Jonathan Katz
Moti Yung (Eds.)

LNCS 4521

Applied Cryptography and Network Security

5th International Conference, ACNS 2007
Zhuhai, China, June 2007
Proceedings

Springer

Jonathan Katz   Moti Yung (Eds.)

Applied Cryptography and Network Security

5th International Conference, ACNS 2007
Zhuhai, China, June 5-8, 2007
Proceedings

Springer

Volume Editors

Jonathan Katz
University of Maryland
Dept. of Computer Science
A.V. Williams Building, College Park, MD 20742, USA
E-mail: jkatz@cs.umd.edu

Moti Yung
RSA Laboratories and
Columbia University, Computer Science Department
S.W. Mudd Building, New York, NY 10027, USA
E-mail: moti@cs.columbia.edu

Library of Congress Control Number: 2007927501

CR Subject Classification (1998): E.3, C.2, D.4.6, H.4, K.4.4, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-540-72737-X Springer Berlin Heidelberg New York
ISBN-13	978-3-540-72737-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12067848 06/3180 5 4 3 2 1 0

Preface

The Fifth International Conference on Applied Cryptography and Network Security (ACNS 2007) was held in Zhuhai, China, June 5–8, 2007. This volume contains papers that were accepted to the academic track of the conference.

The conference received an astounding number of submissions this year, which made the review process a challenging and demanding task. We are indebted to the members of the Program Committee and the external reviewers for all their hard work. The committee accepted 31 papers from roughly 260 submissions. These proceedings contain revised versions of the accepted papers. While revisions are expected to take the referees' comments into account, this was not enforced and the authors bear full responsibility for the content of their papers.

In addition to the academic track, the conference hosted a non-archival industrial track whose papers were also carefully selected from among the submissions.

Shai Halevi deserves the community's gratitude for writing his Web submission and review software, which we used for this conference. On a more personal level, we would like to extend our own deepest thanks to Shai for not only writing his software, but for installing and maintaining the submission server for this conference. Thanks go also to the International Association for Cryptologic Research (IACR) for agreeing to host the server.

It is our pleasure to thank the General Chair Yongfei Han, the Publicity Chair Jianying Zhou, and the Chair of the Organizing Committee Li Nan for their help and support in putting this conference together. Without their help, this conference would not have been possible. Finally, we are grateful to ONETS and Zhuhai College, Jilin University, for sponsoring the conference.

March 2007

Jonathan Katz
Moti Yung

ACNS 2007

Fifth International Conference on Applied Cryptography and Network Security

Zhuhai, China
June 5-8, 2007

Organized and Sponsored by

ONETS, China
and
Zhuhai College, Jilin University, China

Kaoru Kurosawa	Ibaraki University, Japan
Xuejia Lai	Shanghai Jiaotong University, China
Dong Hoon Lee	CIST, South Korea
Phil MacKenzie	Google, USA
Ilya Mironov	Microsoft Research, USA
Pascal Paillier	Gemalto, France
Kenny Paterson	Royal Holloway, University of London, UK
Raphael Phan	EPFL, Switzerland
Benny Pinkas	University of Haifa, Israel
David Pointcheval	CNRS and ENS, France
Zulfikar Ramzan	Symantec, Inc., USA
Phil Rogaway	UC Davis, USA and Chiang Mai University, Thailand
Kazue Sako	NEC, Japan
Palash Sarkar	Indian Statistical Institute, India
Vitaly Shmatikov	University of Texas at Austin, USA
Thomas Shrimpton	Portland State University, USA
Nigel Smart	University of Bristol, UK
Ron Steinfeld	Macquarie University, Australia
Adam Stubblefield	Johns Hopkins University, USA
Mike Szydlo	Akamai, USA
Brent Waters	SRI International, USA
Avishai Wool	Tel Aviv University, Israel
Sung-Ming Yen	National Central University, Taiwan
Jianying Zhou	Institute for Infocomm Research, Singapore

