = 2Z_2[(\theta + \phi)^2 - \theta^2 - \phi^2], \end{gathered} \tag{8}$$

where $\alpha = 3(X_1 + Z_1^2)(X_1 - Z_1^2)$, $\theta = X_1^{(1)} - X_2$, $\omega = 2Y_1^{(1)} - 2Y_2$, $X_2 = \alpha^2 - 2X_1^{(1)}$, $2Y_2 = (\alpha + \theta)^2 - \alpha^2 - \theta^2 - 2Y_1^{(1)}$, $Z_2 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2$, $\gamma = \omega^2 + \phi^2 - (\omega + \phi)^2 - 4Y_2^{(1)}$, $X_1^{(1)} = 4X_1Y_1^2$, $2Y_1^{(1)} = 16Y_1^4$, $X_2^{(1)} = 4X_2\theta^2$, $Y_2^{(1)} = 8Y_2\theta^3$, $\phi = \omega^2 - 4\theta^3 - 3X_2^{(1)}$.

This quintupling formula costs $10M + 12S$. In the general case (random a), the cost is fixed at $9M + 15S$ with the following change of parameters: $\alpha = 3X_1^2 + aZ_1^4$, $X_1^{(1)} = 2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4]$.

Again, following the same procedure for the next addition in (6), it is straightforward to derive the formula for the septupling $7P = (X_7, Y_7, Z_7)$ in Jacobian coord. (case $a = -3$). The new septupling formula is given by:

$$X_7 = \varphi^2 - 4\sigma^3 - 8X_2^{(2)}\sigma^2,\ Y_7 = \varphi[4X_2^{(2)}\sigma^2 - X_7] - 8Y_2^{(2)}\sigma^3,\ Z_7 = 2Z_2^{(2)}\sigma, \tag{9}$$

where $\varphi = \gamma^2 + \sigma^2 - (\gamma + \sigma)^2 - 4Y_2^{(2)}$, $\sigma = \gamma^2 - 4\phi^3 - 3X_2^{(2)}$, $\gamma = \omega^2 + \phi^2 - (\omega + \phi)^2 - 4Y_2^{(1)}$, $X_2^{(2)} = 4X_2^{(1)}\phi^2$, $Y_2^{(2)} = 8Y_2^{(1)}\phi^3$, $Z_2^{(2)} = 2Z_2[(\theta + \phi)^2 - \theta^2 - \phi^2]$, $\phi = \omega^2 - 4\theta^3 - 3X_2^{(1)}$, $\omega = 2Y_1^{(1)} - 2Y_2$, $\theta = X_1^{(1)} - X_2$, $\alpha = 3(X_1 + Z_1^2)(X_1 - Z_1^2)$, $X_2^{(1)} = 4X_2\theta^2$, $Y_2^{(1)} = 8Y_2\theta^3$, $X_2 = \alpha^2 - 2X_1^{(1)}$, $X_1^{(1)} = 4X_1Y_1^2$, $2Y_1^{(1)} = 16Y_1^4$, $2Y_2 = (\alpha + \theta)^2 - \alpha^2 - \theta^2 - 2Y_1^{(1)}$, $Z_2 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2$. This septupling formula costs $14M + 15S$. In the general case (parameter a random), the cost is fixed at $13M + 18S$ with the following change of parameters: $\alpha = 3X_1^2 + aZ_1^4$, $X_1^{(1)} = 2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4]$.

B Proof of the Average Zero and Nonzero Densities of $(w)mb$NAF Using Bases {2,3} and {2,3,5}

The method can be modeled as a Markov chain with three states (case $\mathcal{A}$=$\{2,3\}$) : ”$0^{(2)}$”, ”$0^{(3)}$” and ” $\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}$”, with the following probability matrix:

$$\left(\begin{array}{lcccc} \text{”}0^{(2)}\text{”} & : & 1/2 & \frac{2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor}{2^w} & \frac{2^{w-2}+\lfloor(2^{w-2}+1)/3\rfloor}{2^w} \\ \text{”}0^{(3)}\text{”} & : & 0 & 1/3 & 2/3 \\ \text{”}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}\text{”} & : & 1/2 & \frac{2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor}{2^w} & \frac{2^{w-2}+\lfloor(2^{w-2}+1)/3\rfloor}{2^w} \end{array}\right)$$

This Markov chain is irreducible and aperiodic, and hence, it has stationary distribution, which is given by:

$$\left(\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)},0^{(2)},0^{(3)}\ :\ \frac{2^w}{2^{w+1}+3(2^{w-2}-s)}\ \frac{2^w}{2^{w+1}+3(2^{w-2}-s)}\ \frac{3(2^{w-2}-s)}{2^{w+1}+3(2^{w-2}-s)})\right)$$

Thus, nonzero digits k_i appear 2^w out of $w{\cdot}2^w{+}2^w{+}3(2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor)$, which proves our assertion about the nonzero density. Doublings and triplings (i.e., number of zero and nonzero digits with base 2 and 3, respect.) appear $2^w\cdot w{+}2^w$ and $3(2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor)$ out of $w{\cdot}2^w{+}2^w{+}3(2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor)$, respectively. This proves our assertion about the average density of doublings and triplings.

In the case of $\mathcal{A}=\{2,3,5\}$, there are four states: ”$0^{(2)}$”, ”$0^{(3)}$”, ”$0^{(5)}$” and ” $\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}$”. The probability matrix in this case is as follows (see Theorem 2 for notation):

$$\left(\begin{array}{lccccc} \text{”}0^{(2)}\text{”} & : & 1/2 & \frac{2^{w-2}-s}{2^w} & \frac{2^{w-2}-r+s-t}{2^{w+2}} & \frac{3\cdot2^{w-2}+r+3s+t}{2^{w+2}} \\ \text{”}0^{(3)}\text{”} & : & 0 & 1/3 & 1/6 & 1/2 \\ \text{”}0^{(5)}\text{”} & : & 0 & 0 & 1/5 & 4/5 \\ \text{”}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}\text{”} & : & 1/2 & \frac{2^{w-2}-s}{2^w} & \frac{2^{w-2}-r+s-t}{2^{w+2}} & \frac{3\cdot2^{w-2}+r+3s+t}{2^{w+2}} \end{array}\right)$$

This Markov chain is irreducible and aperiodic with stationary distribution:

$$\left(\text{”}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}\text{”},\text{”}0^{(2)}\text{”},\text{”}0^{(3)}\text{”},\text{”}0^{(5)}\text{”}\ :\ \frac{2^{w+3}}{\omega}\ \frac{2^{w+3}}{\omega}\ \frac{24(2^{w-2}-s)}{\omega}\ \frac{5(2^{w-1}-r-t)}{\omega}\right)$$

where $\omega=49{\cdot}2^{w-1}{-}5r{-}24s{-}5t$. Therefore, nonzero digits k_i appear 2^{w+3} out of $2^{w+3}{\cdot}w{+}2^{w+3}{+}24(2^{w-2}-s){+}5(2^{w-1}-r-t)$, which proves our assertion about the nonzero density. Doublings, triplings and quintuplings appear $2^{w+3}\cdot w+2^{w+3}$, $24(2^{w-2}-s)$ and $5(2^{w-1}-r-t)$ out of $2^{w+3}\cdot w+2^{w+3}+24(2^{w-2}-s)+5(2^{w-1}-r-t)$, respectively. This proves our assertion about the average density of the aforementioned operations.

C Proof of the Average Zero and Nonzero Densities of Fractional *wmb*NAF Using Bases {2,3}

Let us consider the following states to model this fractional windows method using Markov chains: ”$0^{(2)}$”, ”$0^{(3)}$”, ”$\underbrace{0^{(2)}\ldots 0^{(2)}}_{w-2} k_i^{(2)}$”, and ”$\underbrace{0^{(2)}\ldots 0^{(2)}}_{w-1} k_i^{(2)}$”. Then, the probability matrix is as follows (see Theorem 3 for notation):

$$\left(\begin{array}{lccccc}
\text{”}0^{(2)}\text{”} & : & 1/2 & \frac{t-\lfloor (t+1)/3\rfloor}{4t} & \frac{(2^{w-2}-t)(t+\lfloor (t+1)/3\rfloor)}{2^w t} & \frac{t+\lfloor (t+1)/3\rfloor}{2^w} \\
\text{”}0^{(3)}\text{”} & : & 0 & 1/3 & \frac{2^{w-2}-t}{3\cdot 2^{w-3}} & \frac{t}{3\cdot 2^{w-3}} \\
\text{”}\underbrace{0^{(2)}\ldots 0^{(2)}}_{w-2} k_i^{(2)}\text{”} & : & 0 & \alpha & \beta & 1-\alpha-\beta \\
\text{”}\underbrace{0^{(2)}\ldots 0^{(2)}}_{w-1} k_i^{(2)}\text{”} & : & 1/2 & \frac{t-\lfloor (t+1)/3\rfloor}{4t} & \frac{(2^{w-2}-t)(t+\lfloor (t+1)/3\rfloor)}{2^w t} & \frac{t+\lfloor (t+1)/3\rfloor}{2^w}
\end{array}\right)$$

where $\alpha = \frac{(2^{w-2}-t)-\lfloor (2^{w-2}-t+1)/3\rfloor}{2(2^{w-2}-t)}$ and $\beta = \frac{(2^{w-2}-t)+\lfloor (2^{w-2}-t+1)/3\rfloor}{2^{w-1}}$. This Markov chain is irreducible and aperiodic with the following stationary distribution:

$$\left(\underbrace{0^{(2)}\ldots 0^{(2)}}_{w-1} k_i^{(2)}, \underbrace{0^{(2)}\ldots 0^{(2)}}_{w-2} k_i^{(2)}, 0^{(2)}, 0^{(3)} \;:\; \frac{16t}{\mu} \;\; \frac{12((u+v)-2^{w-2})}{\mu} \;\; \frac{16(t-2^{w-2})}{\mu} \;\; \frac{16t}{\mu}\right)$$

where $\mu = 16t - 12(u+v) + 7\cdot 2^w$. Therefore, the nonzero digits k_i appear 2^w out of $8t - 3(u+v) + 2^{w-2}(4w-1)$, which proves our assertion about the nonzero density. Doublings and triplings appear $8t + 2^w(w-1)$ and $3(2^{w-2} - (u+v))$ out of $8t - 3(u+v) + 2^{w-2}(4w-1)$, respectively. This proves our assertion about the average density of doublings and triplings.

Revocable Group Signature Schemes with Constant Costs for Signing and Verifying

Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki

Department of Communication Network Engineering, Okayama University,
3-1-1 Tsushima-Naka, Okayama 700-8530, Japan
{nakanisi,funabiki}@cne.okayama-u.ac.jp

Abstract. Lots of revocable group signature schemes have been proposed so far. In one type of revocable schemes, signing and/or verifying algorithms have $O(N)$ or $O(R)$ complexity, where N is the group size and R is the number of revoked members. On the other hand, in Camenisch-Lysyanskaya scheme and the followers, signing and verifying algorithms have $O(1)$ complexity. However, before signing, updates of the secret key are required. The complexity is $O(R)$ in the worst case. In this paper, we propose a revocable scheme with signing and verifying of $O(1)$ complexity, where no updates of secret key are required. The compensation is the long public key of $O(N)$. In addition, we extend it to the scheme with $O(\sqrt{N})$-size public key, where signing and verifying have constant extra costs.

1 Introduction

Group signatures [15] allow a signer to sign a message anonymously as a group member, while only a designated trusted party can identify the signer from the signature. The group is managed by a group manager (GM) who permits a user to join the group. For simplicity, this paper assumes that GM is also the trusted party to identify the signer (This assumption can be easily relaxed). The applications of group signatures include anonymous credentials, direct anonymous attestations, and ID management reported in [13,11,20].

Toward making the group signatures practicable, Boneh et al. have proposed a short group signature scheme based on pairings [7], where signatures are shorter than existing RSA-based group signature schemes. With the advance of the implementations of pairings (e.g., [3,19]), we can obtain the implementations of the signing/verifying of the group signatures with practical computation times and data sizes, if the revocation is neglected. The revocation means that the membership of a group member can be easily revoked to exclude the member from the group.

Lots of revocable group signature schemes have been proposed so far (e.g, [10,14,24,7,8,12]). However, one type of schemes [10,24,8] has a disadvantage: Singing and/or verifying have the computational complexity (exponentiations or pairings) of $O(N)$ or $O(R)$, where N is the group size and R is the number of

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 463–480, 2009.

revoked members. Thus, these schemes are not suitable for large and dynamic groups.

Camenisch and Lysyanskaya proposed an elegant revocable scheme [14] using dynamic accumulators. In this scheme, the complexity of signing/verifying is $O(1)$ w.r.t. N and R. However, the disadvantage of this scheme is that, whenever making a signature, the signer has to modify his secret key. The modification requires some data of joining and removed members since the last time he signed. This implies that a signer requires a computation of $O(N)$. This is relaxed into $O(R)$ in [12,7]. However, since R becomes large in large and dynamic groups, we should explore another approach for signing/verifying with $O(1)$ costs.

Another RSA-based approach without the key update is proposed in [23]. In the approach, the large group is partitioned to sub-groups, and the scheme of [24] is utilized for each smaller sub-groups. The scheme of [24] achieves $O(1)$ exponentiations on signing/verifying for middle-scale sub-groups with less than about 1000 members. The computational cost on partitioning in [23] is also $O(1)$. Thus, this approach achieves signing/verifying with $O(1)$ costs. However, this approach has a weakness for larger groups; The signer has to obtain public data reflecting the current membership situation for each member, but the size of the data is $O(N)$. Namely, the signer has to fetch the data with $O(N)$ size whenever signing. Another problem is that it is based on RSA, and thus the long RSA modulus leads to long signatures and revocation data. Therefore, for larger groups, the communication cost becomes vast.

Our contributions. In this paper, we propose a pairing-based revocable scheme satisfying

1. signing/verifying requires only $O(1)$ computational costs,
2. any update of member's secret key is not required, and
3. data related to revocation, which is fetched by the signer, has $O(R)$ size.

Therefore, signing/verifying is sufficiently fast even for larger or dynamic groups, while the communication does not cause large delay.

On the other hand, the compensation is the long public key with $O(N)$ size. In general applications (e.g., authentications for Web services), the public key is distributed once in joining, together with the software of the applications. Thus, the compensation is not so serious. However, for millions of members, the long time of downloading may disgust users. To solve this, in this paper, we also propose an extended scheme with $O(\sqrt{N})$-size public key, where signing/verifying has constant extra costs.

The reduction to the $O(\sqrt{N})$ size means the sufficient practicality of our schemes for large groups. In case of the 112-bit security level and $N = 1,000,000$, the public key size of the extended scheme is less than 100 KBytes. This size shows the sufficient practicality of the storage, not only in usual PCs but also in smart phones. Furthermore, since clients have only to download the public key once, the communication cost does not matter.

We can consider a trivial revocation method, where a non-revoked member fetches a new secret key whenever revocation happens or a time interval proceeds.

To fetch it anonymously, the signer has to fetch the keys of whole non-revoked members. This means fetching $O(N)$-size data. In our scheme, the fetched data is reduced to $O(R)$ size.

The signer's fetching $O(R)$-size date is a concern for the realization of group signatures. However, except the VLR (Verifier-Local Revocation) method [8], every revocation mechanism such as [10,14,7,12,24] needs such a communication cost. Note that the VLR mechanism requires $O(R)$ computational cost in the verification. Thus, we consider that our revocation method is currently better than the other methods in the natural situation that the signer can use a broadband channel. The reduction in the fetched data is an open problem.

Related works. One of trends in researches on group signatures is to exclude the random oracle model. From the viewpoint of both efficiency and security, Groth's scheme [18] is currently the best choice. Although this scheme achieves the constant length of signatures, it still is greatly less efficient than efficient schemes [7,16] in the random oracle model. Since our aim is the realization of group signatures in practical applications, we adopt the efficient underlying scheme [16] to add our novel revocation mechanism. This means that our scheme is secure in the random oracle model. A construction without the random oracle is one of our future works.

2 Model and Security Definitions

We show a model of revocable group signature scheme. This model and the following security requirements are derived from [8,5]. Mainly, for the simplicity, we construct our model on the basis of the model of [8]. The differences from [8] are the revocation mechanism, **Join** algorithm (and non-frameability), and **Open** algorithm. In [8], **Verify** algorithm is given revocation list $\boldsymbol{RL}_t$, but in our model, **Sign** algorithm is given $\boldsymbol{RL}_t$. We consider that the non-frameability (i.e., preventing GM from forging a member's signature) is important, and thus, add **Join** algorithm and the definition of the non-frameability, based on [5]. Also, the **Open** algorithm is added for the purpose of identifying an illegal member.

Revocable group signature scheme consists of the following algorithms:

Setup: This probabilistic initial setup algorithm, on input 1^ℓ, outputs public parameters $param$.

KeyGen: This probabilistic key generation algorithm for GM, on input N that is the maximum number of members and $param$, outputs the group public key gpk and GM's secret key msk.

Join: This is an interactive protocol between a probabilistic algorithm **Join-U** for the i-th user and a probabilistic algorithm **Join-GM** for GM, where the user joins the group controlled by GM w.r.t. gpk. **Join-U**, on input gpk, outputs $\boldsymbol{usk}[i]$ that is the user's secret key. On the other hand, **Join-GM**, on inputs gpk, msk, outputs $\boldsymbol{reg}[i]$, which means the registration log of the i-th user.

$\boldsymbol{usk}$ denotes the list of all users' secret keys, $\boldsymbol{reg}$ denotes the list of all users' registration log, and i denotes user's ID. We index the secret key and registration log of user i by $\boldsymbol{usk}[i]$ and $\boldsymbol{reg}[i]$, respectively.

Revoke: This probabilistic algorithm, on inputs gpk, t and RU that is a set of revoked members' IDs, outputs revocation list $\boldsymbol{RL}_t$.

t denotes a time counter, and $\boldsymbol{RL}_t$ denotes the revocation list of data on revoked users at time t.

Sign: This probabilistic algorithm, on inputs gpk, $\boldsymbol{usk}[i]$, t, $\boldsymbol{RL}_t$, and signed message M, outputs the signature σ.

Verify: This is a deterministic algorithm for verification. The input is gpk, t, a signature σ, and the message M. Then the output is 'valid' or 'invalid'.

Open: This deterministic algorithm, on inputs gpk, msk, $\boldsymbol{reg}$, t, σ and M, outputs i, which indicates the signer of σ.

The security requirements, *Traceability, Anonymity*, and *Non-frameability*, are defined as follows.

2.1 Traceability

The following traceability requirement captures the unforgeability and the revocability of group signatures. Consider the following traceability game between an adversary $\mathcal{A}$ and a challenger, where $\mathcal{A}$ tries to forge a signature that cannot be traced to one of members corrupted by $\mathcal{A}$ or to forge a signature that is traced to a revoked member corrupted by $\mathcal{A}$.

Setup: The challenger runs **Setup** and **KeyGen**, and obtains gpk and msk. He provides $\mathcal{A}$ with gpk, and run $\mathcal{A}$. He sets $t = 0$ and CU and RU with empty, where CU denotes the set of IDs of users corrupted by $\mathcal{A}$, and RU denotes the set of IDs of revoked users.

Queries: $\mathcal{A}$ can query the challenger about the following.

H-Join: $\mathcal{A}$ can request the i-th user's join. Then, the challenger executes the join protocol, where the challenger plays the both role of the joining user and GM.

C-Join: $\mathcal{A}$ can request the i-th user's join. Then, $\mathcal{A}$ as the joining user executes the join protocol with the challenger as GM. The challenger adds i to CU.

Revocation: $\mathcal{A}$ requests the revocation of a member i. The challenger increases t by 1, add i to RU, and responds $\boldsymbol{RL}_t$ for t and RU.

Signing: $\mathcal{A}$ requests a signature on a message M for a member i. The challenger responds the corresponding signature using the current $\boldsymbol{RL}_t$, if $i \notin CU$.

Corruption: $\mathcal{A}$ requests the secret key of a member i. The challenger responds $\boldsymbol{usk}[i]$ if $i \notin CU$. The challenger adds i to CU.

Open: $\mathcal{A}$ requests to open a signature σ on the message M. The challenger responds the corresponding signer's ID i.

Output: Finally, $\mathcal{A}$ outputs a message M^* and a signature σ^* on the current $\boldsymbol{RL}_t$.

Then, $\mathcal{A}$ wins if

1. $\mathbf{Verify}(gpk, t, \sigma^*, M^*) = \text{valid}$,
2. for $i^* = \mathbf{Open}(gpk, msk, \boldsymbol{reg}, t, \sigma^*, M^*)$, $i^* \notin CU \setminus RU$, and
3. $\mathcal{A}$ did not obtain σ^* by making a signing query at M^*.

Traceability requires that for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the traceability game is negligible.

2.2 Anonymity

The following anonymity requirement captures the anonymity and unlinkability of signatures. Consider the following anonymity game.

Setup: The challenger runs **KeyGen**, and obtains gpk and msk. He provides $\mathcal{A}$ with gpk, and run $\mathcal{A}$. He sets $t = 0$ and RU and CU with empty.

Queries: $\mathcal{A}$ can query the challenger. The available queries are the same ones as in the traceability game.

Challenge: $\mathcal{A}$ outputs a message M and two members i_0 and i_1. If $i_0 \notin CU$ and $i_1 \notin CU$, the challenger chooses $\phi \in_R \{0, 1\}$, and responds the signature on M of member i_ϕ using the current $\boldsymbol{RL}_t$.

Restricted queries: Similarly, $\mathcal{A}$ can make the queries. However, $\mathcal{A}$ cannot query opening of the signature responded in the challenge.

Output: Finally, $\mathcal{A}$ outputs a bit ϕ' indicating its guess of ϕ.

If $\phi' = \phi$, $\mathcal{A}$ wins. We define the advantage of $\mathcal{A}$ as $|\Pr[\phi' = \phi] - 1/2|$.

Anonymity requires that for all PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ on the anonymity game is negligible.

2.3 Non-frameability

This property requires that a signature of an honest member cannot be computed by other members and even GM.

Consider the following non-frameability game.

Setup: The challenger uses **Setup** to obtain $param$, and sets $t = 0$ and RU and HU with empty, where HU denotes the set of IDs of honest users who are not corrupted by $\mathcal{A}$. Then, run $\mathcal{A}$ on $param$, who initially outputs gpk.

Queries: After the key output in the run, $\mathcal{A}$ issues the following queries to the challenger.

Join: $\mathcal{A}$ can request the i-th honest user's join. Then, $\mathcal{A}$ as GM executes the join protocol with the challenger as the i-the user. The challenger adds i to HU.

Revocation: $\mathcal{A}$ can request the revocation of member i. The challenger increases t by 1, add i to RU, and responds $\boldsymbol{RL}_t$ for t and RU.

Sign: $\mathcal{A}$ can request a signature on message M for user's ID i using the current $\boldsymbol{RL}_t$. The challenger replies $\textbf{Sign}(gpk, \boldsymbol{usk}[i], t, \boldsymbol{RL}_t, M)$, if $i \in HU$.

Corruption: $\mathcal{A}$ can request to corrupt a member by sending the member's ID i. The challenger returns $\boldsymbol{usk}[i]$, if $i \in HU$. The challenger deletes i from HU.

Output: Finally, $\mathcal{A}$ outputs a message M^* and a signature σ^*.

Then, $\mathcal{A}$ wins if

1. $\textbf{Verify}(gpk, t, \sigma^*, M^*) =$ valid,
2. for $i^* = \textbf{Open}(gpk, msk, \boldsymbol{reg}, t, \sigma^*, M^*)$, $i^* \in HU$, and
3. $\mathcal{A}$ did not obtain σ^* by making a signing query at M^*.

Non-Frameability requires that for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the non-frameability game is negligible.

3 Preliminaries

3.1 Bilinear Groups

Our scheme utilizes bilinear groups and bilinear maps as follows:

1. $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{T}$ are multiplicative cyclic groups of prime order p,
2. g and h are randomly chosen generators of $\mathcal{G}$ and $\mathcal{H}$, respectively.
3. e is an efficiently computable bilinear map: $\mathcal{G} \times \mathcal{H} \rightarrow \mathcal{T}$, i.e., (1) for all $u, u' \in \mathcal{G}$ and $v, v' \in \mathcal{H}$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$, and thus for all $u \in \mathcal{G}$, $v \in \mathcal{H}$ and $a, b \in Z$, $e(u^a, v^b) = e(u, v)^{ab}$, and (2) $e(g, h) \neq 1$.

We can set $\mathcal{G} = \mathcal{H}$, but we allow $\mathcal{G} \neq \mathcal{H}$ for the generality. Thus, our scheme can be implemented on both supersingular curves and ordinary curves. The bilinear map can be efficiently implemented with the Tate pairing (or the η_T pairing [3] on supersingular curves or the Ate pairing [19] on ordinary curves).

3.2 Assumptions

Our schemes are based on the q-SDH assumption [7,8] on $\mathcal{G}$. Furthermore, our schemes adopt the tracing mechanism of [16], where, in addition to the bilinear groups, another group $\mathcal{F}$ with the same order p and the DDH assumption is required.

Definition 1 (q-SDH assumption). *For all PPT algorithm $\mathcal{A}$, the probability*

$$\Pr[\mathcal{A}(u, u^a, \ldots, u^{(a^q)}) = (b, u^{(1/a+b)}) \wedge b \in Z_p]$$

is negligible, where $u \in_R \mathcal{G}$ and $a \in_R Z_p$.

Definition 2 (DDH assumption). *For all PPT algorithm $\mathcal{A}$, the probability*

$$|\Pr[\mathcal{A}(u, u^a, u^b, u^{ab}) = 1] - \Pr[\mathcal{A}(u, u^a, u^b, u^c) = 1]|$$

is negligible, where $u \in_R \mathcal{F}$ and $a, b, c \in_R Z_p$.

Based on the q-SDH assumption, the DL (Discrete Logarithm) assumption also holds.

Definition 3 (DL assumption). *For all PPT algorithm $\mathcal{A}$, the probability*

$$\Pr[\mathcal{A}(u, u^a) = a]$$

is negligible, where $u \in_R \mathcal{G}$ and $a \in_R Z_p$.

3.3 BB Signatures

Our group signature schemes utilize Boneh-Boyen (BB) signature scheme [6]. As shown in [7], the knowledge of this signature (and the message) can be proved by a zero-knowledge proof for a representation, which is shown in Sec. 3.5.

BB-Setup: Select bilinear groups $\mathcal{G}, \mathcal{H}, \mathcal{T}$ with a prime order p and a bilinear map e. Select $g \in_R \mathcal{G}$ and $h \in_R \mathcal{H}$.

BB-KeyGen: Select $X \in_R Z_p$ and compute $Y = h^X$. The secret key is X and the public key is $(p, \mathcal{G}, \mathcal{H}, \mathcal{T}, e, g, h, Y)$.

BB-Sign: Given message $m \in Z_p$, compute $A = g^{1/(X+m)}$. The signature is A.

BB-Verify: Given message m and the signature A, check $e(A, Yh^m) = e(g, h)$.

BB signatures are existentially unforgeable against *weak* chosen message attack under the q-SDH assumption [6]. In this attack, the adversary must choose messages queried for the oracle, before the public key is given.

3.4 BBS+ Signatures

This signature scheme is an extension from BB signature scheme, and is informally introduced in [7], and the concrete construction is shown in [16,1]. We call this signature BBS+ signature, as well as [1]. This scheme allows us to sign a set of messages. Also, the knowledge of the signature and messages can be proved [16,1], as well as BB signatures.

BBS+-Setup: Select bilinear groups $\mathcal{G}, \mathcal{H}, \mathcal{T}$ with a prime order p and a bilinear map e. Select $g, g_1, \ldots, g_{L+1} \in_R \mathcal{G}$ and $h \in_R \mathcal{H}$.

BBS+-KeyGen: Select $X \in_R Z_p$ and compute $Y = h^X$. The secret key is X and the public key is $(p, \mathcal{G}, \mathcal{H}, \mathcal{T}, e, g, g_1, \ldots, g_{L+1}, h, Y)$.

BBS+-Sign: Given messages $m_1, \ldots, m_L \in Z_p$, select $y, z \in_R Z_p$ and compute

$$A = (g_1^{m_1} \cdots g_L^{m_L} g_{L+1}^{y} g)^{1/(X+z)}.$$

The signature is (A, y, z).

BBS+-Verify: Given messages $m_1, \ldots, m_L$ and the signature (A, y, z), check

$$e(A, Yh^z) = e(g_1^{m_1} \cdots g_L^{m_L} g_{L+1}^{y} g, h).$$

BBS+ signatures are existentially unforgeable against adaptively chosen message attack under the q-SDH assumption [2].

3.5 Proving Relations on Representations

As well as [7,8,16], we adopt signatures converted by Fiat-Shamir heuristic (using a hash function) from zero-knowledge proofs of knowledge (PK), where a signer can convince a verifier of knowledge with relations on representations. We call the signatures SPKs. The SPKs we adopt are the generalization of the Schnorr signature. We introduce the following notation.

$$SPK\{(x_1, \ldots, x_t) : R(x_1, \ldots, x_t)\}(M),$$

which means a signature of message M by a signer who knows secret values $x_1, \ldots, x_t$ satisfying a relation $R(x_1, \ldots, x_t)$. In this paper, the following SPKs on $\mathcal{G}, \mathcal{T}, \mathcal{F}$ are utilized.

SPK **of representation:** An SPK proving the knowledge of a representation of $C \in \mathcal{G}$ to the bases $g_1, g_2, \ldots, g_t \in \mathcal{G}$ on message M is denoted as

$$SPK\{(x_1, \ldots, x_t) : C = g_1^{x_1} \cdots g_t^{x_t}\}(M).$$

This can be also constructed on groups $\mathcal{T}, \mathcal{F}$.

SPK **of representations with equal parts:** An SPK proving the knowledge of representations of $C, C' \in \mathcal{G}$ to the bases $g_1, \ldots, g_t \in \mathcal{G}$ on message M, where the representations include equal values as parts, is denoted as

$$SPK\{(x_1, \ldots, x_u) : C = g_{i_1}^{x_{j_1}} \cdots g_{i_v}^{x_{j_v}} \wedge C' = g_{i'_1}^{x_{j'_1}} \cdots g_{i'_{v'}}^{x_{j'_{v'}}}\}(M),$$

where indices $i_1, \ldots i_v, i'_1, \ldots i'_{v'} \in \{1, \ldots, t\}$ refer to the bases $g_1, \ldots, g_t$, and indices $j_1, \ldots j_v, j'_1, \ldots, j'_{v'} \in \{1, \ldots, u\}$ refer to the secrets $x_1, \ldots, x_u$. This SPK can be extended for different groups $\mathcal{G}, \mathcal{T}$ and $\mathcal{F}$ with the same order p, such as

$$SPK\{(x_1, \ldots, x_u) : C = g_{i_1}^{x_{j_1}} \cdots g_{i_v}^{x_{j_v}} \wedge C' = h_{i'_1}^{x_{j'_1}} \cdots h_{i'_{v'}}^{x_{j'_{v'}}}\}(M),$$

where $C, g_1, \ldots, g_t \in \mathcal{G}$ and $C', h_1, \ldots, h_t \in \mathcal{T}$.

In the random oracle model, the SPK can be simulated without the knowledge using a simulator in the zero-knowledge-ness of the underlying PK. Moreover, the SPK has an extractor of the proved secret knowledge given two accepting protocol views whose commitments are the same and whose challenges are different.

4 Proposed Scheme

4.1 Idea

The mechanism of conventional (non-revocable) group signature schemes is informally as follows. When a member joins, the member sends $f(x)$ to GM, where f is a one-way function and x is a secret. GM returns a membership certificate $S = Sign(x)$ to the member, where $Sign$ is a signing function of GM. Then, the group signature consists of $E = Enc(f(x))$, where Enc is an encryption function using the manager's public key, and the following SPK on the signed message M.

$$SPK\{(x, S) : S = Sign(x) \wedge E = Enc(f(x))\}(M).$$

When opening the group signature, the manager decrypts E to check the sender of $f(x)$ in joining.

We borrow this mechanism from the Furukawa-Imai scheme [16], which is the one improved on the efficiency from [7] and currently is the most efficient pairing-based scheme. To this component, we add a novel revocation mechanism for realizing the constant computational complexity in signing/verifying.

The membership certificate in our scheme is modified to $S = Sign(x, UID)$, where UID is the ID of the member. On the other hand, a revocation list $\boldsymbol{RL}_t$ consists of

$$(Sign(t, RID_0, RID_1), \ldots, Sign(t, RID_r, RID_{r+1})).$$

RID_i $(1 \le i \le r)$ is the UID of a revoked member. Then, we can assume that RID_i is sorted such that $RID_i < RID_{i+1}$ for all $1 \le i \le r$. In addition, RID_0 and RID_{r+1} are special IDs, where $RID_0 < UID$ and $RID_{r+1} > UID$ for any UID.

The group signature for $\boldsymbol{RL}_t$ is the following SPK.

$$\begin{aligned} SPK\{&(x, UID, S, RID_i, RID_{i+1}, \tilde{S}) : \\ &S = Sign(x, UID) \wedge E = Enc(f(x)) \\ &\wedge \tilde{S} = Sign(t, RID_i, RID_{i+1}) \\ &\wedge RID_i < UID \wedge UID < RID_{i+1}\}(M). \end{aligned}$$

Clearly any non-revoked member can prove this SPK. On the other hand, if the member with UID is revoked, $UID = RID_{\tilde{\imath}}$ holds for some $\tilde{\imath}$. Then, $RID_i < UID$ holds for all $i < \tilde{\imath}$, and $UID < RID_i$ for all $i > \tilde{\imath}$. Thus, the revoked member cannot find i such that $RID_i < UID < RID_{i+1}$, which means that the member cannot prove this SPK, since the correctness of UID, RID_i and RID_{i+1} are also ensured by the certificates S and $\tilde{S}$ at the current time t.

The costs of the SPKs for inequations have $O(1)$ complexity, as the construction idea of the SPK is shown later. Thus, the computational costs for signing/verifying are $O(1)$ w.r.t. R and N. The size of a signature is also $O(1)$. The size of $\boldsymbol{RL}_t$ is $O(R)$. The overhead is the revocation complexity of GM, that is, each revocation requires $O(R)$ computation, and the long public key with $O(N)$ size due to the following SPK for inequations.

Remark 1. Our novel idea is to use the above integer inequations on UID and RID's. Instead, we easily get a simple solution of proving that his UID is not equal to **all** RID's, but it requires $O(R)$ complexity. Camenisch and Lysyanskaya proposed an elegant idea of the dynamic accumulator [14], but it needs the incremental update of signer's secret key. On the other hand, in our solution, the signer has only to prove the inequations for **some** i, and thus has $O(1)$ complexity. And, note that it does not need any update of the secret key. As far as we know, such a solution is unknown, and thus our construction idea has a sufficient novelty together with the practicality on efficiency.

Remark 2. Note that this group signature does not reveal any information on i, RID_i and RID_{i+1} to the verifier. This is because the SPK proves only the fact that there are secrets i, RID_i and RID_{i+1} satisfying the inequations $RID_i < UID < RID_{i+1}$, without revealing i, RID_i and RID_{i+1}.

Proving Integer Inequations. In the above construction idea, we need an efficient SPK proving an integer inequation on secrets, as $RID_i < UID$. In the strong RSA setting, Boudot's SPK [9] can be used. However, this methodology cannot be easily adopted to the pairing-based setting. On the other hand, Teranishi and Sako utilize an SPK [25] proving that a secret is in an integer interval, and the SPK is effective even in the pairing-based setting. We adopt this SPK and extend it to the SPK proving the integer inequation.

At first, we consider the SPK proving that a secret w is in the interval $[1, N]$. This SPK needs a special setup, where the trusted party (GM in our setting) issues certificates. The certificate is a BB signature on every element from the interval $[1, N]$, as $Sign(1), \ldots, Sign(N)$. These certificates are given to each prover, as a part of the public key. Then, the SPK proving $w \in [1, N]$ is computed as

$$SPK\{(w, S') : S' = Sign(w)\}(M).$$

Since issued certificates are for only $1, \ldots, N$, it ensures $w \in [1, N]$.

Next, using this SPK, we consider the SPK proving $y > x$, in the situation that $x, y \in [1, N]$ is ensured (Note that it is ensured in the above group signature, due to $S = Sign(x, UID) \wedge \tilde{S} = Sign(t, RID_i, RID_{i+1})$). The SPK is obtained as

$$SPK\{(x, y, S') : S' = Sign(y - x \bmod p)\}(M).$$

Then, this ensures $z \in [1, N]$, where $z = y - x \pmod p$. On the other hand, from $x, y \in [1, N]$, we obtain $z \in [1, N]$ when $y > x$, and $z \in [p - N, p - 1]$ or $z = 0$ when $y \leq x$. Since we can assume $N < p/2$, $z \in [1, N]$ means $y > x$.

The computational cost of this SPK is constant w.r.t. N, although the distributions of N certificates is an overhead.

Remark 3. As another approach of the SPK proving the interval relation, a bit-by-bit approach is also known, which is described and used in [1]. However, in this approach, the size of the proof is linear in the size of the secret to be proved. In case of adopting it in our group signatures, the size is $\lceil \log_2 N \rceil$, and the proof size is about $4 \lceil \log_2 N \rceil$ multiplied by the size of the group element. If we use the same parameters as Sect. 7 and $N = 1,000,000$, the SPK amounts to more than 2,000 Bytes. Since we use interval proofs two times in our group signatures, the group signature is about more than 4,000 Bytes. Even in case of $N = 1,000$, it is more than 2,000 Bytes. On the other hand, our group signature proposed in this section is about 650 Bytes, and our extended group signature proposed in Sect. 6 is about 1,200 Bytes.

In the bit-by-bit approach, the public key size is constant and better. However, signatures frequently occur in authentications or signings, and thus, in lots of applications, the signature size influences the efficiency of communication and storage more than the size of the public key that is not changed. This is why we adopt Teranishi-Sako's SPK proving interval relations in this paper.

4.2 Proposed Algorithms

Assume that the total number of group members, N, is fixed in advance, and we can assume that $N < p/2$.

Setup: The input of this algorithm is security parameter 1^ℓ, and the output is *param*.

1. Select bilinear groups $\mathcal{G}, \mathcal{H}, \mathcal{T}$ with the same prime order p of length ℓ, and the bilinear map e. In addition, select a group $\mathcal{F}$ with the DDH assumption and the same prime order p. Select hash function $H : \{0,1\}^* \rightarrow Z_p$.
2. Select $g, g_1, g_2, g_3, \tilde{g}, \tilde{g}_1, \hat{g}, \hat{g}_1, \hat{g}_2, \hat{g}_3, \hat{g}_4 \in_R \mathcal{G}$, $h, \tilde{h}, \hat{h} \in_R \mathcal{H}$, and $f \in_R \mathcal{F}$.
3. Output $param = (p, \mathcal{G}, \mathcal{H}, \mathcal{T}, \mathcal{F}, e, H, g, g_1, g_2, g_3, \tilde{g}, \tilde{g}_1, \hat{g}, \hat{g}_1, \hat{g}_2, \hat{g}_3, \hat{g}_4, h, \tilde{h}, \hat{h}, f)$.

KeyGen: The inputs of this algorithm are N and $param = (p, \mathcal{G}, \mathcal{H}, \mathcal{T}, \mathcal{F}, e, H, g, g_1, g_2, g_3, \tilde{g}, \tilde{g}_1, \hat{g}, \hat{g}_1, \hat{g}_2, \hat{g}_3, \hat{g}_4, h, \tilde{h}, \hat{h}, f)$, and the output consists of *gpk* and *msk*.

1. Select $X, \tilde{X}, \hat{X} \in_R Z_p$ and compute $Y = h^X$, $\tilde{Y} = \tilde{h}^{\tilde{X}}$, and $\hat{Y} = \hat{h}^{\hat{X}}$.
2. Select $X_1, X_2 \in_R Z_p$ and compute $Y_1 = f^{X_1}$ and $Y_2 = f^{X_2}$.
3. For all $j \in 1, \ldots, N$, generate BB signatures for j, namely compute $F_j = \tilde{g}^{1/(\tilde{X}+j)}$.
4. Output $gpk = (p, \mathcal{G}, \mathcal{H}, \mathcal{T}, \mathcal{F}, e, H, g, g_1, g_2, g_3, \tilde{g}, \tilde{g}_1, \hat{g}, \hat{g}_1, \hat{g}_2, \hat{g}_3, \hat{g}_4, h, \tilde{h}, \hat{h}, f, Y, \tilde{Y}, \hat{Y}, Y_1, Y_2, F_1, \ldots F_N)$ and $msk = (X, \tilde{X}, \hat{X}, X_1, X_2)$.

Join: This is an interactive protocol between **Join-U** (the i-th joining user) and **Join-GM** (GM)**.** The common input is *gpk*, and the input of **Join-GM** is *msk*. The output of **Join-U** is $\boldsymbol{usk}[i]$. The output is **Join-GM** is $\boldsymbol{reg}[i]$. Assume that $i = 1$ and $i = N$ are assigned to fictitious users out of the group (Assume that the users of $i = 1, N$ are always revoked).

1. [**Join-U**] Select $x_i, y'_i \in Z_p$, compute $A'_i = g_1^{x_i} g_3^{y'_i}$ and $D_i = f^{x_i}$, and send A'_i, D_i to **Join-GM**. In addition, prove the validity of A'_i and D_i using an SPK for representations.

2. [**Join-GM**] Select $y''_i, z_i \in_R Z_p$, compute
$$A_i = (A'_i g_2^i g_3^{y''_i} g)^{1/(X+z_i)},$$
and return (i, A_i, y''_i, z_i) to **Join-U**. Output $\boldsymbol{reg}[i] = D_i$.
3. [**Join-U**] Compute $y_i = y'_i + y''_i \bmod p$, verify $e(A_i, Yh^{z_i}) = e(g_1^{x_i} g_2^i g_3^{y_i} g, h)$, and output $\boldsymbol{usk}[i] = (A_i, x_i, y_i, z_i)$ s.t. $A_i^{X+z_i} = g_1^{x_i} g_2^i g_3^{y_i} g$. The BBS+ signatures (A_i, y_i, z_i) on secret x_i and UID i is correspondent to the membership certificate.

Revoke: The input of this algorithm consists of gpk, t and RU. The output is $\boldsymbol{RL}_t$.

1. Sort elements of RU, according to ascending order. Let $\hat{\imath}_1, \ldots, \hat{\imath}_r$ be the sorted ones, where $r = |RU|$. In addition, set $\hat{\imath}_0 = 1$ and $\hat{\imath}_{r+1} = N$.
2. For every $\hat{\imath}_j$ $(0 \leq j \leq r)$, generate a BBS+ signature on $t, \hat{\imath}_j, \hat{\imath}_{j+1}$, namely select $\hat{y}_j, \hat{z}_j \in_R Z_p$, and compute $B_{\hat{\imath}_j} = (\hat{g}_1^t \hat{g}_2^{\hat{\imath}_j} \hat{g}_3^{\hat{\imath}_{j+1}} \hat{g}_4^{\hat{y}_j} \hat{g})^{1/(\hat{X}+\hat{z}_j)}$.
3. Output
$$\boldsymbol{RL}_t = ((\hat{\imath}_0, \hat{\imath}_1, B_{\hat{\imath}_0}, \hat{y}_0, \hat{z}_0), \ldots, (\hat{\imath}_r, \hat{\imath}_{r+1}, B_{\hat{\imath}_r}, \hat{y}_r, \hat{z}_r)).$$

Sign: The input of this algorithm consists of gpk, $\boldsymbol{usk}[i] = (A_i, x_i, y_i, z_i)$, t, $\boldsymbol{RL}_t = ((\hat{\imath}_0, \hat{\imath}_1, B_{\hat{\imath}_0}, \hat{y}_0, \hat{z}_0), \ldots, (\hat{\imath}_r, \hat{\imath}_{r+1}, B_{\hat{\imath}_r}, \hat{y}_r, \hat{z}_r))$ and $M \in \{0,1\}^*$. The output is σ.