Publicity Chair

Jianying Zhou	Institute for Infocomm Research, Singapore

Organizing Committee

Li Nan	ONETS, China

Steering Committee

Yongfei Han	ONETS, China
Moti Yung	Columbia University, USA
Jianying Zhou	Institute for Infocomm Research, Singapore

External Reviewers

Gergely Acs	Dan Bailey	Constantinos Bartzis
Ben Adida	Lucas Ballard	Ohad Ben-Cohen
Toshinori Araki	Gregory V. Bard	Boldizsar Bencsath
Joonsang Baek	Elad Barkan	Bobby Bhattacharjee

Marina Blanton	Carmit Hazay	Toru Nakanishi
Jin Wook Byun	Swee-Huay Heng	Juanma Nieto
Srdjan Capkun	T. Heydt-Benjamin	Satoshi Obana
Aldar Chan	Shoichi Hirose	Jong Whan Park
Melissa Chase	James Hoagland	Maura Paterson
Sanjit Chatterjee	Chao-Chih Hsu	Michael Ø. Pedersen
Chien-Ning Chen	Toshiyuki Isshiki	Chris Peikert
Pau-Chen Cheng	Ik Rae Jeong	Duong Hieu Phan
Benoit Chevallier-Mames	Antoine Joux	Le Trieu Phong
Han-Fei Chiang	Marcelo Kaihara	Josef Pieprzyk
Kuo-Zhe Chiou	Edward Kaiser	Axel Poschman
Eun Young Choi	Yael Tauman Kalai	Julio Quinteros
Kyu Young Choi	Seny Kamara	Moheeb Abu Rajab
Seung Geol Choi	Aggelos Kiayias	David Safford
JM Combes	Eike Kiltz	Peter Schaffer
Scott Contini	Bum Han Kim	Jacob Schuldt
Debbie Cook	Hugo Krawczyk	Hovav Shacham
Laszlo Csik	Jeong Ok Kwon	Radu Sion
Yang Cui	Amit Lakhani	William Skeith
Reza Curtmola	Loukas Lazos	Sam Small
Dimitri DeFigueiredo	Hwa Sung Lee	Angelo Spognardi
Blandine Debraize	Hyun Sook Lee	Martijn Stam
Benessa Defend	Tieyan Li	Keisuke Tanaka
Alex Dent	Xiangxue Li	Isamu Teranishi
Laszlo Dora	Wei-Chih Lien	Dominique Unruh
Ehud Doron	Hsi-Chung Lin	Matthew Vail
Markus Duermuth	Lang Lin	Istvan Vajda
Wu-chang Feng	Yehuda Lindell	Yongdong Wu
Pierre-Alain Fouque	Nathan Linger	Guilin Wang
Aurelien Francillon	Matteo Maffei	Huaxiong Wang
Eiichiro Fujisaki	Wenbo Mao	Enav Weinreb
Jun Furukawa	Josh Mason	Stephen A. Weis
Steven Galbraith	Breno de Medeiros	Chi-Dian Wu
Craig Gentry	Kazuhiko Minematsu	Kazuo Yanoo
Vipul Goyal	Atsuko Miyaji	Po-Wah Yau
Matt Green	Nagendra Modadugu	Lidong Zhou
David Gross-Amblard	Kengo Mori	Huafei Zhu
Fanglu Guo	Yoichiro Morita	
Goichiro Hanaoka	Masayuki Nakae	

Table of Contents

Generic Transformation to Strongly Unforgeable Signatures[⋆]

Qiong Huang[1], Duncan S. Wong[1], and Yiming Zhao[2]

[1] Dept. of Computer Science,
City University of Hong Kong
Hong Kong, China
{csqhuang,duncan}@cityu.edu.hk

[2] Dept. of Computer Science and Engineering,
Fudan University
Shanghai 200433, China
zhym@fudan.edu.cn