1. Select a random $\alpha \in_R Z_p$, and compute a commitment $C = A_i g_3^{\alpha}$. Set $\zeta = y_i + \alpha z_i$.
2. Find $\hat{\imath}_j$ s.t. $\hat{\imath}_j < i < \hat{\imath}_{j+1}$. Select a random $\hat{\alpha} \in_R Z_p$, and compute a commitment $\hat{C} = B_{\hat{\imath}_j} \hat{g}_4^{\hat{\alpha}}$. Set $\hat{\zeta} = \hat{y}_j + \hat{\alpha}\hat{z}_j$.
3. Set $\delta_1 = i - \hat{\imath}_j$ and $\delta_2 = \hat{\imath}_{j+1} - i$. Find F_{δ_1} and F_{δ_2}, select $\beta_1, \beta_2 \in_R Z_p$, and compute commitments $C_{F_{\delta_1}} = F_{\delta_1} \tilde{g}_1^{\beta_1}$ and $C_{F_{\delta_2}} = F_{\delta_2} \tilde{g}_1^{\beta_2}$. Set $\theta_1 = \beta_1 \delta_1$, and $\theta_2 = \beta_2 \delta_2$.
4. Select a random $\gamma \in_R Z_p$, and compute ciphertext $T_1 = f^{x_i+\gamma}$, $T_2 = Y_1^{\gamma}$, and $T_3 = Y_2^{\gamma}$.
5. Compute an SPK V on message M proving knowledge of $x_i, i, \zeta, \alpha, z_i, \hat{\imath}_j$, $\hat{\imath}_{j+1}$, $\hat{\zeta}, \hat{\alpha}, \hat{z}_j, \theta_1, \beta_1, \theta_2, \beta_2, \gamma$ s.t.
$$\begin{aligned}
&e(C, Y)e(g, h)^{-1} = e(g_1, h)^{x_i} e(g_2, h)^i e(g_3, h)^{\zeta} e(g_3, Y)^{\alpha} e(C, h)^{-z_i},\\
&e(\hat{C}, \hat{Y})e(\hat{g}, \hat{h})^{-1} e(\hat{g}_1, \hat{h})^{-t}\\
&\qquad\qquad = e(\hat{g}_2, \hat{h})^{\hat{\imath}_j} e(\hat{g}_3, \hat{h})^{\hat{\imath}_{j+1}} e(\hat{g}_4, \hat{h})^{\hat{\zeta}} e(\hat{g}_4, \hat{Y})^{\hat{\alpha}} e(\hat{C}, \hat{h})^{-\hat{z}_j},\\
&e(C_{F_{\delta_1}}, \tilde{Y})e(\tilde{g}, \tilde{h})^{-1} = e(\tilde{g}_1, \tilde{h})^{\theta_1} e(\tilde{g}_1, \tilde{Y})^{\beta_1} e(C_{F_{\delta_1}}, \tilde{h})^{-(i-\hat{\imath}_j)},\\
&e(C_{F_{\delta_2}}, \tilde{Y})e(\tilde{g}, \tilde{h})^{-1} = e(\tilde{g}_1, \tilde{h})^{\theta_2} e(\tilde{g}_1, \tilde{Y})^{\beta_2} e(C_{F_{\delta_2}}, \tilde{h})^{-(\hat{\imath}_{j+1}-i)},\\
&T_1 = f^{x_i+\gamma}, \quad T_2 = Y_1^{\gamma}, \quad T_3 = Y_2^{\gamma}.
\end{aligned}$$
As indicated in Lemma 1, the above relations in V ensure the validity of signatures (A_i, y_i, z_i), $(B_{\hat{\imath}_j}, \hat{y}_j, \hat{z}_j)$, F_{δ_1}, F_{δ_2} and the validity of T_1, T_2 and T_3, respectively.
6. Output $\sigma = (C, \hat{C}, C_{F_{\delta_1}}, C_{F_{\delta_2}}, T_1, T_2, T_3, V)$.

Verify: The inputs are gpk, t, a target signature $\sigma = (C, \hat{C}, C_{F_{\delta_1}}, C_{F_{\delta_2}}, T_1, T_2,$ T_3, $V)$, and the message M. Check the SPK V. Output 'valid' (resp., 'invalid') if it is correct (resp., incorrect).

Open : The inputs are gpk, the secret key $msk = (X, \tilde{X}, \hat{X}, X_1, X_2)$, $\boldsymbol{reg}$ with $\boldsymbol{reg}[i] = D_i$, t, a target signature $\sigma = (C, \hat{C}, C_{F_{\delta_1}}, C_{F_{\delta_2}}, T_1, T_2, T_3, V)$ and the message M.

1. Verify σ. If it is invalid, abort.
2. Using X_1, compute $T_1/T_2^{1/X_1}$ to obtain f^{x_i}. Search $\boldsymbol{reg}$ for i with $D_i = f^{x_i}$.
3. Output i.

5 Security

Here, we show a lemma and theorems on the security of our scheme.

Lemma 1. *The SPK V proves the knowledge of* x_i, i, z_i, $y_i, \hat{\imath}_j, \hat{\imath}_{j+1}, \hat{z}_j, \hat{y}_j, \eta_1$, η_2, $\gamma, A_i, B_{\hat{\imath}_j}, F'_{\delta_1}, F'_{\delta_2}$ *s.t.*

$$A_i = (g_1^{x_i} g_2^{i} g_3^{y_i} g)^{1/(X+z_i)}, \qquad B_{\hat{\imath}_j} = (\hat{g}_1^{t} \hat{g}_2^{\hat{\imath}_j} \hat{g}_3^{\hat{\imath}_{j+1}} \hat{g}_4^{\hat{y}_j} \hat{g})^{1/(\hat{X}+\hat{z}_j)},$$

$$F'_{\delta_1} = (\tilde{g}_1^{\eta_1} \tilde{g})^{1/(\tilde{X}+(i-\hat{\imath}_j))}, \qquad F'_{\delta_2} = (\tilde{g}_1^{\eta_2} \tilde{g})^{1/(\tilde{X}+(\hat{\imath}_{j+1}-i))},$$

$$T_1 = f^{x_i+\gamma}, \qquad T_2 = Y_1^{\gamma}, \qquad T_3 = Y_2^{\gamma}.$$

This proof will be in the full paper. Note that F'_{δ_1} and F'_{δ_2} are variants of BB signatures, and are not the same as F_{δ_1} and F_{δ_2}, due to the parts $\tilde{g}_1^{\eta_1}, \tilde{g}_1^{\eta_2}$. However, in the traceability game, the difference can be treated well. Note that, as well as [21], the adopted SPKs of F'_{δ_1} and F'_{δ_2} are more efficient than those of F_{δ_1} and F_{δ_2} described in [7].

Theorem 1. *The proposed scheme satisfies the traceability under the q-SDH assumption, in the random oracle model.*

This proof will be in the full paper. In the proof, if an adversary wins the traceability game for our scheme, using this adversary, we can forge BBS+ signatures (A_i, y_i, z_i) or $(B_{\hat{\imath}_j}, \hat{y}_j, \hat{z}_j)$, or a BB signature F_k. Thus, we can construct adversaries for the BBS+ signatures or BB+ signatures, which are secure under the q-SDH assumption.

Theorem 2. *The proposed scheme satisfies the anonymity under the DDH assumption, in the random oracle model.*

This proof is similar to [17], which is the full-paper version of [16]. As well as [16], our group signature consists of a double encryption of an ID f^{x_i}, (T_1, T_2, T_3), and the non-interactive zero-knowledge proof including statistically hiding commitments, which mean an IND-CCA2 secure encryption. Thus, easily we can reduce the anonymity game of our scheme to the IND-CCA2 game for the IND-CCA2 secure encryption under the DDH assumption.

Theorem 3. *The proposed scheme satisfies the non-frameability under the DL assumption, in the random oracle model.*

This proof is also similar to [17]. In this proof, the DL adversary is constructed using the adversary in the non-frameability game. In the game, instead of computing (f, f^{x_i}), the input of the DL adversary is used as (f, f^{x_i}). In an honest user's joining, f^{x_i} in the input (x_i is unknown) is used. The SPKs for x_i in joining and signing can be simulated by the zero-knowledge simulator. From the output of the adversary in the non-frameability game (which outputs the signature for the honest user with x_i with a non-negligible probability), we can extract x_i, which means breaking the DL assumption.

6 Extension

The weak point of the proposed scheme is $O(N)$ size of gpk. This section shows an extended scheme with $O(\sqrt{N})$-size public key.

6.1 Idea

The extension is obtained by improving the SPK proving integer inequation. A positive integer $w \in [1, N]$ can be expressed as $w_1^2 + w_2$, where w_1 is the greatest square less than w, and w_2 is a non-negative integer less than $2\sqrt{N}$ [9]. Then, note that $1 \le w_1 < \sqrt{N}$ and $0 \le w_2 < 2\sqrt{N}$. The extended integer inequation proof is as follows. Define $N_1 = \lfloor \sqrt{N} \rfloor$ and $N_2 = \lfloor 2\sqrt{N} \rfloor$.

In the setup, the trusted party issues two types of certificates based on BB signatures. In one type, $Sign(1), \ldots, Sign(N_1)$ are issued. In the other, $Sign'(0), \ldots, Sign'(N_2)$ are issued.

Then, the SPK proving $y > x$ is computed as

$$SPK\{(x, y, w_1, w_2, S_1, S_2) : y - x = w_1^2 + w_2 \pmod{p} \\ \wedge S_1 = Sign(w_1) \wedge S_2 = Sign'(w_2)\}(M).$$

The relation $y - x = w_1^2 + w_2 \pmod{p}$ can be efficiently proved by an SPK, as [9]. Due to the BB signatures, $w_1 \in [1, N_1]$ and $w_2 \in [0, N_2]$ are ensured. Then, $w_1^2 + w_2 \in [1, N_1^2 + N_2]$. By assuming $N_1^2 + N_2 < p/2$, we obtain $w_1^2 + w \in [1, p/2 - 1]$. Namely, for $z = y - x \pmod{p}$, we have $z \in [1, p/2 - 1]$. Since $x, y \in [1, N]$ is ensured, as well as the basic scheme, this means $y - x > 0$ and thus $y > x$ in Z.

On the other hand, the number of the issued certificates is $O(\sqrt{N})$, which means $O(\sqrt{N})$-size public key.

6.2 Extended Algorithms

Define $N_1 = \lfloor \sqrt{N} \rfloor$ and $N_2 = \lfloor 2\sqrt{N} \rfloor$. We assume that $N_1^2 + N_2 < p/2$. **Revoke**, **Verify**, and **Open** are similar to the basic scheme in Sec. 4. The others are modified as follows.

Setup: In addition to **Setup** of the basic scheme, select $\dot{g}, \dot{g}_1 \in \mathcal{G}, \dot{h} \in \mathcal{H}$, which are added to *param*.

KeyGen: In addition to Step 1 and 2 of **KeyGen** of the basic scheme, select $\dot{X} \in_R Z_p$, and compute $\dot{Y} = \dot{h}^{\dot{X}}$. Step 3 is modified as follows.

3. For all $j \in 1, \ldots, N_1$, generate BB signatures for j, namely compute $F_j = \tilde{g}^{1/(\tilde{X}+j)}$. Additionally, for all $j \in 0, \ldots, N_2$, generate BB signatures for j, namely compute $\dot{F}_j = \dot{g}^{1/(\dot{X}+j)}$.

Add $\dot{X}$ to *msk*, and add $\dot{Y}$, $F_1, \ldots, F_{N_1}$, $\dot{F}_0, \ldots, \dot{F}_{N_2}$ to *gpk*.

Sign: Step 1, 2, 4 are the same as the basic scheme. Step 3, 5 are modified as follows.

3. Set $\delta_1 = i - \hat{\imath}_j$ and $\delta_2 = \hat{\imath}_{j+1} - i$. Find $\delta_{1,1}$ and $\delta_{1,2}$ s.t. $\delta_1 = {\delta_{1,1}}^2 + \delta_{1,2}$, $1 \leq \delta_{1,1} \leq N_1$ and $0 \leq \delta_{1,2} \leq N_2$. Find $\delta_{2,1}$ and $\delta_{2,2}$ s.t. $\delta_2 = {\delta_{2,1}}^2 + \delta_{2,2}$, $1 \leq \delta_{2,1} \leq N_1$ and $0 \leq \delta_{2,2} \leq N_2$. Find $F_{\delta_{1,1}}$, $\dot{F}_{\delta_{1,2}}$, $F_{\delta_{2,1}}$, and $\dot{F}_{\delta_{2,2}}$, select $\beta_{1,1}, \beta_{1,2}, \beta_{2,1}, \beta_{2,2} \in_R Z_p$, and compute commitments $C_{F_{\delta_{1,1}}} = F_{\delta_{1,1}} \tilde{g}_1^{\beta_{1,1}}$, $C_{\dot{F}_{\delta_{1,2}}} = \dot{F}_{\delta_{1,2}} \dot{g}_1^{\beta_{1,2}}$, $C_{F_{\delta_{2,1}}} = F_{\delta_{2,1}} \tilde{g}_1^{\beta_{2,1}}$, $C_{\dot{F}_{\delta_{2,2}}} = \dot{F}_{\delta_{2,2}} \dot{g}_1^{\beta_{2,2}}$. Set $\theta_{1,1} = \beta_{1,1}\delta_{1,1}$, $\theta_{1,2} = \beta_{1,2}\delta_{1,2}$, $\theta_{2,1} = \beta_{2,1}\delta_{2,1}$, and $\theta_{2,2} = \beta_{2,2}\delta_{2,2}$.
   Furthermore, select $\xi_1, \xi_1', \xi_2, \xi_2' \in_R Z_p$, and compute commitments $C_{\delta_{1,1}} = \tilde{g}^{\delta_{1,1}} \tilde{g_1}^{\xi_1}$, $C_{{\delta_{1,1}}^2} = \tilde{g}^{{\delta_{1,1}}^2} \tilde{g_1}^{\xi_1'}$, $C_{\delta_{2,1}} = \tilde{g}^{\delta_{2,1}} \tilde{g_1}^{\xi_2}$, $C_{{\delta_{2,1}}^2} = \tilde{g}^{{\delta_{2,1}}^2} \tilde{g_1}^{\xi_2'}$. Set $\xi_1'' = \xi_1' - \xi_1\delta_{1,1}$ and $\xi_2'' = \xi_2' - \xi_2\delta_{2,1}$.
5. Compute an *SPK* V on message M proving knowledge of $x_i, i, \zeta, \alpha, z_i, \hat{\imath}_j$, $\hat{\imath}_{j+1}$, $\hat{\zeta}, \hat{\alpha}, \hat{z}_j, \delta_{1,1}, \delta_{1,2}, \delta_{2,1}$, $\delta_{2,2}, \theta_{1,1}, \theta_{1,2}, \theta_{2,1}, \theta_{2,2}, \beta_{1,1}, \beta_{1,2}, \beta_{2,1}, \beta_{2,2}, \xi_1, \xi_1'$, ξ_1'', ξ_2, ξ_2', ξ_2'', γ s.t.

$$\begin{aligned}
&e(C, Y)e(g, h)^{-1} = e(g_1, h)^{x_i} e(g_2, h)^{i} e(g_3, h)^{\zeta} e(g_3, Y)^{\alpha} e(C, h)^{-z_i}, \\
&e(\hat{C}, \hat{Y})e(\hat{g}, \hat{h})^{-1} e(\hat{g}_1, \hat{h})^{-t} \\
&\qquad\qquad = e(\hat{g}_2, \hat{h})^{\hat{\imath}_j} e(\hat{g}_3, \hat{h})^{\hat{\imath}_{j+1}} e(\hat{g}_4, \hat{h})^{\hat{\zeta}} e(\hat{g}_4, \hat{Y})^{\hat{\alpha}} e(\hat{C}, \hat{h})^{-\hat{z}_j}, \\
&e(C_{F_{\delta_{1,1}}}, \tilde{Y})e(\tilde{g}, \tilde{h})^{-1} = e(\tilde{g}_1, \tilde{h})^{\theta_{1,1}} e(\tilde{g}_1, \tilde{Y})^{\beta_{1,1}} e(C_{F_{\delta_{1,1}}}, \tilde{h})^{-\delta_{1,1}}, \\
&e(C_{\dot{F}_{\delta_{1,2}}}, \dot{Y})e(\dot{g}, \dot{h})^{-1} = e(\dot{g}_1, \dot{h})^{\theta_{1,2}} e(\dot{g}_1, \dot{Y})^{\beta_{1,2}} e(C_{\dot{F}_{\delta_{1,2}}}, \dot{h})^{-\delta_{1,2}}, \\
&e(C_{F_{\delta_{2,1}}}, \tilde{Y})e(\tilde{g}, \tilde{h})^{-1} = e(\tilde{g}_1, \tilde{h})^{\theta_{2,1}} e(\tilde{g}_1, \tilde{Y})^{\beta_{2,1}} e(C_{F_{\delta_{2,1}}}, \tilde{h})^{-\delta_{2,1}}, \\
&e(C_{\dot{F}_{\delta_{2,2}}}, \dot{Y})e(\dot{g}, \dot{h})^{-1} = e(\dot{g}_1, \dot{h})^{\theta_{2,2}} e(\dot{g}_1, \dot{Y})^{\beta_{2,2}} e(C_{\dot{F}_{\delta_{2,2}}}, \dot{h})^{-\delta_{2,2}}, \\
&C_{\delta_{1,1}} = \tilde{g}^{\delta_{1,1}} \tilde{g_1}^{\xi_1}, \quad C_{{\delta_{1,1}}^2} = C_{\delta_{1,1}}^{\delta_{1,1}} \tilde{g_1}^{\xi_1''}, \quad C_{{\delta_{1,1}}^2} = \tilde{g}^{-\delta_{1,2}+(i-\hat{\imath}_j)} \tilde{g_1}^{\xi_1'}, \\
&C_{\delta_{2,1}} = \tilde{g}^{\delta_{2,1}} \tilde{g_1}^{\xi_2}, \quad C_{{\delta_{2,1}}^2} = C_{\delta_{2,1}}^{\delta_{2,1}} \tilde{g_1}^{\xi_2''}, \quad C_{{\delta_{2,1}}^2} = \tilde{g}^{-\delta_{2,2}+(\hat{\imath}_{j+1}-i)} \tilde{g_1}^{\xi_2'}, \\
&T_1 = f^{x_i+\gamma}, \quad T_2 = Y_1^{\gamma}, \quad T_3 = Y_2^{\gamma}.
\end{aligned}$$

Security. The proof sketch of the traceability will also be in the full paper, which is similar to the proof of the basic scheme. The anonymity and non-frameability can be proved as well as the basic scheme.

7 Efficiency

Computational efficiency. At first, we discuss the efficiency of the basic scheme in Sec. 4. As well as [22], all pairings in the **Sign** algorithm can be pre-computed, and pairings except 4 pairing in the **Verify** algorithm can be also pre-computed. Then, the **Sign** algorithm requires 4 exponentiations on $\mathcal{G}$, 6 exponentiations on $\mathcal{F}$, and 4 multi-exponentiations on $\mathcal{T}$. The **Verify** algorithm requires 3 exponentiations on $\mathcal{F}$, 4 multi-exponentiations on $\mathcal{H}$, 4 multi-exponentiations on $\mathcal{T}$, and 4 pairings. The computational time of each exponentiation or each pairing does not depend on N and R, and thus constant computational costs for both **Sign** and **Verify** are achieved.

The next is the efficiency of the extended scheme in Sec. 6. By adopting the same pre-computations, the **Sign** algorithm requires 16 multi-exponentiations on $\mathcal{G}$, 6 exponentiations on $\mathcal{F}$, and 6 multi-exponentiations on $\mathcal{T}$. The **Verify** algorithm requires 6 multi-exponentiations on $\mathcal{G}$, 3 exponentiations on $\mathcal{F}$, 6 multi-exponentiations on $\mathcal{H}$, 6 multi-exponentiations on $\mathcal{T}$, and 6 pairings. Thus, we achieve $O(1)$ computational costs in signing/verifying in both schemes, although the extended one has some overheads for obtaining $O(\sqrt{N})$-size public key.

Data size. Here, we discuss the data size. To confirm the practicality, we use the following concrete parameters. To obtain the 112-bit security level, we can represent $\mathcal{G}$- and $\mathcal{F}$-elements with 224 bits for the ECC DL security. We assume the BN curves [4] with efficient pairing computations and the embedding degree 12. Then, we can represent $\mathcal{T}$-elements with 2688 bits, which satisfies the DL security corresponding the 112-bit security level.

Note that both schemes have signatures with constant sizes. In the above concrete setting, the signature of the basic scheme is about 650 Bytes. This is because the signature σ has 4 $\mathcal{G}$-elements, 3 $\mathcal{F}$-elements, and 16 Z_p-elements. On the other hand, the signature of the extended scheme is about 1,200 Bytes, since the signature has 10 $\mathcal{G}$-elements, 3 $\mathcal{F}$-elements, and 30 Z_p-elements.

Next, we discuss the public key size. In the basic scheme, the length of gpk is $O(N)$, due to the dominant $F_1, \ldots, F_N$. On the other hand, in the extended one, it is reduced to $O(\sqrt{N})$, due to the dominant $F_1, \ldots, F_{N_1}$, $\dot{F}_0, \ldots, \dot{F}_{N_2}$. To confirm the practicality of the extended one, we also use the above concrete parameters. Then, in case of $N = 1,000,000$, the public parameters $F_1, \ldots, F_{N_1}, \dot{F}_0, \ldots,$ $\dot{F}_{N_2}$ only need about 84 KBytes in total. This concrete size shows the sufficient practicality of the storage, not only in usual PCs but also in smart phones. Furthermore, since clients have only to download the public key once, the communication cost does not matter.

As the final remark, in both schemes, the length of $\boldsymbol{RL}_t$ is $O(R)$.

8 Conclusion

In this paper, we have proposed revocable group signature schemes, where both signing and verifying require only constant computational costs w.r.t. the group size and the number of revoked members. In the schemes, any secret key update is not required, and the data related to revocation has $O(R)$ size.

One of our future works is to integrate this scheme into anonymous client authentications in WEB services, and to evaluate it in practical environments. Other future works are to decrease the size of the revocation list and to exclude the random oracle.

Acknowledgments

This work was supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE) from the Ministry of Internal Affairs and Communications of Japan. We would like to thank the anonymous reviewers and Yasuyuki Nogami for helpful comments.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
2. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. Cryptology ePrint Archive: Report 2008/136 (2008); this is the extended version of [1]
3. Barreto, P.S.L.M., Galbraith, S.D., O'hEigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007)
4. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
5. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
8. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proc. 11th ACM Conference on Computer and Communications Security (ACM-CCS 2004), pp. 168–177 (2004)
9. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
10. Bresson, E., Stern, J.: Group signature scheme with efficient revocation. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001)
11. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proc. 11th ACM Conference on Computer and Communications Security (ACM-CCS 2004), pp. 132–145 (2004)

12. Camenisch, J., Groth, J.: Group signatures: Better efficiency and new theoretical aspects. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 120–133. Springer, Heidelberg (2005)
13. Camenisch, J., Herreweghen, E.V.: Design and implementation of the idemix anonymous credential system. In: Proc. 9th ACM Conference on Computer and Communications Security (ACM-CCS 2002), pp. 21–30 (2002)
14. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
15. Chaum, D., van Heijst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 241–246. Springer, Heidelberg (1991)
16. Furukawa, J., Imai, H.: An efficient group signature scheme from bilinear maps. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 455–467. Springer, Heidelberg (2005)
17. Furukawa, J., Imai, H.: An efficient group signature scheme from bilinear maps. IEICE Trans. Fundamentals E89-A(5), 1328–1338 (2006)
18. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
19. Hess, F., Smart, N., Vercauteren, F.: The eta pairing revisited. IEEE Trans. Information Theory 52(10), 4595–4602 (2006)
20. Isshiki, T., Mori, K., Sako, K., Teranishi, I., Yonezawa, S.: Using group signatures for identity management and its implementation. In: Proc. 2nd ACM Workshop on Digital Identity Management, pp. 73–78 (2006)
21. Nakanishi, T., Funabiki, N.: A short verifier-local revocation group signature scheme with backward unlinkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 17–32. Springer, Heidelberg (2006)
22. Nakanishi, T., Funabiki, N.: Short verifier-local revocation group signature scheme with backward unlinkability. IEICE Trans. Fundamentals E90-A(9), 1793–1802 (2007)
23. Nakanishi, T., Kubooka, F., Hamada, N., Funabiki, N.: Group signature schemes with membership revocation for large groups. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 443–454. Springer, Heidelberg (2005)
24. Nakanishi, T., Sugiyama, Y.: A group signature scheme with efficient membership revocation for reasonable groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 336–347. Springer, Heidelberg (2004)
25. Teranishi, I., Sako, K.: k-times anonymous authentication with a constant proving cost. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 525–542. Springer, Heidelberg (2006)

An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials

Jan Camenisch[1], Markulf Kohlweiss[2], and Claudio Soriente[3]

[1] IBM Research Zurich
jca@zurich.ibm.com
[2] Katholieke Universiteit Leuven / IBBT
markulf.kohlweiss@esat.kuleuven.be
[3] University of California, Irvine
csorient@ics.uci.edu

Abstract. The success of electronic authentication systems, be it eID card systems or Internet authentication systems such as CardSpace, highly depends on the provided level of user-privacy. Thereby, an important requirement is an efficient means for revocation of the authentication credentials. In this paper we consider the problem of revocation for certificate-based privacy-protecting authentication systems. To date, the most efficient solutions for revocation for such systems are based on cryptographic accumulators. Here, an accumulate of all currently valid certificates is published regularly and each user holds a *witness* enabling her to prove the validity of her (anonymous) credential while retaining anonymity. Unfortunately, the users' witnesses must be updated at least each time a credential is revoked. For the know solutions, these updates are computationally very expensive for users and/or certificate issuers which is very problematic as revocation is a frequent event as practice shows.

In this paper, we propose a new dynamic accumulator scheme based on bilinear maps and show how to apply it to the problem of revocation of anonymous credentials. In the resulting scheme, proving a credential's validity and updating witnesses both come at (virtually) no cost for credential owners and verifiers. In particular, updating a witness requires the issuer to do only one multiplication per addition or revocation of a credential and can also be delegated to untrusted entities from which a user could just retrieve the updated witness. We believe that thereby we provide the first authentication system offering privacy protection suitable for implementation with electronic tokens such as eID cards or drivers' licenses.

Keywords: dynamic accumulators, anonymous credentials, revocation.

1 Introduction

The desire for strong electronic authentication is growing not only for the Internet but also in the physical world where authentication tokens such as electronic identity cards, driving licenses, and e-tickets are being widely deployed and are

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 481–500, 2009.

set to become a pervasive means of authentication. It has been realized that thereby the protection of the citizens' privacy is of paramount importance and hence that the principle of *data minimization* needs to be applied: any individual should only disclose the minimal amount of personal information necessary for the transaction at hand. While privacy is of course not a major concern for the primary use of these tokens, e.g., for e-Government, it becomes vital for their so-called secondary use. For instance, when accessing a teenage chat room with an e-ID card, users should only have to reveal that they are indeed between, say, 10 and 16 years old but should not reveal any other information stored on the card such as birth date, name or address.

In the literature, there exist a fair number of privacy-preserving technologies that allow one to meet these requirements. These technologies include anonymous credential systems [1,2,3], pseudonym systems [4,5,6,7], anonymous e-cash [8,9,10], or direct anonymous attestation [11]. Almost all of these schemes exhibit a common architecture with certificate issuers, users (certificate recipients) and certificate verifiers: Users obtain a signature from an issuing authority on a number of attributes and, at later time, can convince verifiers that they indeed possess a signature on those attributes [12]. Individual transactions are anonymous and unlikable by default and users can select which portions of a certificate to reveal, which portions to keep hidden, and what relations between certified items to expose.

A crucial requirement for all authorization and authentication systems is that certificates issued can be later revoked, in case of unexpected events or malicious use of the certificate. For traditional certificates, this is typically achieved either by publishing a certificate revocation list or by enforcing a short certificate lifetime via expiration date. For anonymous certificates, the former approach violates privacy while the latter is typically rather inefficient as it would require the users to frequently engage in the usually quite involved issuing protocol.

In principle, the approach of certificate revocation list can be made to work also for anonymous credentials by having the user to prove in zero-knowledge that her certificate is not contained on the (black) list. Such a proof, however, would not be efficient as the computational and communication cost of the user and the verifier become preventive as they grow at least logarithmic with number of entries in the list. The literature provides two kinds solutions that overcome this.

The first kind is called verifier local revocation [13,14,11,15]. In the best solution here, the cost for the user is independent of the number of entries in the revocation list, but the computational cost of the verifier is linear in this number (at least a modular exponentiation or, worse, a pairing operation per entry). Thus, these solutions are not at all suited for large scale deployments.

The second kind [16,17] employs cryptographic accumulators [18]. Such accumulators allow one to hash a large set of inputs in a single short value, the *accumulator,* and then provide evidence by an *accumulator witness* that a given value is indeed contained in the accumulator. Thus, the serial numbers of all currently valid credentials are accumulated and the resulting value is published. Users can then show to verifiers that their credential is still valid, by using their witness to prove (in zero-knowledge) that their credential's serial number is contained in the published accumulator. Such proofs can be realized with practical efficiency [16,17] and incur only cost to the user and the verifier that are independent of the number of revoked or currently valid credentials. The drawback of these solutions, however, is that the users need to update their accumulator witnesses and an update requires at least one modular exponentiation for each newly revoked credential. Assuming a driving license application and based on the, e.g., 0.07% rate of driver's license revocation in West Virginia USA [19], the number of credentials revoked will quickly become a couple of thousands per day. Thus, these solutions incur a computational (and communication) cost far greater that what an electronic token such as a smart card can possibly handle.

Our contribution. In this paper we are therefore considering revocation solutions that incur (virtually) no cost to the verifier and the users, and only limited costs to the issuer (or the revocation authority). More precisely, for each revocation epoch (e.g., every day), verifiers and users need to retrieve the issuer's current public key (i.e., the new accumulator value) while users further need to retrieve their witnesses (a single group element). Upon revocation of a credential, the revocation authority only needs to perform one multiplication per remaining user to update (and provide) the users' witnesses, a cost which can easily be handled by today's standards. We note that this update operation requires no secret keys and does not need to be performed by the issuer, i.e., it could be performed by other untrusted entities.

As building block for this solution, we introduce a novel dynamic accumulator based on bilinear maps and show how to employ it for revocation at the example of the Bangerter, Camenisch and Lysyanskaya private certificate framework [12], which is essentially a generalization of e-cash, anonymous credentials, and group signatures. Thus we provide for the first time a practical solution for anonymous authentication with e-ID cards.

Related Work. Camenisch and Lysyanskaya [17] introduce a dynamic accumulator and show its applicability to revocation in Anonymous Credential Systems as well as Identity Escrow and Group Signatures. Update of the proposed accumulator, as well as user witnesses, require a number of exponentiations that is linear in the number of users added to or revoked from the system. In [20], the authors extend the above accumulator, introducing witnesses and proofs that a value was *not* accumulated.

Nguyen [21] constructs a dynamic accumulator from bilinear pairings. Its application to an anonymous credential system require users to store large system parameters, in order to prove validity of their credential. Moreover, updating a witness takes one exponentiation per event and therefore is not efficient enough for what we are after (in the paper the authors write multiplication and use addition as base operation for the algebraic group as is done sometimes in connection with bi-linear maps and elliptic curve groups).

In [22], the authors propose a dynamic accumulator for batch update. Users who missed many witness updates, can request update information to the issuer and update their witness with one multiplication. In our scheme, we can provide the same feature, relaxing the requirement that the issuer takes part to the witness update. We note, however, that the authors do not show how to achieve an efficient proof of knowledge of an element contained in the accumulator as is needed for the use of the accumulator for revocation of credentials.

Outline. The rest of the paper is organized as follow. In Section 2 we discuss assumptions and recall existing building blocks. In Section 3 we introduce our novel dynamic accumulator. In Section 4 we show how to extend the Bangerter et al. private certificate framework with an efficient revocation mechanism. Conclusion and further discussion are given in Section 5.

2 Preliminaries

In this section we list assumptions and cryptographic tools used as building blocks of the introduced accumulator as well as our anonymous credential revocation system.

A function ν is *negligible* if, for every integer c, there exists an integer K such that for all $k > K$, $|\nu(k)| < 1/k^c$. A problem is said to be *hard* (or *intractable)* if there exists no probabilistic polynomial time (p.p.t.) algorithm on the size of the input to solve it.

Bilinear Pairings. Let G and G_T be groups of prime order q. A map $e : G \times G \rightarrow G_T$ must satisfy the following properties:

(a) *Bilinearity*: a map $e : G \times G \rightarrow G_T$ is bilinear if $e(a^x, b^y)t = e(a, b)^{xy}$;
(b) *Non-degeneracy*: for all generators $g, h \in G$, $e(g, h)$ generates G_T;
(c) *Efficiency*: There exists an efficient algorithm $\mathsf{BMGen}(1^k)$ that outputs (q, G, G_T, e, g) to generate the bilinear map and an efficient algorithm to compute $e(a, b)$ for any $a, b \in G$.

The security of our scheme is based on the following number-theoretic assumptions. Our accumulator construction is based on the Diffie-Hellman Exponent assumption. The unforgeability of credentials is based on the Strong Diffie-Hellman assumption. For credential revocation we need to prove possession of an accumulator witness for a credential. This proof is based on our new Hidden Strong Diffie-Hellman Exponent (SDHE) assumption.

Definition 1 (n-DHE). *Diffie-Hellman Exponent (DHE) assumption: The n-DHE problem in a group G of prime order q is defined as follows: Let $g_i = g^{\gamma^i}$, $\gamma \leftarrow_R \mathbb{Z}_q$. On input $\{g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}\} \in G^{2n}$, output g_{n+1}.*

The n-DHE assumption states that this problem is hard to solve.

Boneh, Boyen, and Goh [23] introduced the Bilinear Diffie-Hellman Exponent (BDHE) assumption that is defined over a bilinear map. Here the adversary has to compute $e(g, h)^{\gamma^{n+1}} \in G_T$.

Lemma 1. *The n-DHE assumption for a group G with a bilinear pairing $e : G \times G \rightarrow G_T$ is implied by the n-BDHE assumption for the same groups.*

Boneh and Boyen introduced the Strong Diffie-Hellman assumption in [24].

Definition 2 (n-SDH [24]). *On input $g, g^x, g^{x^2}, \ldots, g^{x^n} \leftarrow G$, it is computationally infeasible to output $(g^{1/(x+c)}, c)$.*

Boyen and Waters [25] introduced the Hidden Strong Diffie-Hellman assumption under which BB signatures [24] are secure for any message space. We require a variant of the Hidden Strong Diffie-Hellman assumption that we call the Hidden Strong Diffie-Hellman Exponent (n-HSDHE) assumption. The two assumptions are hitherto incomparable.

Definition 3 (n-HSDHE). *Given $g, g^x, u \in G$, $\{g^{1/(x+\gamma^i)}, g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}$, and $\{g^{\gamma^i}\}_{i=n+2\ldots 2n}$, it is infeasible to compute a new tuple $(g^{1/(x+c)}, g^c, u^c)$.*

2.1 Known Discrete-Logarithm-Based, Zero-Knowledge Proofs

In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [26], (2) proof of knowledge of equality of some elements in different representation [27], (3) proof that a commitment opens to the product of two other committed values [28,29,30], and also (4) proof of the disjunction or conjunction of any two of the previous [31].

When referring to the above proofs, we will follow the notation introduced by Camenisch and Stadler [32] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha, \beta, \delta) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers α, β, and δ such that $y = g^\alpha h^\beta$ and $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta$ holds,*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$ that have the same order. (Note that the some elements in the representation of y and $\tilde{y}$ are equal.) The convention is that values (α, β, δ) denote quantities of which knowledge is being proven (and are kept secret), while all other values are known to the verifier. For prime-order groups which include all groups we consider in this paper, it is well known that there exists a knowledge extractor which can extract these quantities from a successful prover.

2.2 Signature Scheme with Efficient Protocols

For our credential system we use a signature scheme that is loosely based on weak Boneh and Boyen signatures [24,16]. It is described in [33] and has been proven secure under the n-SDH assumption [34,35]. It assumes a non-degenerate bilinear map $e : G \times G \rightarrow G_T$ of prime order q with generators $h, h_0, h_1, \ldots, h_\ell, h_{\ell+1}$. The signer's secret key is $x \in \mathbb{Z}_q$ while the public key is $y = h^x$.

A signature on a message $m \in \mathbb{Z}_q^*$ is computed by picking $c, s \leftarrow \mathbb{Z}_q^*$ and computing $\sigma = (h_0 h_1^m h_2^s)^{\frac{1}{x+c}}$. The signature is (σ, c, s). It is verified by checking whether $e(\sigma, yh^c) = e(h_0 h_1^m h_2^s, h)$. Multiple messages $m_1, \ldots, m_\ell \in \mathbb{Z}_q^*$ can be signed as $\sigma = (h_0 h_1^{m_1} \cdots h_\ell^{m_\ell} h_{\ell+1}^s)^{\frac{1}{x+c}}$ and verification is done by checking whether $e(\sigma, yh^c) = e(h_0 h_1^m \cdots h_\ell^{m_\ell} h_{\ell+1}^s, h)$.

Proving Knowledge of a Signature. Now assume that we are given a signature (σ, c, s) on messages $m_1 \ldots, m_\ell \in \mathbb{Z}_q$ and want to prove that we indeed possess such a signature. To this end, we need to augment the public key with a value $\tilde{h} \in G$ such that $\log_h \tilde{h}$ are not known.

Knowledge of a signature is proven as follows:

1. Choose random values $r \leftarrow \mathbb{Z}_q$ and $open \leftarrow \mathbb{Z}_q$ and compute a commitment $B = h^r \tilde{h}^{open}$ and a blinded signature $A = \sigma \tilde{h}^r$.
2. Compute the following proof

$$PK\{(c, s, r, open, mult, tmp, m_1, \ldots, m_\ell) : \\ B = h^r \tilde{h}^{open} \;\wedge\; 1 = B^c h^{-mult} \tilde{h}^{-tmp} \;\wedge \\ \frac{e(h_0, h)}{e(A, y)} = e(A, h)^c \cdot e(\tilde{h}, y)^{-r} \cdot e(\tilde{h}, h)^{-mult} \cdot \prod_{i=1}^{\ell} e(h_i, h)^{-m_i} \cdot e(h_{\ell+1}, h)^{-s}\} \;.$$

Why this proof works is explained in [36].

3 A Pairing Based Dynamic Accumulator with Efficient Updates

We define and build a dynamic accumulator with efficient updates and assess its security. With efficient updates we mean that witnesses can be updated by any party without knowledge of any secret key and require only multiplications (no exponentiations) linear in the number of changes to the accumulator. Our construction is based on the broadcast encryption scheme by Boneh, Gentry and Waters [37].

3.1 Definition of Dynamic Accumulators

A secure accumulator consists of the five algorithms AccGen, AccAdd, AccUpdate, AccWitUpdate, and AccVerify.

These algorithms are used by the accumulator authority (short authority), an untrusted update entity, a user and a verifier. The authority creates an accumulator key pair (sk_A, pk_A), the accumulator $acc_\emptyset$ and a public state $state_\emptyset$ using the AccGen algorithm; it can add a new value i to the accumulator acc_V using the AccAdd algorithm to obtain a new accumulator $acc_{V \cup \{i\}}$ and state $state_{U \cup \{i\}}$, together with a witness wit_i. The accumulator for a given set of values V, can be computed using the AccUpdate algorithm.

Throughout these operations, acc_V and wit_i are of constant size (independent of the number of accumulated values). The authority does some bookkeeping about the values contained in the accumulator and the status of the accumulator when a witness wit_i was created. These sets are denoted as V and V_w respectively. The bookkeeping information is made public and is only needed for updating witnesses, it is not needed for verifying that a value is contained in an accumulator.

Each time an accumulator changes, the old witnesses become invalid. It is however possible to update all witnesses for values $i \in V$ contained in the accumulator from the bookkeeping information V_w. This updating is the most performance intensive operation in existing accumulator systems. We show how it can be efficiently offloaded to an untrusted update entity that runs AccWitUpdate and is only given the accumulator state $state_U$ and the bookkeeping information V and V_w. The accumulator state $state_U$ also contains book keeping information U, the set of elements ever added to the accumulator (but not necessarily contained in the current accumulator). This is a superset of V and V_w.[1]

After users obtained an updated witness wit'_i for a value i for the current accumulator, they can prove to any verifier that i is in the accumulator, using the AccVerify algorithm.

$\mathsf{AccGen}(1^k, n)$ creates an accumulator key pair (sk_A, pk_A), an empty accumulator $acc_\emptyset$ (for accumulating up to n values) and an initial state $state_\emptyset$.

$\mathsf{AccAdd}(sk_A, i, acc_V, state_U)$ allows the authority to add i to the accumulator. It outputs a new accumulator $acc_{V\cup\{i\}}$ and state $state_{U\cup\{i\}}$, together with a witness wit_i for i.

$\mathsf{AccUpdate}(pk_A, V, state_U)$ outputs an accumulator acc_V for values $V \subset U$.

$\mathsf{AccWitUpdate}(pk_A, wit_i, V_w, acc_V, V, state_U)$ outputs a witness wit'_i for acc_V if wit_i was a witness for acc_{V_w} and $i \in V$.

$\mathsf{AccVerify}(pk_A, i, wit_i, acc_V)$ verifies that $v \in V$ using an up-to-date witness wit_i and the accumulator acc_V. In that case the algorithm accepts, otherwise it rejects.

Note that the purpose of an accumulator is to have accumulator and witnesses of size independent of the number of accumulated elements.

Correctness. Correctly accumulated values have verifying witnesses.

Security. For all probabilistic polynomial time adversaries $\mathcal{A}$,

$$\begin{aligned} &Pr[(sk_A, pk_A, acc_\emptyset, state_\emptyset) \leftarrow \mathsf{AccGen}(1^k); \\ &\qquad (i, wit_i) \leftarrow \mathcal{A}(pk_A, acc_\emptyset, state_\emptyset)^{\mathcal{O}_{\mathsf{AccAdd}}(\cdot), \mathcal{O}_{\mathsf{AccUpdate}}(\cdot)} : \\ &\qquad\quad \mathsf{AccVerify}(pk_A, i, wit_i, acc_\mathcal{O}) = \mathsf{accept} \wedge i \notin V_\mathcal{O}] \;=\; neg(k)\ , \end{aligned}$$

[1] Allowing accumulators to change their state over time can allow for better performance tradeoffs. While our accumulator construction does not use this possibility in order to keep things simple, we outline such an optimization in [36].

where the oracles $\mathcal{O}_{\mathsf{AccAdd}}(.)$ and $\mathcal{O}_{\mathsf{AccUpdate}}(.)$ keep track of shared variables $acc_{\mathcal{O}}$, $state_{\mathcal{O}}$ and a set $V_{\mathcal{O}}$ that is initialized to $\emptyset$. The oracle $\mathcal{O}_{\mathsf{AccAdd}}(i)$ computes and outputs $(acc_{\mathcal{O}}, state_{\mathcal{O}}, wit_i) \leftarrow \mathsf{AccAdd}(sk_A, i, acc_{\mathcal{O}}, state_{\mathcal{O}})$ and adds i to $V_{\mathcal{O}}$ while $\mathcal{O}_{\mathsf{AccUpdate}}(V)$ computes and outputs $acc_{\mathcal{O}} \leftarrow \mathsf{AccUpdate}(pk_A, V, state_{\mathcal{O}})$ and sets $V_{\mathcal{O}}$ to V.

3.2 Construction

We now construct the algorithms AccGen, AccAdd, AccUpdate, AccWitUpdate, and AccVerify.

$\mathsf{AccGen}(1^k, n)$. Run $\mathsf{BMGen}(1^k)$ to obtain the setup $params_{BM} = (q, G, G_T, e, g)$ of a bilinear map $e : G \times G \rightarrow G_T$.

Pick a random value $\gamma \in \mathbb{Z}_q$. Generate a key pair sk and pk for a secure signature scheme, for instance the BB signature scheme that is secure under the SDH assumption. Let $pk_A = (params_{BM}, pk, z = e(g, g)^{\gamma^{n+1}})$, $sk_A = (params_{BM}, \gamma, sk)$, $acc_{\emptyset} = 1$ and $state_{\emptyset} = (\emptyset, g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}})$.[2]

$\mathsf{AccAdd}(sk_A, i, acc_V, state_U)$. Compute $w = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}$ and a signature σ_i on $g_i \| i$ under signing key sk. The algorithm outputs $wit_i = (w, \sigma_i, g_i)$, an updated accumulator value $acc_{V \cup \{i\}} = acc_V \cdot g_{n+1-i}$, and $state_{U \cup \{i\}} = (U \cup \{i\}, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$.

$\mathsf{AccUpdate}(pk_A, V, state_U)$. Check whether $V \subset U$ and outputs $\perp$ otherwise. The algorithm outputs $acc_V = \prod_{v \in V} g_{n+1-v}$ for values $i \in V$.

$\mathsf{AccWitUpdate}(pk_A, wit_i, V_w, acc_V, V, state_U)$. Parse wit_i as (w, σ_i, g_i). If $i \in V$ and $V \cup V_w \subset U$, compute

$$w' = w \cdot \frac{\prod\limits_{j \in V \setminus V_w} g_{n+1-j+i}}{\prod\limits_{j \in V_w \setminus V} g_{n+1-j+i}} .$$

Output the updated witness $wit'_i = (w', \sigma_i, g_i)$. Otherwise output $\perp$.

$\mathsf{AccVerify}(pk_A, i, wit_i, acc_V)$. Parse $wit_i = (w, \sigma_i, g_i)$. Output accept, if σ_i is a valid signature on $g_i \| i$ under verification key pk and $\frac{e(g_i, acc_V)}{e(g, w)} = z$. Otherwise output reject.

In the construction above, we accumulate the group elements $g_1, \ldots, g_n$ instead of, e.g., the integers $1, \ldots, n$. Depending on the application, one would want to accumulate the latter, or more generally an arbitrary set of size n. In this case, the issuer of the accumulator would need to publish a mapping from this set to the g_i values that get actually accumulated. In order to avoid large public parameters during verification the issuer of the accumulator uses a signature scheme

[2] We define $state_U = (U, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ where U is book keeping information that keeps track of all elements that were ever added to the accumulator (but might have been subsequently removed). The rest of the state is static. See [36] for a modification that reduces the size of $state_U$.

to sign the g_i together with the value to which they map. Thus, the verifier can check whether a given g_i is a (potentially) valid input to the accumulator (cf. discussion in Section 3.3).

We also note that the algorithm to update the witness does not require any secret information. Thus, the witnesses could be kept up-to-date for the users either by the users themselves, the issuer, or by some third party. In the latter two cases, the users can just retrieve the current valid witness whenever needed. In applications, one would typically define epochs such that the accumulator value and witnesses are only updated at the beginning of each epoch and remain valid throughout the epoch. Finally note that maintaining the witnesses for all users is well within reach of current technologies — indeed, all witnesses can be kept in main memory and the update performed rather quickly.

Correctness. Let acc_V be an accumulator for $sk_A = (params_{BM}, \gamma, sk)$, $pk_A = (params_{BM}, pk, z = e(g,g)^{\gamma^{n+1}})$, and $state_U = (U, g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}})$. Then a correct accumulator always has a value $acc_V = \prod_{j \in V} g_{n+1-j}$. Moreover, for each $i \in V$ with up-to-date witness $wit_i = (w = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}, \sigma_i, g_i)$ the following equation holds:

$$\frac{e(g_i, acc_V)}{e(g,w)} = \frac{e(g,g)^{\sum_{j \in V} \gamma^{n+1-j+i}}}{e(g,g)^{\sum_{j \in V}^{j \neq i} \gamma^{n+1-j+i}}} = e(g,g)^{\gamma^{n+1}} = z \ .$$

Security. Suppose there exists an adversary $\mathcal{A}$ that breaks the security of our accumulator. We show how to construct an algorithm $\mathcal{B}$ that either forges the signature scheme used to sign accumulated elements or breaks the n-DHE assumption.

Algorithm $\mathcal{B}$ has access to a signing oracle $\mathcal{O}_\sigma$ and obtains as input the corresponding signature verification key pk, the parameters of a bilinear map $params_{BM} = (q, G, G_T, e, g)$, and an instance of the n-DHE assumption $(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in G^{2n-1}$. $\mathcal{B}$ provides $\mathcal{A}$ with $pk_A = (params_{BM}, pk, z = e(g_1, g_n))$, $acc_\emptyset = 1$ and $state_\emptyset = (\emptyset, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$. The oracle queries of the adversary are answered as defined in the game except that $\mathcal{O}_\sigma$ is called for creating the signatures.

Given an adversary that can compute (i, wit_i) such that the verification succeeds even though $i \notin V_\mathcal{O}$. We parse wit_i as $(w, \hat{\sigma}_i, \hat{g}_i)$. If $\hat{g}_i$ does not correspond to g_i the adversary attacked the signature and $\hat{\sigma}_i$ is a signature forgery. Otherwise we learn from the verification equation that

$$e(g_i, acc_\mathcal{O}) = e(g, w)z$$

and

$$e(g, \prod_{j \in V} g_{n+1-j+i}) = e(g, w g_{n+1}) \ .$$

This means that

$$g_{n+1} = \frac{\prod_{j \in V} g_{n+1-j+i}}{w} \ .$$

For $i \in \{1, \ldots, n\} \setminus V$, all $g_{n+1-j+i}$ are contained in $state_{\emptyset}$ and it is possible to compute this value. This breaks the n-DHE assumption.

3.3 Efficient Proof That a Hidden Value Was Accumulated

It is often only required for a user to prove that she possesses a value that is indeed contained in the current accumulator, or in other words, to prove membership of the current accumulator without revealing which value she possesses (or which index i is assigned to her). In this section, we give an efficient protocol that achieves this for our accumulator construction.

For the accumulator to be secure, the verifier needs to check that the value the user claims to own, is one of $g_1, \ldots, g_n$. In the previous construction, $g_1, \ldots, g_n$ are authenticated either by making them public as a whole or by having each one signed (in which case the user would provide the g_i and the signature to the verifier). However, using a public list would require the prover to either reveal g_i (which would violate privacy) or to prove that the g_i which she claims possession of, is a valid one. The latter, however, would require an involving proof that would make the use of the accumulator inefficient. We therefore resort to sign g_i values and then require the prover to prove that she knows a signature by the accumulator issuer on "her" g_i without revealing neither the signature nor the g_i value. As such a proof needs to be efficient, this requires a special signature scheme. Since user never reveal the accumulated valued they are proving possession of, it is possible to avoid signing $g_i||i$ as it is done in Section 3.2. This allows for a more efficient signature scheme and proof system.

Prerequisites. We instantiate the signature scheme used for signing the g_i with a variant of the weakly secure Boneh-Boyen scheme [24]. Instead of a g_i value we sign γ^i. The authentic g_i is a by-product of the signing process. For simplicity we reduce the security of the accumulator proof directly to the n-HSDHE assumption.[3] The n-HSDHE assumption is the weakest assumption under which we can prove our scheme. The n-HSDHE assumption is implied by the iHSDH assumption of [38].

The signer (the accumulator issuer) picks a fresh $u \leftarrow G$, secret key $sk \leftarrow \mathbb{Z}_q$ and public key $pk = g^{sk}$. A signature consists of the two elements $\sigma_i = g^{1/(sk+\gamma^i)}$ and $u_i = u^{\gamma^i}$ and is verified by checking that $e(pk \cdot g_i, \sigma_i) = e(g, g)$.

Let $pk_A = (params_{BM}, pk, z = e(g,g)^{\gamma^{n+1}})$, $sk_A = (params_{BM}, \gamma, sk)$ and $state_U = (\emptyset, g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}})$ be as generated by the accumulator operations in the previous section. We also pick an additional $\tilde{h} \leftarrow G$ for commitments. The discrete logarithm of h and u with respect to g must be unknown to the prover.

Proof of Knowledge. For arbitrary $V \subset \{1, \ldots, n\}$ and $i \in V$, on input $acc_V = \prod_{i \in V} g_{n+1-i}$ and the corresponding witness $wit_i = (w, \sigma_i, u_i, g_i)$, where $w = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}$, for value i, the prover performs the following randomization:

[3] We do not prove the signature scheme itself secure, but we refer to [38] for a similar scheme.

Pick at random $r, r', r'', r''', open \in \mathbb{Z}_q$ and computing $\mathcal{G} = g_i\tilde{h}^r$, $\mathcal{W} = w\tilde{h}^{r'}$, $D = g^r\tilde{h}^{open}$, $\mathcal{S} = \sigma_i\tilde{h}^{r''}$, and $\mathcal{U} = u_i\tilde{h}^{r'''}$ respectively. Then the prover, proves

$$PK\{(r, r', r'', r''', open, mult, tmp) : D = g^r\tilde{h}^{open} \wedge 1 = D^{r''} g^{-mult}\tilde{h}^{-tmp} \wedge$$
$$\frac{e(pk \cdot \mathcal{G}, \mathcal{S})}{e(g,g)} = e(pk \cdot \mathcal{G}, \tilde{h})^{r''} e(\tilde{h},\tilde{h})^{-mult} e(\tilde{h}, \mathcal{S})^r \wedge$$
$$\frac{e(\mathcal{G}, acc_V)}{e(g, \mathcal{W})z} = e(\tilde{h}, acc_V)^r e(1/g, \tilde{h})^{r'} \wedge \frac{e(\mathcal{G}, u)}{e(g, \mathcal{U})} = e(\tilde{h}, u)^r e(1/g, \tilde{h})^{r'''}\} \ .$$

Theorem 1. *Under the n-DHE and the n-HSDHE assumptions the protocol above is a proof of knowledge of a randomization value r that allows to derandomize $\mathcal{G}$ to a value g_i, where i is accumulated in acc_V, i.e., $i \in V$.* The proof of this theorem can be found in Section A.1.

4 Efficient Revocation of Private Certificates

In this section we will show how to employ our accumulator to achieve efficient revocation for schemes where users get some form of certificate and then later can use these certificates in an anonymity protecting way. Such schemes include group signatures, anonymous credential systems, pseudonym systems, anonymous e-cash, and many others. Most of these schemes work as follows. In a first phase an issuer provides the user with a signature on a number of messages. Then, in a second phase the user convinces the verifier that 1) she owns a signatures by the issuer on a number of messages and 2) that these messages satisfy some further properties that are typically dependent on the particular purpose of the scheme. Based on this observation, Bangerter et al. [12] give a cryptographic framework for the controlled release of certified information. They also show how different applications (such as the ones mentioned above) can be realized. Thus, they basically generalize the concepts of anonymous credentials, anonymous e-cash, and group signatures into a single framework. We therefore just show how their framework can be extended with revocation to provide this features for all these applications. From this it will become clear how to extend particular schemes (e.g., the anonymous credentials and group signatures [16,33]) with our revocation mechanisms.

More precisely, Bangerter et al. employ special signature protocols, called CL signatures [39], for issuing *private certificates* to users. A private certificate (1) consists of attributes and a signature over the attributes much alike a traditional certificate, only that a more powerful signature scheme is used, i.e.,

$$cert = (\sigma, m_1, \ldots, m_l) \text{ with } \sigma = \mathsf{Sign}(m_1, \ldots, m_l; sk_I) \ . \tag{1}$$

Let $(sk_I, pk_I) \leftarrow \mathsf{IssuerKeygen}(1^k)$ be the certificate issuer's keypair. The framework supports two types of protocols: 1) an interactive certificate issuing protocol $\mathsf{ObtainCert}$ that allows to obtain a signature on committed values without revealing these values and 2) efficient zero-knowledge proofs of knowledge of signature possession.

Let $(m_1, \ldots, m_\ell)$ denote a list of data items and $H \subset L = \{1, \ldots, \ell\}$ a subset of data items. Using the first protocol, a user can obtain a certificate on $(m_1, \ldots, m_\ell)$ such that the issuer does not learn any information on the data items in H, while it learns the other data items, i.e., $L \setminus H$.

The private certificates of a user remain private to the user, that is, they are never released (as a whole) to any other party: when using (showing) certificates for asserting attribute information, the user proves that she knows (has) certificates with certain properties. The user may release certain attributes, while only proving the knowledge of the rest of the certificate:

$$\mathsf{PK}\{(\sigma, m_1, \ldots, m_{\ell'}) : 1 = \mathsf{VerifySign}(\sigma, m_1, \ldots m_{\ell'}, m_{\ell'+1}, \ldots, m_\ell; pk_I) \wedge \ldots\} \ .$$

In the above proof only the attribute values of $m_{\ell'+1}$ to m_ℓ are revealed.

Certificate revocation. We now extend the above framework with certificate revocation as follows. Let V be the set of valid certificates for an epoch with epoch information $epoch_V$. A certificate is assigned a unique identifier i (which will be embedded into it as one of the attributes) and a witness wit_i. We require that the user can prove to a verifier that she possesses a non-revoked certificate only if $i \in V$. This is achieved by having the user prove that the identifier embedded into her credential is a valid one for the current epoch. Thus, before engaging in a proof, the user needs to update her witness and both parties (the user and the verifier) need to obtain the most up-to-date epoch information $epoch_V$ for V. The user can either update the witness herself, or just retrieve the currently valid witness from a witness update entity. Indeed, a witness update computation does not require knowledge of any secret and can be performed by untrusted entities (e.g., by a third party or a high availability server cluster at the issuer). In particular, those entities are only responsible for computing user witnesses according to the current epoch information. Misbehavior by such entities would lead in a denial of service (the verification algorithm would *reject*, but would not break the security of the system). Also note that a witness update requires a number of multiplications that is linear in the number of elements added to or removed from the accumulator, hence providing such an update service to users is feasible (one could even hold all users' witnesses in main memory).

More formally, a certificate revocation system for the certification framework consists of updated IssuerKeygen and ObtainCert protocols, new algorithms UpdateEpoch and UpdateWitness for managing revocation, and a zero-knowledge proof system for a new predicate VerifyEpoch that allows to prove possession of a witness wit_i:

$\mathsf{IssuerKeygen}(1^k, n)$ creates the issuer key pair (sk_I, pk_I), the epoch information $epoch_\emptyset$, and $state_\emptyset$ for issuing up to n certificates.

$\mathsf{ObtainCert}(\mathcal{U}(pk_I, H, \{m_j\}_{j \in H}), \mathcal{I}(sk_I, H, \{m_j\}_{j \in L \setminus H}, epoch_V, state_U))$ allows a user to obtain a private certificate $cert_i$ from the issuer. The issuer computes and publishes the user's witness wit_i, and updated epoch information $epoch_{V \cup \{i\}}$ and $state_{U \cup \{i\}}$.

$\mathsf{UpdateEpoch}(V, \mathit{state}_U)$ outputs epoch information epoch_V, if $V \subset U$. Otherwise it outputs $\perp$.

$\mathsf{UpdateWitness}(\mathit{wit}_i, \mathit{epoch}_V, \mathit{state}_U)$ outputs an updated witness wit'_i if $V \subset U$. Otherwise it outputs $\perp$.

A user who knows a certificate cert_i and a corresponding up-to-date witness wit_i can prove, to a verifier, possession of the certificate and its validity for the current epoch using the new predicate $\mathsf{VerifyEpoch}$ as follows. The user's secret input is cert_i. The common input of the protocol is the issuer's public key pk_I, the epoch information epoch_V, and a specification of the proof statement (this includes the information revealed about the certificate). In the example below the user chooses to keep the first ℓ' messages secret while he reveals the rest of the messages.

$$\begin{aligned}\mathsf{PK}\{(\sigma, m_1, \ldots, m_{\ell'}, i, \mathit{wit}_i) : 1 =& \mathsf{VerifySign}(\sigma, m_1, \ldots m_{\ell'}, m_{\ell'+1}, \ldots, m_\ell, i; \mathit{pk}_I) \wedge \\ 1 =& \mathsf{VerifyEpoch}(i, \mathit{wit}_i; \mathit{epoch}_V, \mathit{pk}_I)\}\ .\end{aligned}$$

Using the Bangerter et al. framework [12], it is not hard to extend this proof or combine it with other proof protocols given therein.

4.1 Adapted Signature Scheme for Accumulated Values

As described above, a user would have to prove that the value i encoded into her credential is also contained in the current accumulator. However, the accumulator construction as given in the previous section does not allow one to accumulate i directly but only $g_i = \tilde{g}^{\gamma^i}$. Now, instead of introducing a mapping of i to g_i (and including this in our proofs which would make them inefficient), we are going to make the mapping implicit by including g_i into the credential. Thus, the g_i values will be used both in the private certificate and the accumulator to represent the certificate id i. This requires that we extend the signature scheme in Section 2.2 to allow verification without knowing the secret exponent γ^i:

1. The signer creates $g, h, h_0, h_1, \ldots, h_\ell, h_{\ell+1} \leftarrow G$ and creates keys $x \in \mathbb{Z}_q$ and $y = h^x$.
2. Next, the signer publishes a list $(g_1 = g^\gamma, \ldots, g_n = g^{\gamma^n})$ that he allows in signatures.
3. The signer picks random $c, s \leftarrow \mathbb{Z}_q^*$ and then computes the signature as $(\sigma = (h_0 h_1^{m_1} \cdots h_\ell^{m_\ell} g_i h_{\ell+1}^s)^{\frac{1}{x+c}}, c)$.
4. A signature (σ, c, s) on messages $m_1, \ldots, m_\ell, \hat{g}_i$ is verified by checking that $\hat{g}_i$ is in the list of g_i values and that $e(\sigma, yh^c) = e(h_0(\prod_{j=1}^{\ell} h_j^{m_j})\hat{g}_i h_{\ell+1}^s, h)$ holds.

We note that the check that $\hat{g}_i$ is in the list of g_i values as prescribed in the last step will later on be replaced by a signature/authenticator on g_i as done for the accumulator in Section 3.3.

It is straightforward to reduce the security of this modified signature scheme to the original one with $\ell + 1$ messages as the signer knows the "messages" γ^i encoded by the g_i. We omit the details here.

4.2 Construction

$\mathsf{IssuerKeygen}(1^k, n)$. Run $\mathsf{BMGen}(1^k)$ to generate the parameters $params_{BM} = (q, G, G_T, e, g)$ of a (symmetric) bilinear map $e : G \times G \to G_T$. Pick additional bases $h, h_0, \ldots, h_{\ell+1}, \tilde{h}, u \leftarrow G$ and $x, sk, \gamma \leftarrow \mathbb{Z}_q$ and compute $y = h^x$ and $pk = g^{sk}$.[4] Compute $g_1, ..., g_n, g_{n+2}, ..., g_{2n}$, where $g_i = g^{\gamma^i}$, and $z = e(g, g)^{\gamma^{n+1}}$.

Output $(sk_I, pk_I) = ((params_{BM}, x, sk, \gamma), (params_{BM}, y, h, h_0, \ldots, h_{\ell+1}, \tilde{h}, u, pk, z))$, $epoch_{\emptyset} = (acc_{\emptyset} = 1, \emptyset)$, and $state_{\emptyset} = (\emptyset, g_1, ..., g_n, g_{n+2}, ..., g_{2n})$.

$\mathsf{ObtainCert}(\mathcal{U}(pk_I, H, \{m_j\}_{j \in H}), \mathcal{I}(sk_I, H, \{m_j\}_{j \in L \setminus H}, epoch_V, state_U)$. The user runs the following protocol to obtain a certificate $cert_i$ from the issuer:

1. The user chooses a random $s' \in \mathbb{Z}_q^*$, computes $X = \prod_{j \in H} h_j^{m_j} h_{\ell+1}^{s'}$, and sends X to the issuer.
2. The user (as prover) engages the issuer (as verifier) in the following proof

$$PK\{\{m_j\}_{j \in H}, s') : X = \prod_{j \in H} h_j^{m_j} h_{\ell+1}^{s'}\} ,$$

which will convince the issuer that X is correctly formed.

3. The issuer parses $epoch_V$ as (acc_V, V) and $state_U$ as $(U, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$. He then computes $epoch_{V \cup \{i\}} = (acc_V \cdot g_{n+1-i}, V \cup \{i\})$ and $state_{U \cup \{i\}} = (U \cup \{i\}, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ [5] .
4. The issuer chooses random $c, s'' \in \mathbb{Z}_q^*$ and then computes the signature $\sigma = ((\prod_{j \in L \setminus H} h_j^{m_j}) X g_i h_{\ell+1}^{s''})^{1/(x+c)}$.
5. The issuer computes $w = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}$, $\sigma_i = g^{1/(sk+\gamma^i)}$, and $u_i = u^{\gamma^i}$ and sets $wit_i = (\sigma_i, u_i, g_i, w, V_{\cup}\{i\})$.
6. The issuer sends $(\sigma, c, s'', \{m_j\}_{j \in L \setminus H}, g_i, i)$ to the user and outputs wit_i, $epoch_{V \cup \{i\}}$, and $state_{U \cup \{i\}}$.
7. The user verifies the certificate gotten and outputs $cert_i = (\sigma, c, m_1, \ldots, m_\ell, g_i, s = s' + s'', i)$.

$\mathsf{UpdateEpoch}(V, state_U)$ checks whether $V \subset U$ and outputs $\perp$ otherwise. The algorithm creates $epoch_V$ for proving possessions of $cert_i$, $i \in V$. Let $acc_V = \prod_{i \in V} g_{n+1-i}$, output $epoch_V = (acc_V, V)$.

$\mathsf{UpdateWitness}(wit_i, epoch_V, state_U)$ aborts with $\perp$, if $V \not\subset U$. Otherwise it parses wit_i as $(\sigma_i, u_i, g_i, w, V_w)$. Let $w' = w \frac{\prod_{j \in V \setminus V_w} g_{n+1-j+i}}{\prod_{j \in V_w \setminus V} g_{n+1-j+i}}$. The algorithm outputs $wit_i' = (\sigma_i, u_i, g_i, w', V)$.

[4] Note that the discrete logarithms of $g, h, \tilde{h}$ and u with respect to each other are mutually unknown.

[5] Both $epoch_{V \cup \{i\}}$ and $state_{U \cup \{i\}}$ could be signed by the issuer to prevent proliferation of fake accumulators.

Proof protocol. We now show a protocol that allows a user to prove possession of an unrevoked (and updated) credential $\mathit{cred}_i = (\sigma, c, m_1, \ldots, m_\ell, g_i, s, i)$ using $\mathit{wit}_i = (\sigma_i, g_i, u_i, w, V_w)$. The common input of the protocol is the issuer's public key pk_I, the epoch information epoch_V, and a specification of the proof statement (this includes the information revealed about the certificate). In the example below the user chooses to keep the first ℓ' messages secret while he reveals the rest of the messages.

The user (as prover) picks $\rho, \rho', r, r', r'', r''' \leftarrow \mathbb{Z}_q$, and picks opening open, $\mathit{open}' \leftarrow \mathbb{Z}_q$ to commit to ρ and r respectively. He computes commitments $C = h^\rho \tilde{h}^{\mathit{open}}$, $D = g^r \tilde{h}^{\mathit{open}'}$ and blinded values $A = \sigma \tilde{h}^\rho$, $\mathcal{G} = g_i \tilde{h}^r$, $\mathcal{W} = w\tilde{h}^{r'}$, $\mathcal{S} = \sigma_i \tilde{h}^{r''}$, and $\mathcal{U} = u_i \tilde{h}^{r'''}$. The user sends $C, D, A, \mathcal{G}, \mathcal{W}, \mathcal{S}$ and $\mathcal{U}$ to the verifier and engages the verifier in the following proof:

$$PK\{(c, \rho, \mathit{open}, \mathit{mult}, \mathit{tmp}, m_1, \ldots, m_{\ell'}, s, r, \mathit{open}', \mathit{mult}', \mathit{tmp}', r', r'', r''') :$$

$$C = h^\rho \tilde{h}^{\mathit{open}} \ \wedge\ 1 = C^c h^{-\mathit{mult}} \tilde{h}^{-\mathit{tmp}} \ \wedge \tag{1}$$

$$\frac{e(h_0 \cdot \prod_{j=\ell'+1}^{\ell} h_j^{m_j} \cdot \mathcal{G}, h)}{e(A, y)} = e(A, h)^c \cdot e(\tilde{h}, h)^r \tag{2}$$

$$\cdot\, e(\tilde{h}, y)^{-\rho} \cdot e(\tilde{h}, h)^{-\mathit{mult}} \cdot \prod_{j=1}^{\ell'} e(h_j, h)^{-m_j} \cdot e(h_{\ell+1}, h)^{-s} \wedge$$

$$\frac{e(\mathcal{G}, \mathit{acc}_V)}{e(g, \mathcal{W})z} = e(\tilde{h}, \mathit{acc}_V)^r e(1/g, \tilde{h})^{r'} \wedge \tag{3}$$

$$D = g^r \tilde{h}^{\mathit{open}'} \ \wedge\ 1 = D^c g^{-\mathit{mult}'} \tilde{h}^{-\mathit{tmp}'} \ \wedge \tag{4}$$

$$\frac{e(pk \cdot \mathcal{G}, \mathcal{S})}{e(g, g)} = e(pk \cdot \mathcal{G}, \tilde{h})^{r''} e(\tilde{h}, \tilde{h})^{-\mathit{mult}'} e(\tilde{h}, \mathcal{S})^r \wedge \tag{5}$$

$$\frac{e(\mathcal{G}, u)}{e(g, \mathcal{U})} = e(\tilde{h}, u)^r e(1/g, \tilde{h})^{r'''}\} \ . \tag{6}$$

This proof merges the proof of knowledge of Section 3.3 with a proof of knowledge of an adapted signature as the ones described in Section 4.1. The latter is similar to the proof of knowledge of a signature in Section 2.2. Special care needs to be taken to bind the g_i in the accumulator to the g_i value in the adapted signature.

Theorem 2. *Under the n-HSDHE and the n-DHE assumptions, the protocol above is a proof of knowledge of an adapted signature on* $(m_1, \ldots, m_\ell, g_i)$ *such that* $i \in V$. The proof can be found in Section A.1.

5 Conclusion and Discussion

In this paper we have introduced a novel dynamic accumulator based on bilinear maps and have shown how it can be used to achieve efficient revocation in privacy-preserving systems such as group signatures or anonymous credential systems.

Previous proposals require expensive computations for updating witnesses and are not suitable for electronic token based systems with a large number of users, as the ones that will soon appear with the introduction of e-ID's, e-tickets and alike. Our accumulator overcomes the aforementioned drawback introducing efficient witness updates. In the envisioned system, at the beginning of each epoch, the users retrieve their currently valid witness from an updating authority (as the number of revocation per epoch is likely to be very large, the users will typically not be able to handle them). As updating a witness in our scheme requires only a number of multiplication linear in the number of changes to the accumulator (in particular, linear in $|(V \setminus V_w) \cup (V_w \setminus V)|$) a single authority (which not necessarily needs to be the issuer) can keep the witness values for all users easily up-to-date (and in main memory). This is a key feature that enables the adoption of dynamic accumulators for revocation in privacy-preserving systems with large number of users as, e.g., in the case of electronic driving license systems. Although not necessary, there could even be several witness update entities, responsible for upgrading witnesses for groups of users. For example, in a national e-ID's systems, witness updates could be performed by per-county or per-city witness update entity. The latter requires only public parameters and are only responsible for correct computation of the witness updates for the users in their group. Malicious behavior by one of the witness update entities, does not break system security (recall that they only require public parameters) but can only lead to denial of service. That is, if a witness is not correctly computed (not reflecting the latest changes in the accumulator) it would prevent a user to prove validity of her credential. In this case, users can report to the issuing authority to obtain a valid witness update and signal the misbehaving of the witness update entity.

Acknowledgements

During this work, we enjoyed many discussion with Thomas Gross and Tom Heydt-Benjamin on various kinds of revocation of anonymous credentials. Thank you! The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 216483.

References

1. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. Technical Report Research Report RZ 3419, IBM Research Division (May 2002)
2. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. Technical Report Research Report RZ 3295, IBM Research Division (November 2000)
3. Persiano, G., Visconti, I.: An efficient and usable multi-show non-transferable anonymous credential system. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 196–211. Springer, Heidelberg (2004)

4. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Commun. ACM 28(10), 1030–1044 (1985)
5. Chaum, D., Evertse, J.H.: A secure and privacy-protecting protocol for transmitting personal information between organizations. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 118–167. Springer, Heidelberg (1987)
6. Chen, L.: Access with pseudonyms. In: Dawson, E.P., Golić, J.D. (eds.) Cryptography: Policy and Algorithms 1995. LNCS, vol. 1029, pp. 232–243. Springer, Heidelberg (1996)
7. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
8. Okamoto, T.: An efficient divisible electronic cash scheme. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 438–451. Springer, Heidelberg (1995)
9. Chan, A.H., Frankel, Y., Tsiounis, Y.: Easy come - easy go divisible cash. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 561–575. Springer, Heidelberg (1998)
10. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: [40], pp. 302–321
11. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: [41], pp. 132–145
12. Bangerter, E., Camenisch, J., Lysyanskaya, A.: A cryptographic framework for the controlled release of certified data. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2004. LNCS, vol. 3957, pp. 20–42. Springer, Heidelberg (2006)
13. Ateniese, G., Song, D.X., Tsudik, G.: Quasi-efficient revocation in group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
14. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: [41], pp. 168–177
15. Nakanishi, T., Funabiki, N.: Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. IEICE Transactions 90-A(1), 65–74 (2007)
16. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: [42], pp. 41–55
17. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
18. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
19. West Virginia Department of Transportation, Division of Motor Vehicles : Wvdmv fy 2005 annual report (2005), http://www.wvdot.com/6_motorists/dmv/downloads/DMVAnnualReport2005.pdf
20. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
21. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
22. Wang, P., Wang, H., Pieprzyk, J.: A new dynamic accumulator for batch updates. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 98–112. Springer, Heidelberg (2007)

23. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: [40], pp. 440–456
24. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
25. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
26. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)
27. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
28. Camenisch, J.L., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
29. Camenisch, J.L.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich, Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz (1998)
30. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997)
31. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
32. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich (March 1997)
33. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: [42], pp. 56–72
34. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
35. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic -taa. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
36. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. Cryptology ePrint Archive, Report 2008/634 (2008)
37. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
38. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
39. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
40. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)

41. Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.): Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washingtion, DC, USA, October 25-29, 2004. ACM, New York (2004)
42. Franklin, M. (ed.): CRYPTO 2004. LNCS, vol. 3152. Springer, Heidelberg (2004)

A Proofs

A.1 Proof of Theorem 1

It is standard to show that from a convincing prover of the protocol

$$PK\{(r, r', r'', r''', open, mult, tmp) :$$
$$D = g^r \tilde{h}^{open} \wedge 1 = D^{r''} g^{-mult} \tilde{h}^{-tmp} \wedge \quad (7)$$
$$\frac{e(pk \cdot \mathcal{G}, \mathcal{S})}{e(g, g)} = e(pk \cdot \mathcal{G}, \tilde{h})^{r''} e(\tilde{h}, \tilde{h})^{-mult} e(\tilde{h}, \mathcal{S})^r \wedge \quad (8)$$
$$\frac{e(\mathcal{G}, acc_V)}{e(g, \mathcal{W})z} = e(\tilde{h}, acc_V)^r e(1/g, \tilde{h})^{r'} \wedge \quad (9)$$
$$\frac{e(\mathcal{G}, u)}{e(g, \mathcal{U})} = e(\tilde{h}, u)^r e(1/g, \tilde{h})^{r'''}\} . \quad (10)$$

one can with overwhelming probability extract values r, r', r'', r''', and $mult$ such that the Equations (9), (8), (10) hold. From Equation (9) we learn through simple transformation that $\frac{e(\mathcal{G}\tilde{h}^{-r}, acc_V)}{e(g, \mathcal{W}\tilde{h}^{-r'})} = z$. We distinguish three cases: In the first case $\mathcal{G}\tilde{h}^{-r}$ corresponds to a g_i in $state_U$ and $i \in V$. In this case the extraction was successful.

In the second case $\mathcal{G}\tilde{h}^{-r}$ corresponds to a g_i in $state_U$ but $i \notin V$. In this case we can use a successful prover to break the n-DHE assumption. The reduction obtains as input the parameters of a bilinear map $params_{BM} = (q, G, G_T, e, g)$, and an instance of the n-DHE assumption $(g_1, g_2, \dots, g_n, g_{n+2}, \dots, g_{2n}) \in G^{2n-1}$. It provides the prover with $pk_A = (params_{BM}, pk, z = e(g_1, g_n))$, acc_V and $state_U = (U, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n})$. The reduction computes the additional setup for the proof using a fresh sk. Given a successful prover it can extract r, r', r'', r''', and $mult$ such that

$$e(\mathcal{G}\tilde{h}^{-r}, acc_V) = e(g, \mathcal{W}\tilde{h}^{-r'})z$$

and

$$e(g, \prod_{j \in V} g_{n+1-j+i}) = e(g, \mathcal{W}\tilde{h}^{-r'} g_{n+1}) .$$

This means that

$$g_{n+1} = \frac{\prod_{j \in V} g_{n+1-j+i}}{\mathcal{W}\tilde{h}^{-r'}} .$$

For $i \in \{1, \dots, n\} \setminus V$, all $g_{n+1-j+i}$ are contained in $state_U$ and it is possible to compute this value. This breaks the n-DHE assumption (Consult also the proof in Section 3.2).

In the third case $\mathcal{G}\tilde{h}^{-r}$ does not correspond to a g_i in *state*$_U$. We will show that we can use such a prover to break the dedicated signature scheme (more concretely the n-HSDHE assumption) using the remaining Equations (8) and (6). The reduction works as follows. On input a HSDHE instance $(g, g^x, u, \{g^{1/(x+\gamma^i)}, g^{\gamma^i}, u^{\gamma^i}\}_{i=1\ldots n}, \{g^{\gamma^i}\}_{i=n+2\ldots 2n})$, the reduction uses the g^{γ^i} to build *state*$_U$ and the remaining values to construct the additional setup for the proof by setting $pk = g^x$ (and implicitly $sk = x$).

After extracting r, r', r'', r''', *open*, *mult* and *tmp* note that (based on Equation (7)) $mult = rr''$ and $tmp = openr''$ (or one can compute $\log_g h$ which would in turn allow us to break n-HSDHE). After obtaining a $\mathcal{G}\tilde{h}^{-r}$ that does not correspond to a value in $\{g^{\gamma^i}\}_{i=1\ldots n}$ it is easy to see from Equation (8) that $e(pk\mathcal{G}\tilde{h}^{-r}, \mathcal{S}\tilde{h}^{-r''}) = 1$. Let $c = \log_g \mathcal{G}\tilde{h}^{-r}$. then $\mathcal{S}\tilde{h}^{-r''} = g^{1/(x+c)}$. Similarly from Equation (10) we learn that $\frac{e(\mathcal{G}\tilde{h}^{-r},u)}{e(g,\mathcal{U}\tilde{h}^{-r'''})} = 1$. If $\mathcal{G}\tilde{h}^{-r} = g^c$, then $\mathcal{U}\tilde{h}^{-r'''} = u^c$. This contradicts the n-HSDHE assumption.

As the malicious prover has no way to distinguish between the first or the second reduction (as well as the real setup), we can randomly pick one of the two reductions to break either the n-DHE or the n-HSDHE assumption (we only loose a factor of 1/2 in the tightness of the reduction). □

A.2 Proof of Theorem 2

We extract the value from the above proof. From Equation (1) we know that if $mult \neq \rho c$ or $mult' \neq rr''$, we can compute the discrete logarithm $\log_g h$. This contradicts the DL assumption.

From Equations (3,4,5,6) and the security of the accumulator proof protocol in Section 3.3 we know that $\mathcal{G}\tilde{h}^{-r}$ equals a g_i, $i \in V$, such that $\mathcal{W}\tilde{h}^{-r'}$ is a verifying accumulator witness for this value. Otherwise we break the n-DHE or the n-HSDHE assumption. (The reductions would be set up in the same way as in Appendix A.1.)

Now we consider Equation (2) of the proof. It asserts the prover's knowledge of values $m_1, \ldots, m'_\ell$ such that

$$e(h_0 \cdot \mathcal{G} \cdot (\prod_{j=q}^{\ell'} h_j^{m_j}) \cdot (\prod_{j=\ell'+1}^{\ell} h_j^{m_j}), g) e(\tilde{h}, y)^\rho \cdot e(\tilde{h}, g)^{\rho c} \cdot e(h_{\ell+1}, g)^s = \\ e(A, y) e(A, g)^c \cdot e(\tilde{h}, g)^r \ .$$

Here we have made use of the relation $mult = \rho c$. By simplifying this equation further we obtain

$$e(h_0 \cdot \prod_{j=q}^{\ell'} h_j^{m_j} \cdot \prod_{j=\ell'+1}^{\ell} h_j^{m_j} \cdot {h_{\ell+1}}^s \cdot \mathcal{G}/\tilde{h}^r, g) = e(A/\tilde{h}^\rho, yg^c) \ .$$

This shows that $(A/\tilde{h}^\rho, c, s)$ is a valid adapted signature for $(m_1, \ldots, m_\ell, \tilde{g}^{\gamma^i})$. □

Controlling Access to an Oblivious Database Using Stateful Anonymous Credentials

Scott Coull, Matthew Green, and Susan Hohenberger

The Johns Hopkins University
Information Security Institute
3400 N. Charles Street; Baltimore, MD 21218, USA
{coulls,mgreen,susan}@cs.jhu.edu

Abstract. In this work, we consider the task of allowing a content provider to enforce complex access control policies on oblivious protocols conducted with anonymous users. As our primary application, we show how to construct privacy-preserving databases by combining oblivious transfer with an augmented anonymous credential system. This permits a database operator to restrict which items each user may access, without learning anything about users' identities or item choices. This strong privacy guarantee holds even when users are assigned different access control policies and are allowed to adaptively make many queries. To do so, we show how to augment existing anonymous credential systems so that, in addition to certifying a user's attributes, they also store state about the user's database access history. Our construction supports a wide range of access control policies, including efficient and private realizations of the Brewer-Nash (Chinese Wall) and Bell-LaPadula (Multilevel Security) policies, which are used for financial and defense applications. In addition, our system is based on standard assumptions in the standard model and, after an initial setup phase, each transaction requires only constant time.

1 Introduction

There is an increasing need to provide privacy to users accessing sensitive information, such as medical or financial data. The mere fact that a rare disease specialist accesses a certain patient's medical record exposes information about the private contents of the record. At the same time, newly developed regulations governing such sensitive data (*e.g.,* Sarbanes-Oxley, HIPAA) require content providers to enact strict accounting procedures. These may seem like conflicting goals since the specialist may wish to hide which patient's record she is requesting while the database operator may wish to ensure that the doctor's collective accesses do not violate regulations. The situation becomes even more precarious when a patient uses such a database to look up information about a potentially sensitive medical condition. In such cases, the patient's identity, as well as her access patterns, must remain hidden from the database administrator. The increasing trend toward outsourcing and distributing sensitive databases,

S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 501–520, 2009.

such as the outsourced medical database provided by Google Health [27], makes these concerns all the more compelling.

Previous works have proposed to construct privacy-friendly databases using Private Information Retrieval [20] or Oblivious Transfer [30,15]. In a k-out-of-N Oblivious Transfer protocol, a content provider with messages $M_1, \ldots, M_N$ and a user with indices $\sigma_1, \ldots, \sigma_k \in [1, N]$ interact in such a way that at the end the user obtains $M_{\sigma_1}, \ldots, M_{\sigma_k}$ without learning anything about the other messages and the provider does not learn anything about $\sigma_1, \ldots, \sigma_k$. This tool leads to privacy-friendly databases *when the user gets her choice of any files with no restrictions.* Unfortunately, that scenario rules out many practical database applications. Worse, the previous work in this area provides no insight as to how access control might ever be incorporated into such a database, since traditional access control mechanisms assume knowledge of the items being requested.

Thus, to realize a practical "oblivious database" for our users, we must couple it with enforceable access controls. We make three design choices that act as guiding principles for our system. Our first is to maintain all anonymity and privacy guarantees provided by the oblivious transfer protocol. We reject any solutions that use pseudonyms or allow for some form of transaction linking, since it is too difficult to infer what compromise to privacy might result. Secondly, we wish to enforce a strong notion of access control where the database operator may limit each access based on the user's identity, item requested, and even a history of the user's previous requests. Finally, we require our solution to be efficient, and thus each transaction should take constant time regardless of a user's access history, or the complexity of the access policy which she must follow.

Contributions. To achieve the goals above, we show how to efficiently couple an adaptive, oblivious transfer protocol with an anonymous credential scheme [18,11], to provide non-trivial, real-world access controls for oblivious databases. Specifically, we present an extension to existing anonymous credential systems to support history-dependent access controls by embedding the user's current state into the credential, and dynamically updating that state according to well-defined policies governing the user's actions. These *stateful anonymous credentials* are built on top of well-known signatures with efficient protocols [29,11,12,4]. Our constructions are secure in the standard model under basic assumptions, such as Strong RSA. Additionally, we introduce a technique for efficiently proving that a committed value lies in a hidden range that is unknown to the verifier, which may be of independent interest.

Our constructions can be used to achieve non-trivial access control policies, including the Brewer-Nash (Chinese Wall) [7] and Bell-LaPadula (Multilevel Security) [2] model, which are used in a number of settings, including financial institutions and classified government systems. We discuss simulation-based security definitions for our stateful anonymous credentials, as well as an anonymous and oblivious database system with access controls.

Related Work. Several previous works sought to limit user actions while maintaining privacy, either directly within an existing protocol or through the use of anonymous credentials. Aiello, Ishai, and Reingold [1] proposed priced oblivious transfer, in which each user is given a declining balance that can be spent on each transfer. However, here user anonymity is not protected, and the protocol is also vulnerable to selective-failure attacks in which a malicious server induces faults to deduce the user's selections [30,15]. The more general concept of conditional oblivious transfer was proposed by Di Crescenzo, Ostrovsky, and Rajagopolan [23] and subsequently strengthened by Blake and Kolesnikov [3]. In conditional oblivious transfer, the sender and receiver maintain private inputs (x and y, respectively) to some publicly known predicate $q(\cdot,\cdot)$ (*e.g.,* the greater than equal to relation on integers). The items in the oblivious transfer scheme are encrypted such that the receiver can complete the oblivious transfer and recover her data if and only if $q(x, y) = 1$. In addition, techniques from e-cash and anonymous credentials have been used to place simple limitations on an anonymous user's actions, such as preventing a user from logging in more than once in a given time period [8], authenticating anonymously at most k times [34], or preventing a user from exchanging too much money with a single merchant [9]. Rather than providing a specific type of limitation or restricting the limitation to a particular protocol, our proposed system instead provides a general method by which arbitrary access control policies can be added to a wide variety of anonymous and oblivious protocols.

2 Stateful Credentials: Model and Definitions

The goal of typical anonymous credential systems is to provide users with a way of proving certain attributes about themselves (*e.g.,* age, or height) without revealing their identity. Users conduct this proof by obtaining a credential from an organization, and subsequently "showing" the credential without revealing their identity. In addition to the properties of typical credentials, a stateful anonymous credential system adds the additional notion of credential state, which is embedded as an attribute within the credential. The user may update the state in her credential according to some well-defined *policy* dictated by the credential provider. In practice, this may limit the user to a finite number of states, or a particular sequential ordering of states. To maintain the user's anonymity, it is important that the update protocol not leak information about the credential's current state beyond what the user chooses to reveal.

At a high level, the stateful anonymous credential system, which is defined by the tuple of algorithms ($\mathsf{Setup}, \mathsf{ObtainCred}, \mathsf{UpdateCred}, \mathsf{ProveCred}$), operates as follows. First, the user and credential provider negotiate the use of a specified policy using the $\mathsf{ObtainCred}$ protocol. The negotiated policy determines the way in which the user will be allowed to update her credential. After the protocol completes, the user receives an anonymous credential that embeds her initial state in the policy, in addition to other attributes. Next, the user can prove (in zero-knowledge) that the credential she holds embeds a given state, or attribute, just as she would in other anonymous credential systems by using the $\mathsf{ProveCred}$ protocol. This allows the user anonymous access to services, while the entity checking the credential is assured of the user's attributes, as well as her state in the specified policy. These proof can be done in such a way that the verifying entity learns nothing about the user's state or attributes. Finally, when the user wishes to update her credential to reflect a change in her state, she interacts with the credential provider using the UpdateCred protocol, to prove (again, in zero-knowledge) her current state and the existence of a transition in the policy from her current state to her intended next state. As with the ProveCred protocol, the provider learns nothing about the user other than the fact that her state change is allowed by the policy previously negotiated within the ObtainCred protocol.

Policy Model. To represent the policies for our stateful credential system, we use directed graphs, which can be thought of as a state machine that describes the user's behavior over time. We describe the *policy graph* Π_{pid} as the set of *tags* of the form $(\mathsf{pid}, S \to T)$, where pid is the identity of the policy and $S \to T$ represents a directed edge from state S to state T. Thus, the user's credential embeds the identity of the policy pid and the user's current state in the policy graph. When the user updates her credential, she chooses a tag and then proves that the policy id she is following is the same as what is provided in the tag and that the tag encodes an edge from her current state to her desired next state.

These policy graphs can be created in such a way that the users may reach a terminal state, and therefore would be unable to continue updating (and consequently using) their credential. In this case, it may be possible for an adversary to perform traffic analysis to infer the policy that the user is following. To prevent this, we consider the use of *null transitions* in the graph. The null transitions occur as self-loops on the terminal states of the policy graph, and allow the user to update her credential as often as she wishes to prevent such traffic analysis attacks. However, the updates performed on these credentials only allow the user access to a predefined null resource. The specifics of this null resource are dependent on the anonymous protocol that the credential system is coupled with, and we describe an implementation for them in oblivious databases in Section 5.

While these policy graphs are rather simplistic, they can represent complicated policies. For instance, a policy graph can encode the user's history with respect to accessing certain resources up to the largest cycle in the graph. Moreover, we can extend the policy graph tags to include auxiliary information about the actions that the user is allowed to perform at each state. By doing so, we allow the graph to dynamically control the user's access to various resources according to her behavior and history, as well as her other attributes. In Section 5, we examine how to extend these policy graphs to provide non-trivial, real-world access control policies for oblivious databases, as well as a variety of other anonymous and oblivious application.

2.1 Protocol Descriptions and Definitions for Stateful Credentials

A stateful anonymous credential scheme consists of the four protocols: Setup, ObtainCred, UpdateCred, and ProveCred. We will now describe their input/output behavior and intended functionality. For the remainder of the paper, let 1^κ be the security parameter.

$\mathsf{Setup}(\mathcal{U}(1^k), \mathcal{P}(1^k, \Pi_1, \ldots, \Pi_n))$: The provider $\mathcal{P}$ generates parameters *params* and a keypair $(pk_{\mathcal{P}}, sk_{\mathcal{P}})$ for the credential scheme. For each graph Π to be enforced, $\mathcal{P}$ also generates a cryptographic representation $\Pi_{\mathcal{C}}$ and publishes this value via an authenticated channel. Each user $\mathcal{U}$ generates a keypair and requests that it be certified by a trusted CA.

$\mathsf{ObtainCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \Pi_{\mathcal{C}}), \mathcal{P}(pk_{\mathcal{U}}, sk_{\mathcal{P}}, \Pi_{\mathcal{C}}, S))$: $\mathcal{U}$ identifies herself to $\mathcal{P}$ and then receives her credential Cred which binds her to a policy graph Π and starting state S.

$\mathsf{UpdateCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}, T), \mathcal{P}(sk_{\mathcal{P}}, D))$: $\mathcal{U}$ and $\mathcal{P}$ interact such that Cred is updated from its current state to state T, but only if this transition is permitted by the policy Π. Simultaneously, $\mathcal{P}$ should not learn $\mathcal{U}$'s identity, attributes, or current state. To prevent replay attacks, $\mathcal{P}$ maintains a database D, which it updates as a result of the protocol.

$\mathsf{ProveCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}), \mathcal{P}(pk_{\mathcal{P}}, E))$: $\mathcal{U}$ proves possession of a credential Cred in a particular state. To prevent re-use of credentials, $\mathcal{P}$ maintains a database E, which it updates as a result of the protocol.

Security Definitions. Security definitions for anonymous credentials have traditionally been game-based. Unfortunately, the existing definitions may be insufficient for the applications considered in this work, as these definitions do not necessarily capture *correctness*. This can lead to problems when we integrate our credential system with oblivious transfer protocols (see *e.g.,* [30,15]). To capture the security requirements needed for our applications, we instead use a simulation-based definition, in which security of our protocols is analyzed with respect to an "ideal world" instantiation. We do not require security under concurrent executions, but rather restrict our analysis to atomic, sequential execution of each protocol. We do so because our constructions, which employ standard zero-knowledge techniques, require rewinding in their proof of security and thus are not concurrently secure. An advantage of the simulation paradigm is that our definitions will inherently capture correctness (*i.e.,* if parties honestly follow the protocols then they will each receive their expected outputs). Informally, the security of our system is captured by the following two definitions:

Provider Security: A malicious user (or set of colluding users) must not be able to falsely prove possession of a credential without first obtaining that credential, or arriving at it via an admissable sequence of credential updates. For our purposes, we require that the malicious user(s) cannot provide a proof of being in a state if that state is not present in her credential.

User Security: A malicious provider controlling some collection of corrupted users cannot learn any information about a user's identity or her state in the policy graph beyond what is available through auxiliary information from the environment.

Due to space considerations, we defer the formal security definitions for stateful anonymous credentials to full version of this paper [21]. In Section 5.1 and Appendix A, we provide definitions for oblivious databases with access control.

3 Technical Preliminaries

In this section, we recall some basic building blocks, and then introduce a new primitive, *hidden range proofs*, which may be of independent interest.

Pedersen and Fujisaki-Okamoto Commitments. In the Pedersen commitment scheme [32], the public parameters are a group $\mathbb{G}$ of prime order q, and generators $(g_0, \ldots, g_m)$. In order to commit to the values $(v_1, \ldots, v_m) \in \mathbb{Z}_q^m$, the user picks a random $r \in \mathbb{Z}_q$ and sets $C = \text{Commit}(v_1, \ldots, v_m; r) = g_0^r \prod_{i=1}^m g_i^{v_i}$. Fujisaki and Okamoto [26] provided a composite order variant.

Signatures with Efficient Protocols. Camenisch and Lysyanskaya (CL) [11] designed a signature scheme with two efficient protocols: (1) a protocol for a user to obtain a signature on the value(s) in a Pedersen (or Fujisaki-Okamoto) commitment [32,26] without the signer learning anything about the message(s), and (2) a proof of knowledge of a signature. Our constructions may be implemented with the Strong RSA signature scheme [11] (and with minor modifications, using bilinear signatures based on the LRSW assumption [12]). Both schemes consist of the algorithms $(\mathsf{CLKeyGen}, \mathsf{CLSign}, \mathsf{CLVerify})$, which we describe below:

$\mathsf{CLKeyGen}(1^\kappa)$. On input a security parameter, outputs a keypair (pk, sk).
$\mathsf{CLSign}(sk, M_1, \ldots, M_n)$. On input one or more messages and a secret signing key, outputs the signature σ.
$\mathsf{CLVerify}(pk, \sigma, M_1, \ldots, M_n)$. On input a signature, message(s) and public verification key, outputs 1 if the signature verifies, 0 otherwise.

We could also use other bilinear signatures with efficient protocols (*e.g.,* [4]), though we do not make use of these in our construction.

Zero-Knowledge Protocols. We use several standard results for proving statements about discrete logarithms, such as (1) a proof of knowledge of a discrete logarithm modulo a prime [33] or a composite [26,24], (2) a proof of knowledge of equality of representation modulo two (possibly different) prime [19] or composite [14] moduli, (3) a proof that a commitment opens to the product of two other committed values [13,16,6], and (4) a proof of the disjunction or conjunction of any two of the previous [22]. These composite-based protocols are secure under Strong RSA and the prime-based ones under the discrete logarithm assumption.

Note that there are several building blocks that are not used in our basic scheme, but which can be used to provide extended functionality or improved performance. These building blocks include:

Bilinear Groups. Let $\mathsf{BMsetup}$ be an algorithm that, on input 1^κ, outputs the parameters for a bilinear mapping as $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e, g \in \mathbb{G})$, where g generates $\mathbb{G}$, the groups $\mathbb{G}, \mathbb{G}_T$ each have prime order p, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

Hidden-Range Proofs. Standard techniques [17,13,13,5] allow us to efficiently prove that a committed value lies in a *public* integer interval (*i.e.,* where the interval is known to both the prover and verifier). In our protocols, it is useful to *hide* this interval from the verifier, and instead have the prover show that a committed value lies between the openings of two other commitments.

Fortunately, this can be done efficiently as follows. Suppose we wish to show that $a \leq j \leq b$, for positive numbers a, j, b without revealing them. This is equivalent to showing that $0 \leq (j - a)$ and $0 \leq (b - j)$. We only need to get these two sums reliably into commitments, and can then employ the standard techniques since the range (≥ 0) is now public. Using a group $\mathbf{G} = \langle \mathbf{g} \rangle$, where n is a special RSA modulus, $\mathbf{g}$ is a quadratic residue modulo n and $\mathbf{h} \in \mathbf{G}$. The prover commits to these values as $A = \mathbf{g}^a \mathbf{h}^{r_a}$, $J = \mathbf{g}^j \mathbf{h}^{r_j}$, and $B = \mathbf{g}^b \mathbf{h}^{r_b}$, for random values $r_a, r_j, r_b \in \{0, 1\}^\ell$ where ℓ is a security parameter. The verifier next computes a commitment to $(j - a)$ as J/A and to $(b - j)$ as B/J. The prover and verifier then proceed with the standard public interval proofs with respect to these commitments, which for technical reasons require groups where Strong RSA holds.

4 Stateful Anonymous Credentials

In this section, we describe how to realize stateful credentials. The state records information about the user's attributes as well as her prior access history. We will consider two separate modes for "showing" a credential. In the first mode, the user exposes her portions of her state during the ProveCred protocol. This is useful for, say, a DRM application where the user's goal is to prove that her software is in a "licensed" state without revealing her name. In mode two, the user uses her credential to gain access to resources *without* revealing her state through the use of zero knowledge proofs. Specifically, we show how to tie this credential system to protocols, such as adaptive oblivious transfer, where the user wants to hide both her identity and the item she is requesting while simultaneously proving that she has the credentials to obtain the item.

4.1 Basic Construction

Our construction begins with the anonymous credentials of Camenisch and Lysyanskaya [29,11,12], where the state is embedded as a field in the signature. The core innovation here is a protocol for performing state updates, and a technique for "translating" a history-dependent update policy into a cryptographic representation that can be used as an input to this protocol.

The setup, credential granting, and credential update protocols are presented in Figure 1. We will now briefly describe the intuition behind them.

Setup. First, the credential provider $\mathcal{P}$ generates its keypair and identifies one or more access policies it wishes to enforce. Each policy — encoded as a graph — may be applied to one or more users. The provider next "translates" the graph into a cryptographic representation which consists of the graph description, and a separate CL signature for each tag in the graph. Recall from Section 2 that

$\mathsf{Setup}(\mathcal{U}(1^k), \mathcal{P}(1^k, \Pi_1, \ldots, \Pi_n))$**:** The provider $\mathcal{P}$ generates parameters for the CL signature, as well as for the Pedersen commitment scheme.

Party $\mathcal{P}$ runs $\mathsf{CLKeyGen}$ twice, to create the CL signature keypairs $(spk_{\mathcal{P}}, ssk_{\mathcal{P}})$ and $(gpk_{\mathcal{P}}, gsk_{\mathcal{P}})$. It retains $(pk_{\mathcal{P}}, sk_{\mathcal{P}}) = ((spk_{\mathcal{P}}, gpk_{\mathcal{P}}), (ssk_{\mathcal{P}}, gsk_{\mathcal{P}}))$ as its keypair. The provider's public key $pk_{\mathcal{P}}$ must be certified by a trusted CA.

Each party $\mathcal{U}$ selects $u \xleftarrow{\$} \mathbb{Z}_q$ and computes the keypair $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) = (g^u, u)$. The user's public key $pk_{\mathcal{U}}$ must be certified by a trusted CA.

Next, for each policy graph Π, $\mathcal{P}$ generates a cryptographic representation $\Pi_{\mathcal{C}}$.

1. $\mathcal{P}$ parses Π to obtain a unique policy identifier pid.
2. For each tag $t = (\mathsf{pid}, S, T)$ in Π, $\mathcal{P}$ computes a signature $\sigma_{S \to T} \leftarrow \mathsf{CLSign}(gsk_{\mathcal{P}}, (\mathsf{pid}, S, T))$.
3. $\mathcal{P}$ sets $\Pi_{\mathcal{C}} \leftarrow \langle \Pi, \forall t : \sigma_{S \to T} \rangle$ and publishes this value via an authenticated channel.

$\mathsf{ObtainCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \Pi_{\mathcal{C}}), \mathcal{P}(pk_{\mathcal{U}}, sk_{\mathcal{P}}, \Pi_{\mathcal{C}}, S))$**:** On input a graph Π and initial state S, $\mathcal{U}$ first obtains $\Pi_{\mathcal{C}}$. $\mathcal{U}$ and $\mathcal{P}$ then conduct the following protocol:

1. $\mathcal{U}$ picks random usage and update nonces $N_s, N_u \in \mathbb{Z}_q$ and computes $A \leftarrow \mathrm{Commit}(sk_{\mathcal{U}}, N_s, N_u)$.
2. $\mathcal{U}$ conducts an interactive proof to convince $\mathcal{P}$ that A correlates to $pk_{\mathcal{U}}$.
3. $\mathcal{U}$ and $\mathcal{P}$ run the CL signing protocol on committed values so that $\mathcal{U}$ obtains the state signature $\sigma_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_{\mathcal{P}}, (sk_{\mathcal{U}}, N_s, N_u, \mathsf{pid}, S))$ with pid, S contributed by $\mathcal{P}$.
4. $\mathcal{U}$ stores the credential $\mathsf{Cred} = (\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s, N_u)$.

$\mathsf{UpdateCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}, T), \mathcal{P}(sk_{\mathcal{P}}, D))$**:** Given a credential Cred currently in state S, $\mathcal{U}$ and $\mathcal{P}$ interact to update the credential to state T:

1. $\mathcal{U}$ parses $\mathsf{Cred} = (\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s, N_u)$ and identifies a signature $\sigma_{S \to T}$ in $\Pi_{\mathcal{C}}$ that corresponds to a transition from state S to T (if none exists, $\mathcal{U}$ aborts).
2. $\mathcal{U}$ selects $N'_s, N'_u \xleftarrow{\$} \mathbb{Z}_q$ and computes $A \leftarrow \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, N'_u, \mathsf{pid}, T)$.
3. $\mathcal{U}$ sends (N_u, A) to $\mathcal{P}$. $\mathcal{P}$ looks in the database D for a pair $(N_u, A' \neq A)$. If no such pair is found, then $\mathcal{P}$ adds (N_u, A) to D. Otherwise $\mathcal{P}$ aborts.
4. $\mathcal{U}$ proves to $\mathcal{P}$ knowledge of values $(sk_{\mathcal{U}}, \mathsf{pid}, S, T, N'_s, N'_u, N_s, \sigma_{\mathsf{state}}, \sigma_{S \to T})$ such that:
   (a) $A = \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, N'_u, \mathsf{pid}, T)$.
   (b) $\mathsf{CLVerify}(spk_{\mathcal{P}}, \sigma_{\mathsf{state}}, (sk_{\mathcal{U}}, N_s, N_u, \mathsf{pid}, S)) = 1$.
   (c) $\mathsf{CLVerify}(gpk_{\mathcal{P}}, \sigma_{S \to T}, (\mathsf{pid}, S, T)) = 1$
5. If these proofs do not verify, $\mathcal{P}$ aborts. Otherwise $\mathcal{U}$ and $\mathcal{P}$ run the CL signing protocol on committed values to provide $\mathcal{U}$ with $\sigma'_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_{\mathcal{P}}, A)$.
6. $\mathcal{U}$ stores the updated credential $\mathsf{Cred}' = (\Pi_{\mathcal{C}}, T, \sigma'_{\mathsf{state}}, N'_s, N'_u)$.

Fig. 1. Basic algorithms for obtaining and updating a stateful anonymous credential

the tags embed the graph id, start, and end states. The cryptographic policy representations are distributed to users via an authenticated broadcast channel (*e.g.,* by signing and publishing them on a website). The user $\mathcal{U}$ generates a keypair that is certified by the CA.

Obtaining a Credential. When a user $\mathcal{U}$ wishes to obtain a credential, she first negotiates with the provider to select an update policy to which the credential will be bound, as well the credential's initial state within the policy graph. The user next engages in a protocol to blindly extract a CL signature under the provider's secret key, which binds the user's public key, her initial state, the policy id, and two random nonces chosen by the user. The *update nonce* N_u is revealed when the user updates the credential and the *usage nonce* N_s is revealed when the user show's her credential. This signature, as well as the nonce and state information, form the credential. While the protocol for obtaining a credential, as currently described, reveals the user's identity through the use of her public key, we can readily apply the techniques found in [10,11] to provide a randomized pseudonym rather than the public key.

Updating the Credential's State. When the user wishes to update a credential, she first identifies a valid tag within the credential's associated policy. She then generates a new pair of nonces and a commitment embedding these values, as well as the new state from her chosen tag. Next, the user sends the update nonce from her current credential, along with the commitment, to the provider. The provider records this nonce and the commitment into a database — however, if the nonce is already in the database but associated with a *different* commitment, the provider aborts the protocol, which prevents the user from re-using an old version of a credential. By recording the nonce and commitment together, we allow the user to restart the protocol if it has failed as long as she uses the same commitment. If the nonce and commitment are not in the database, the user and provider then interact to conduct a zero-knowledge proof that: (1) the remainder of the information in the commitment is identical to the current credential, (2) the user has knowledge of the secret key corresponding to her current credential, and (3) the policy graph contains a signature on a tag from the current state to the new state. If these conditions are met, the user obtains a new credential embedding the new state.

Showing (or Privately Proving Possession of) a Credential. The approach to using a single-show credential, shown in Figure 2, follows [11,12]. When a user wishes to prove possession of a credential to $\mathcal{P}$, she first reveals the credential usage nonce and the current state of the credential. $\mathcal{P}$ must check that the usage nonce has not been used before. The user then proves knowledge of: (1) a CL signature embedding this state value and nonce formed under $\mathcal{P}$'s public key, and (2) a secret key that is consistent with the CL signature. Alternatively, if the user does not want to reveal her state explicitly, the user may generate a commitment to her state and prove (in zero knowledge) that it is the same as that which is found in her credential.

Single-show vs. multi-show. This is an example of a single-show credential. It can be shown only once, or the verifier will recognize the repeated usage nonce. To restore its anonymity, the user may return to $\mathcal{P}$ and execute the update protocol to replace the usage nonce, assuming it is allowed by the user's policy. This update policy gives users a way to use a single credential multiple times. One can also adapt this scheme to support k-times anonymous use of the credential by using the Dodis-Yampolskiy [25] pseudorandom function to generate the nonces from a common seed, as shown in [8].

A Note on Efficiency. The efficiency of our protocols is of utmost importance in ensuring their practical use in oblivious databases. During the Setup protocol, the provider must "translate" each of the graphs into a cryptographic representation by signing each tag associated with the graphs. This means that the complexity of the Setup protocol is linear in the size of the policy graphs used in controlling access to the database. While this may seem onerous at first, it is important to emphasize that this process may be conducted offline, and only as a one time cost to the provider. Once the setup procedure is completed, the complexity of the remaining protocols is *constant* and independent of the size of the policy in use since they deal with only a single tag at a time. Thus, our scheme is practical even for extremely complex policies containing thousands of distinct states and transition rules.

$\mathsf{ProveCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}), \mathcal{P}(pk_{\mathcal{P}}, E))$**:** User $\mathcal{U}$ proves knowledge of the Cred as follows:

1. $\mathcal{U}$ parses Cred as $(\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s, N_u)$, and sends its usage nonce N_s to $\mathcal{P}$ (who aborts if $N_s \in E$).
2. Otherwise, $\mathcal{U}$ continues with either:
   – (mode one) Sending her current credential state S to $\mathcal{P}$ in the clear.
   – (mode two) Sending a commitment to S.
3. $\mathcal{U}$ then conducts an interactive proof to convince $\mathcal{P}$ that it possesses a CL signature σ_{state} embedding N_s, S, and that it has knowledge of the secret key $sk_{\mathcal{U}}$.
4. $\mathcal{P}$ adds N_s to E.

Fig. 2. Basic algorithm for proving knowledge of a single-show anonymous credential

Theorem 1. *When instantiated with the RSA (resp., bilinear) variant of CL signatures, the anonymous credential scheme above achieves user, and provider security under the strong RSA (resp., LRSW) assumption.*

The proof of Theorem 1 is in the full version of this work [21].

5 Oblivious Database Access Control

In this section, we show how stateful anonymous credentials can be used to control access to oblivious databases. Recall that an oblivious database permits users to request data items without revealing their item choices or their identities to the database operator (*e.g.,* where the item choices are sensitive).

Although we possess efficient building blocks such as k-out-of-N Oblivious Transfer (OT), little progress has been made towards the deployment of practical oblivious databases. In part, this is due to a fundamental tension with the requirements of a database operator to provide some form of access control. In this section, we show that it is possible to embed flexible, history-dependent access controls into an oblivious database without compromising the user's privacy. Specifically, we show how to combine our stateful anonymous credential system with an adaptive Oblivious Transfer protocol to construct a multi-user oblivious database that supports complex access control policies. We show how to efficiently couple stateful credentials with the recent standard-model adaptive OT scheme due to Camenisch, Neven and shelat [15]. Our stateful credentials can also be efficiently coupled with the adaptive OT of Green and Hohenberger [28].

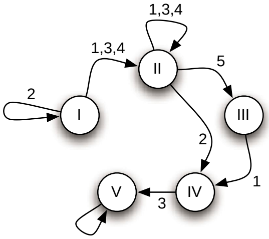


Fig. 3. Sample access policy for a small oblivious database. The labels on each transition correspond to the database item indices that can be requested when a user traverses the edge, with null transitions represented by unlabeled edges.

Linking Policies to Database Items. To support oblivious database access, we extend our policy graphs to incorporate *tags* of the form $(\mathsf{pid}, S \to T, i)$, where pid is the policy, $S \to T$ is the edge, and i is the message index in the database that is allowed by that tag. Each edge in the graph may be associated with one or more tags, which correspond to the items that can be obtained from the database when traversing that edge. As described in Section 2, we place null transitions on each terminal state that allow the user to update her credential and access a predefined null message. The set of all tags, both legitimate and null, are signed by the database and published.

Figure 3 shows an example policy for a small database. The interested reader can view a fuller discussion of the non-trivial access control policies, including Bell-Lapadula and Brewer-Nash that are allowed by our credential system in the full version of this work [21].

5.1 Protocol Descriptions and Security Definitions for Oblivious Databases

Our oblivious database protocols combine the scheme of Section 4.1 with a multi-receiver OT protocol. Each transaction is conducted between one of a collection of users and a single database server $\mathcal{D}$. We describe the protocol specifications.

- $\mathsf{Setup}(\mathcal{U}(1^k), \mathcal{D}(1^k, \Pi_1, \ldots, \Pi_n, M_1, \ldots, M_N))$**:** The database server $\mathcal{D}$ generates parameters *params* for the scheme. As in the basic credential scheme, it generates a cryptographic representation $\Pi_{\mathcal{C}}$ for each policy graph, and publishes those values via an authenticated channel. In addition, $\mathcal{D}$ initializes its database of messages according to the OT protocol in use. Each user $\mathcal{U}$ generates a keypair and requests that it be certified by a trusted CA.
- $\mathsf{OTObtainCred}(\mathcal{U}(pk_{\mathcal{D}}, sk_{\mathcal{U}}, \Pi_{\mathcal{C}}), \mathcal{D}(pk_{\mathcal{U}}, sk_{\mathcal{D}}, \Pi_{\mathcal{C}}, S))$**:** $\mathcal{U}$ registers with the system and receives a credential Cred which binds her to a policy graph Π_{id} and starting state S.
- $\mathsf{OTAccessAndUpdate}(\mathcal{U}(pk_{\mathcal{D}}, sk_{\mathcal{U}}, \mathsf{Cred}, t), \mathcal{D}(sk_{\mathcal{D}}, E))$**:** $\mathcal{U}$ requests an item at index i in the database from state S by selecting a tag $t = (\mathsf{pid}, S \to T, i)$ from the policy graph. The user then updates her credential Cred, in such a way that $\mathcal{D}$ does not learn her identity, her attributes, or her current state. Simultaneously, $\mathcal{U}$ obtains a message from the database at index i. At the end of a successful protocol, $\mathcal{U}$ updates the state information in Cred, and $\mathcal{D}$ updates a local datastore E.

Security. We informally describe the security properties of an oblivious database system, with a formal definition given in Appendix A.

Database Security: No (possibly colluding) subset of corrupted users can obtain any collection of items that is not specifically permitted by the users' policies.

User Security: A malicious database controlling some collection of corrupted users cannot learn any information about a user's identity or her state in the policy graph, beyond what is available through auxiliary information from the environment.

5.2 The Construction

In our model, many users share access to a single database server. To construct our protocols, we extend the basic credential scheme of Section 4.1 by linking it to the adaptive OT protocol of Camenisch *et al.* [15]. The database operator commits to a collection of N messages, along with a special *null* message at index $N + 1$. It them distributes these commitments (*e.g.,* via a website). Each user then registers with the database using the $\mathsf{OTObtainCred}$ protocol, and agrees to be bound by a policy that will control her ability to access the database.

To obtain items from the database, the user runs the $\mathsf{OTAccessAndUpdate}$ protocol, which proves (in zero knowledge) that its request is consistent with its policy. Provided the user does not violate her policy, the user is assured that the database operator learns nothing about its identity, or the nature of its request. Figures 4 and 5 describe the protocols in detail.

$\mathsf{Setup}(\mathcal{U}(1^k), \mathcal{D}(1^k, \Pi_1, \ldots, \Pi_n, M_1, \ldots, M_N))$**:** When the database operator $\mathcal{D}$ is initialized with a database of messages $M_1, \ldots, M_N$, it conducts the following steps:

1. $\mathcal{D}$ selects parameters for the OT scheme as $\gamma = (q, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMsetup}(1^\kappa)$, $h \xleftarrow{\$} \mathbb{G}$, $x \xleftarrow{\$} \mathbb{Z}_q$, and $H \leftarrow e(g, h)$. $\mathcal{D}$ generates two CL signing keypairs $(spk_\mathcal{D}, ssk_\mathcal{D})$ and $(gpk_\mathcal{D}, gsk_\mathcal{D})$, and $\mathcal{U}$ generates her keypair $(pk_\mathcal{U}, sk_\mathcal{U})$ as in the credential Setup protocol of Figure 1.
2. For $i = 1$ to $(N+1)$, $\mathcal{D}$ computes a ciphertext $C_i = (A_i, B_i)$ as:
   (a) If $i \leq N$, then $A_i = g^{\frac{1}{x+i}}$ and $B_i = e(h, A_i) \cdot M_i$.
   (b) If $i = (N+1)$, compute A_i as above and set $B_i = e(h, A_i)$.
3. For every graph Π to be enforced, $\mathcal{D}$ generates a cryptographic representation $\Pi_\mathcal{C}$ as follows:
   (a) $\mathcal{D}$ parses Π to obtain a unique policy identifier pid.
   (b) For each tag $t = (\mathsf{pid}, S, T, i)$ with $i \in [1, N+1]$, $\mathcal{D}$ computes the signature $\sigma_{S \to T, i} \leftarrow \mathsf{CLSign}(gsk_\mathcal{P}, (\mathsf{pid}, S, T, i))$. Finally, $\mathcal{D}$ sets $\Pi_\mathcal{C} \leftarrow \langle \Pi, \forall t : \sigma_{S \to T, i} \rangle$.
4. $\mathcal{D}$ sets $pk_\mathcal{D} = (spk_\mathcal{D}, gpk_\mathcal{D}, \gamma, H, y = g^x, C_1, \ldots, C_n)$ and $sk_\mathcal{D} = (ssk_\mathcal{D}, gsk_\mathcal{D}, h)$. $\mathcal{D}$ then publishes each $\Pi_\mathcal{C}$ and the OT parameters $pk_\mathcal{D}$ via an authenticated channel.

$\mathsf{OTObtainCred}(\mathcal{U}(pk_\mathcal{D}, sk_\mathcal{U}, \Pi_\mathcal{C}), \mathcal{D}(pk_\mathcal{U}, sk_\mathcal{D}, \Pi_\mathcal{C}, S))$**:** When user $\mathcal{U}$ wishes to join the system, it negotiates with $\mathcal{D}$ to agree on a policy Π and initial state S, then:

1. $\mathcal{U}$ picks a random show nonce $N_s \in \mathbb{Z}_q$ and computes
   $A \leftarrow \mathrm{Commit}(sk_\mathcal{U}, N_s)$.
2. $\mathcal{U}$ conducts an interactive proof to convince $\mathcal{D}$ that A correlates to $pk_\mathcal{U}$, and $\mathcal{D}$ conducts an interactive proof of knowledge to convince $\mathcal{U}$ that $e(g, h) = H$.[a]
3. $\mathcal{U}$ and $\mathcal{P}$ run the CL signing protocol on committed values so that $\mathcal{U}$ obtains the state signature $\sigma_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_\mathcal{P}, (sk_\mathcal{U}, N_s, \mathsf{pid}, S))$ with pid, S contributed by $\mathcal{P}$.
4. $\mathcal{U}$ stores the credential $\mathsf{Cred} = (\Pi_\mathcal{C}, S, \sigma_{\mathsf{state}}, N_s)$.

[a] This proof can be conducted efficiently in four rounds as in [15].

Fig. 4. Setup and user registration algorithms for an access controlled oblivious database based on the Camenisch, Neven and shelat oblivious transfer protocol [15]. The database operator and users first run the Setup portion of the protocol. Each user subsequently registers with the database using OTObtainCred.

Theorem 2. *The scheme described above satisfies database and user security (as defined in Definition 1) under the q-PDDH, q-SDH, and Strong RSA assumptions.*

A full proof of Theorem 2 appears in the full version of this work [21]. We sketch the broad outlines of the proof in Appendix B.

$\mathsf{OTAccessAndUpdate}(\mathcal{U}(pk_{\mathcal{D}}, sk_{\mathcal{U}}, \mathsf{Cred}, t), \mathcal{D}(pk_{\mathcal{D}}, E))$: When $\mathcal{U}$ wishes to obtain the message indexed by $i \in [1, N+1]$, it first identifies a tag t in Π such that $t = (\mathsf{pid}, S \to T, i)$.

1. $\mathcal{U}$ parses $\mathsf{Cred} = (\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s)$, and parses $\Pi_{\mathcal{C}}$ to find $\sigma_{S \to T, i}$.
2. $\mathcal{U}$ selects $N'_s \xleftarrow{\$} \mathbb{Z}_q$ and computes $A \leftarrow \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, \mathsf{pid}, T)$.
3. $\mathcal{U}$ then sends (N_s, A) to $\mathcal{D}$. $\mathcal{D}$ checks the database E for $(N_s, A' \neq A)$, and if it finds such an entry it aborts. Otherwise it adds (N_s, A) to E.
4. $\mathcal{U}$ parses $C_i = (A_i, B_i)$. It selects a random $v \leftarrow \mathbb{Z}_q$ and sets $V \leftarrow (A_i)^v$. It sends V to $\mathcal{D}$ and proves knowledge of $(i, v, sk_{\mathcal{U}}, \sigma_{S \to T, i}, \sigma_{\mathsf{state}}, \mathsf{pid}, S, T, N'_s)$ such that the following conditions hold:
   (a) $e(V, y) = e(g, g)^v e(V, g)^{-i}$.
   (b) $A = \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, \mathsf{pid}, T)$.
   (c) $\mathsf{CLVerify}(spk_{\mathcal{P}}, \sigma_{\mathsf{state}}, (sk_{\mathcal{U}}, N_s, \mathsf{pid}, S)) = 1$.
   (d) $\mathsf{CLVerify}(\mathcal{P}, \sigma_{S \to T, i}, (\mathsf{pid}, S, T, i)) = 1$.
5. If these proofs verify, $\mathcal{U}$ and $\mathcal{D}$ run the CL signing protocol on committed values such that $\mathcal{U}$ obtains $\sigma'_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_{\mathcal{D}}, A)$. $\mathcal{U}$ stores the updated credential $\mathsf{Cred}' = (\Pi_{\mathcal{C}}, T, \sigma'_{\mathsf{state}}, N'_s)$.
6. Finally, $\mathcal{D}$ returns $U = e(V, h)$ and interactively proves that U is correctly formed (see [15]). $\mathcal{U}$ computes the message $M_i = B_i / U^{1/v}$.

Fig. 5. Database access protocol for an access-controlled oblivious database based on the Camenisch, Neven and shelat adaptive oblivious transfer protocol [15]

5.3 Extensions to Compact Access Policies in Practice

Extension #1: Equivalence Classes. Thus far, the protocol requires that a tag in the policy graph must be defined on every item index in the database. Yet, there are cases where many items may have the same access rules applied, and therefore we can reduce the number tags used by referring to the entire group with a single tag. A simple solution is to replace specific item indices with general equivalence classes in the graph tags. The OT database can be reorganized to support this concept by renumbering the item indices (previously $[1, N]$) using values of the form $(c||i) \in \mathbb{Z}_q$, where c is the identity of the item class, and $||$ represents concatenation. During the $\mathsf{OTAccessAndUpdate}$ protocol, $\mathcal{U}$ can obtain any item $(c||i)$ by performing a zero-knowledge proof on the first half of the selection index, showing that the selected tag contains the class c.

Extension #2: Encoding Contiguous Ranges. An alternative approach requires the database operator to arrange the identities of objects in the same class so that they fall in contiguous ranges. In this case, we will label the graph edges with *ranges* of items rather than single values. The credentials will also replace the value i with an upper and lower bound for the range that the holder of the credential is permitted to access. We make a slight change to the $\mathsf{OTAccessAndUpdate}$ protocol so that rather than proving equality between the requested object and the object present in the tag, the user now proves that the requested object lies in the range described in the user selected tag, as described by the hidden range proof technique in Section 3. Notice that while this approach requires that the database be reorganized such that classes of items remain in contiguous index ranges, it can be used to represent more advanced data structures, such as hierarchical classes.

6 Conclusion

In this paper, we presented a flexible and efficient system that allows content providers to control access to their data, while simultaneously maintaining the privacy provided by the oblivious and anonymous protocols. Specifically, we described techniques for augmenting traditional anonymous credentials with state, and showed how to combine these credentials with Oblivious Transfer to permit oblivious access to a database enforcing a variety of non-trivial access control policies. The flexibility of our approach makes it relatively straightforward to apply to a diverse set of anonymous and oblivious protocols. For example, our stateful anonymous credentials can be used to control which messages are signed with several blind signature schemes, including those of Waters [35], Boneh and Boyen [4], and Camenisch and Lysyanskaya [11,12], without ever revealing the message to the signer. Other interesting applications include augmenting oblivious versions of Identity-Based key extraction [28] and keyword search protocols [31] with strong access controls.

Acknowledgments

The authors thank Zachary Crisler for helpful comments on a prior draft. The work of Scott Coull was supported in part by the U.S. Department of Homeland Security Science & Technology Directorate under Contract No. FA8750-08-2-0147. Matthew Green and Susan Hohenberger gratefully acknowledge the support of NSF grant CNS-0716142 and a Microsoft New Faculty Fellowship.

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
2. Elliot Bell, D., Elliot Bell, D., LaPadula, L.J.: Secure Computer System: Unified Exposition and Multics Interpretation. Comm. of the ACM 1, 271–280 (1988)
3. Blake, I.F., Kolesnikov, V.: Strong Conditional Oblivious Transfer and Computing on Intervals. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 515–529. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 382–400. Springer, Heidelberg (2004)
5. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)

6. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997)
7. Brewer, D.F.C., Nash, M.J.: The Chinese Wall Security Policy. In: IEEE Symposium on Security and Privacy, pp. 206–214 (1989)
8. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: Efficient periodic n-times anonymous authentication. In: ACM CCS 2006, pp. 201–210 (2006)
9. Camenisch, J.L., Hohenberger, S., Lysyanskaya, A.: Balancing Accountability and Privacy Using E-Cash (Extended Abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 141–155. Springer, Heidelberg (2006)
10. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
11. Camenisch, J.L., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
12. Camenisch, J.L., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
13. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number n is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
14. Camenisch, J., Michels, M.: Separability and efficiency for generic group signature schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 413–430. Springer, Heidelberg (1999)
15. Camenisch, J.L., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
16. Camenisch, J.L.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich (1998)
17. Chan, A., Frankel, Y., Tsiounis, Y.: Easy come – easy go divisible cash. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 561–575. Springer, Heidelberg (1998)
18. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
19. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
20. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
21. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. Cryptology ePrint Archive, Report 2008/474 (2008), `http://eprint.iacr.org/`
22. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
23. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)
24. Damgård, I.B., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)

25. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs an Keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
26. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
27. Google. Google Health (2008), `http://www.google.com/intl/en-US/health/about/index.html`
28. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
29. Lysyanskaya, A.: Signature schemes and applications to cryptographic protocol design. PhD thesis, MIT, Cambridge, Massachusetts (September 2002)
30. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
31. Ogata, W., Kurosawa, K.: Oblivious keyword search. Special issue on coding and cryptography J. of Complexity 20(2-3), 356–371 (2004)
32. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
33. Schnorr, C.-P.: Efficient signature generation for smart cards. J. of Cryptology 4(3), 239–252 (1991)
34. Teranishi, I., Furukawa, J., Sako, K.: k-Times Anonymous Authentication. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 308–322. Springer, Heidelberg (2004)
35. Waters, B.: Efficient Identity-Based Encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A Security Definition for an Oblivious Database

Definition 1 (Security for Oblivious Databases with Access Controls). Security is defined according to the following experiments, which are based on those of Camenisch *et al.* [15]. Although we do not explicitly specify auxiliary input to the parties, we note that this information can be provided in order to achieve sequential composition.

Real experiment. The real-world experiment $\mathbf{Real}_{\hat{\mathcal{D}},\hat{\mathcal{U}}_1,\ldots,\hat{\mathcal{U}}_\eta}(\eta, N, k, \Pi_1, \ldots, \Pi_\eta, M_1, \ldots, M_N, \Sigma)$ is modeled as k rounds of communication between a possibly cheating database $\hat{\mathcal{D}}$ and a collection of η possibly cheating users $\{\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_\eta\}$. In this experiment, $\hat{\mathcal{D}}$ is given the policy graph for each user $\Pi_1, \ldots, \Pi_\eta$, a message database $M_1, \ldots, M_N$ and the users are given an adaptive strategy Σ that, on input of the user's identity and current graph state, outputs the next action to be taken by the user.

At the beginning of the experiment, the database and users conduct the Setup and $\mathsf{OTObtainCred}$ protocols. At the end of this step, $\hat{\mathcal{D}}$ outputs an initial state D_1, and each user $\hat{\mathcal{U}}_i$ output state $U_{1,i}$. For each subsequent round $j \in [2, k]$, each user may interact with $\hat{\mathcal{D}}$ to request an item $i \in [1, N+1]$ as required by the strategy Σ. Following each round, $\hat{\mathcal{D}}$ outputs D_j, and the users output

$(U_{1,j},\dots,U_{\eta,j})$. At the end of the k^{th} round the output of the experiment is $(D_k, U_{1,k},\dots,U_{j,k})$.

We will define the *honest* database $\mathcal{D}$ as one that honestly runs its portion of Setup in the first round, honestly runs its side of the $\mathsf{OTObtainCred}$ and $\mathsf{OTAccessAndUpdate}$ protocols when requested by a user at round $j > 1$, and outputs $D_k = \mathit{params}$. Similarly, an honest user $\mathcal{U}_i$ runs the Setup protocol honestly in the first round, and executes the user's side of the $\mathsf{OTObtainCred}$, $\mathsf{OTAccessAndUpdate}$ protocols, and eventually outputs the received value Cred along with all database items received.

Ideal experiment. In experiment $\mathbf{Ideal}_{\hat{\mathcal{D}}',\hat{\mathcal{U}}'_1,\dots,\hat{\mathcal{U}}'_\eta}(\eta, N, k, \Pi_1,\dots,\Pi\eta, M_1,\dots, M_N, \Sigma)$ the possibly cheating database $\hat{\mathcal{D}}'$ generates messages $(M_1^*,\dots,M_N^*)$ and sends these, along with the policy graphs to the trusted party $\mathcal{T}$ (each policy graph specifies its initial state). In each round $j \in [1,k]$, every user $\hat{\mathcal{U}}'$ (following strategy Σ) selects a message index $i \in [1, N+1]$ and sends a message containing the user's identity and (i, S, T) to $\mathcal{T}$. $\mathcal{T}$ then checks the policy graph corresponding to that user to determine if $\hat{\mathcal{U}}'$ is in state S and a transition to T is permitted, sending $\hat{\mathcal{D}}'$ a bit b_1 indicating the outcome of this test. $\hat{\mathcal{D}}'$ then returns a bit b_2 determining whether the transaction should succeed. If $b_1 \wedge b_2$, then $\mathcal{T}$ returns M_i^* (or $\perp$ if $i = N+1$) to $\hat{\mathcal{U}}'_i$, otherwise it returns $\perp$. Following each round, $\hat{\mathcal{D}}'$ outputs D_j, and the users output $(U_{1,j},\dots,U_{\eta,j})$. At the end of the k^{th} round the output of the experiment is $(D_k, U_{1,k},\dots,U_{\eta,k})$.

We will define the *honest* database $\mathcal{D}$ as one that sends $M_1,\dots,M_N$ in the first round, returns $b_2 = 1$ in all rounds, and outputs $D_k = \epsilon$. Similarly, an honest user $\mathcal{U}'_i$ runs the Setup protocol honestly in the first round, makes queries and transitions according to the selection policy, and eventually outputs all received database items as its output.

Let $\ell(\cdot), c(\cdot), d(\cdot)$ be polynomials. We now define database and user security in terms of the experiments above.

Database Security. A stateful anonymous credential scheme is database-secure if for every collection of (possibly corrupted) real-world p.p.t. receivers $\hat{\mathcal{U}}_1,\dots,\hat{\mathcal{U}}_\eta$ there exists a collection of p.p.t. ideal-world receivers $\hat{\mathcal{U}}'_1,\dots,\hat{\mathcal{U}}'_\eta$ such that $\forall N = \ell(\kappa)$, $\eta = d(\kappa)$, $k \in c(\kappa)$, Σ, and every p.p.t. distinguisher:

$$\mathbf{Real}_{\mathcal{D},\hat{\mathcal{U}}_1,\dots,\hat{\mathcal{U}}_\eta}(\eta, N, k, \Pi_1,\dots,\Pi_\eta, M_1,\dots,M_N,\Sigma) \overset{c}{\approx} \mathbf{Ideal}_{\mathcal{D},\hat{\mathcal{U}}'_1,\dots,\hat{\mathcal{U}}'_\eta}(\eta, N, k, \Pi_1,\dots,\Pi_\eta, M_1,\dots,M_N,\Sigma)$$

User Security. A stateful anonymous credential scheme provides Receiver security if for every real-world (possibly corrupted) p.p.t. database $\hat{\mathcal{D}}$ and collection of (possibly corrupted) users $\hat{\mathcal{U}}_1,\dots,\hat{\mathcal{U}}_\eta$, there exists a p.p.t. ideal-world sender $\hat{\mathcal{D}}'$ and ideal (possibly corrupted) users $\hat{\mathcal{U}}'_1,\dots,\hat{\mathcal{U}}'_\eta$ such that $\forall N = \ell(\kappa)$, $\eta = d(\kappa)$, $k \in c(\kappa)$, Σ, and every p.p.t. distinguisher:

$$\mathbf{Real}_{\hat{\mathcal{D}},\hat{\mathcal{U}}_1,\dots,\hat{\mathcal{U}}_\eta}(\eta, N, k, \Pi_1,\dots,\Pi_\eta, M_1,\dots,M_N,\Sigma) \overset{c}{\approx} \mathbf{Ideal}_{\hat{\mathcal{D}}',\hat{\mathcal{U}}'_1,\dots,\hat{\mathcal{U}}'_\eta}(\eta, N, k, \Pi_1,\dots,\Pi_\eta, M_1,\dots,M_N,\Sigma)$$

B Proof Sketch of Theorem 2

We now sketch a proof of Theorem 2, arguing security for the oblivious database protocol of §5.2. Our sketch will refer substantially to the original proof of the underlying adaptive oblivious transfer protocol, which is due to Camenisch, Neven and shelat [15]. Our proof will consider two components: (1) the security of the underlying OT scheme (which is based on the proof of [15]), and a separate proof of the anonymous credential scheme.

Proof outline. We separately consider User and Database security.

User Security. Let us assume that an adversary has corrupted a database $\mathcal{D}$ and some subset of the users $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_N$. In this model, corruptions will be static. We show that for every such adversary, we can construct a simulator such that the output of the ideal experiment conducted with the simulator will be indistinguishable from the output of the real experiment.

Our simulator operates as follows. First, $\mathcal{D}$ outputs the parameters for the credential system, the cryptographic representation of each graph, and the values $pk, C_1, \ldots, C_N$. If these parameters are incorrectly formed, the simulator aborts. The simulator next generates a credential key for each uncorrupted user and negotiates with $\mathcal{D}$ to join the system under an appropriate policy. When $\mathcal{D}$ executes the proof of knowledge that $H = e(g, h)$ with some uncorrupted user, our simulator rewinds to extract the value h (this extraction succeeds with all but negligible probability). For $i = 1$ to N, the simulator decrypts C_i using h to obtain M_i. This collection of plaintexts is sent to the trusted party $\mathcal{T}$.

When a corrupted user $\hat{\mathcal{U}}$ queries the database, we pass its communications along to $\hat{\mathcal{D}}$ unmodified. Whenever an uncorrupted user $\mathcal{U}$ queries $\mathcal{T}$ to obtain message i (according to a state transition defined in their policy), $\mathcal{T}$ verifies that this request is permitted by policy and updates its view of the user's state. Next, it notifies our simulator which runs the $\mathsf{OTAccessAndUpdate}$ protocol on an arbitrary (uncorrupted) user's policy under index $N + 1$ (this is the "dummy" transition and is always permitted by the credential system). If this protocol succeeds, the simulator sends a bit 1 to $\mathcal{T}$ which returns M_i to the user.

Claim. The transcript produced by this simulator is indistinguishable from the transcript produced by the real experiment. This is true for following reasons:

1. The probability that the simulator *incorrectly* extracts h (or fails to extract it) is negligible.
2. The probability that the adversary distinguishes a protocol executed on an arbitrary user/dummy index is negligible: this is due to (a) the fact that the element V transmitted to $\mathcal{D}$ during $\mathsf{OTAccessAndUpdate}$ is randomly distributed, and (b) the attached proof-of-knowledge is witness indistinguishable and therefore does not reveal the value of i or the user's identity.

We do not need to argue the unforgeability of the anonymous credential scheme here, since we consider only actions taken by the uncorrupted user.

Database Security. Let us assume that an adversary has corrupted some subset of the users $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_N$ (corruptions are static). We show that for every such adversary, we can construct a simulator such that the output of the ideal experiment conducted with the simulator will be indistinguishable from the output of the real experiment.

Our simulator operates as follows. First, it generates the public and privacy parameters for the credential scheme along with the cryptographic representation of the policies provided by $\mathcal{T}$. It generates the parameters for the OT scheme pk, sk as normal, but sets the plaintext for each database element to a dummy value (the identity element) and produces ciphertexts $C_1, \ldots, C_N$ (and generates the dummy message $C_{(N+1)}$ as normal). It sends these parameters to each corrupted user, and to each user proves that $H = e(g, h)$.

Whenever a corrupted user initiates the $\mathsf{OTAccessAndUpdate}$ protocol with $\mathcal{D}$, the simulator verifies that the user's request (including ZK proofs) verifies, and that neither N_u or N_s has been seen before. If so, it rewinds and uses the extractors for the ZK proofs to learn the user's identity, the index of the message i being requested, the blinding factor v, and the user's current and previous credential state S, T. The server transmits the user's identity values (i, S, T) to $\mathcal{T}$ which verifies that they satisfy the policy (updating the policy state in the process). If $\mathcal{T}$ returns $\perp$, then $\mathcal{D}$ aborts the protocol with the user. Otherwise if $\mathcal{T}$ returns M_i, then the simulator parses $C_i = (A_i, B_i)$ and returns $U = (B_i^v)/M_i$. The simulator uses rewinding to simulate the proof and convince the user that U has been correctly formed.

Claim. The transcript produced by this simulator is indistinguishable from one produced by the real experiment. This claim rests on the following points:

1. The false message collection $C_1, \ldots, C_{(N+1)}$ is indistinguishable from the real message by the semantic security of the encryption scheme, which holds under the q-PDDH assumption (see [15] for the full argument).
2. The simulated proof of U's structure is indistinguishable from a real proof.
3. The simulator never queries $\mathcal{T}$ on a tuple (i, S, T) that violates the user's policy. This reduces to the unforgeability of the CL signature (which is in turn based on Strong RSA). Specifically, to violate policy, a user must satisfy one of the following conditions:
   (a) Prove knowledge of a signature σ_δ that it was not given, or
   (b) Prove knowledge of a signature $\sigma_{S \to T}$ that it was not given. In either case, the simulator can use the extractor for the proof system to obtain the forged signature and win the CL signature forgery game.
   (c) Misuse the CL signing protocol such that it receives a signature that is not equivalent to a signature on the commitment A (or mispresent the structure of A).

Author Index