Abstract. Recently, there are several generic transformation techniques proposed for converting unforgeable signature schemes (the message in the forgery has not been signed yet) into strongly unforgeable ones (the message in the forgery could have been signed previously). Most of the techniques are based on trapdoor hash functions and all of them require adding supplementary components onto the original key pair of the signature scheme. In this paper, we propose a new generic transformation which converts *any* unforgeable signature scheme into a strongly unforgeable one, and also keeps the key pair of the signature scheme unchanged. Our technique is based on *strong one-time signature schemes*. We show that they can be constructed efficiently from any one-time signature scheme that is based on one-way functions. The performance of our technique also compares favorably with that of those trapdoor-hash-function-based ones. In addition, this new generic transformation can also be used for attaining strongly unforgeable signature schemes in other cryptographic settings which include certificateless signature, identity-based signature, and several others. To the best of our knowledge, similar extent of versatility is not known to be supported by any of those comparable techniques. Finally and of independent interest, we show that our generic transformation technique can be modified to an *on-line/off-line* signature scheme, which possesses a very efficient signing process.

1 Introduction

When considering the security of a signature scheme, we usually refer to the existential unforgeability against adaptive chosen message attacks [16]. The

[⋆] The first two authors are supported by a grant from CityU (Project No. 7001844). The third author is supported by National Natural Science Foundation of China under Grant No. 60573054.

security requirement is to prevent forgery of signatures on new messages not previously signed. However, most signature schemes are randomized and allow many possible signatures for a single message. In some applications, a stronger security notion, called *strong unforgeability*, is desirable. It prevents forgery of signatures on messages that could have been signed previously. Applications of strongly unforgeable signature schemes include signcryption [2], encryption secure against chosen ciphertext attacks [13,10], group signature [8,3], authenticated group key exchange [18] and etc. [9]. Unfortunately, many signature schemes in the literature are not strongly unforgeable. Recently, some techniques [9,30,6,29] have been proposed to convert existing schemes to strongly unforgeable ones. However, these techniques require to add some supplementary parameters onto the original key pairs of the signature schemes. This may introduce some inconvenience or operational issue in practice, for example, new public key certificates may need to be requested for those augmented public keys.

A *Generic* and *Universal* Transformation. In this paper, we present a new generic transformation which converts *any* signature scheme to a strongly unforgeable one. When comparing with existing techniques [9,30,29] which are based on trapdoor hash functions, our method has the following merits.

1. The transformation adds *no* additional component into the original public/private key pair; and
2. the transformation is *universal* in the sense that the same transformation technique can be used to convert schemes in other cryptographic settings to strongly unforgeable ones. These cryptographic settings include identity-based signature [27], certificateless signature [1] and several others (Sec. 4).

Furthermore, a strongly-unforgeable signature scheme obtained from our transformation can also be used as an *on-line/off-line* signature [14,28]. Most of the computational-intensive part of the signing process can be done off-line, and this leaves only a little work to be carried out on-line (essentially, only one hash evaluation is left to be done). This helps improve the efficiency of the signing process significantly.

Strong One-time Signature. Our transformation is based on strong one-time signature. A strong one-time signature scheme is a signature scheme which prevents the adversary, making *at most one* signing query, from producing a new signature on a message that could have already been signed. Currently, almost all the one-time signature schemes in the literature [23,19,14,24] have only been shown to be one-time unforgeable rather than strongly one-time unforgeable, that is, they are only ensured to prevent forgery of signatures on new messages not previously signed. The transformation technique to strong one-time signature proposed in [15] requires $O(\ell)$ universal one-way hash functions [21] where ℓ is the length of messages to be signed. In this paper, we propose a simple modification of the method in [15] that improves the efficiency greatly by requiring only *one* collision-resistant hash function.

Related Work. At PKC 2006, Boneh, Shen and Waters [9] presented a transformation technique which converts a large class of existentially unforgeable signature schemes (in the sense of [16]) into strongly unforgeable ones. Their transformation is based on trapdoor hash functions and applies to a class of signature schemes, named *partitioned* signatures. A signature is said to be partitioned if (1) part of the signature, denoted by σ_2, is independent of the message m, and (2) given m and σ_2, the signature can be fully determined. Although many standard signature schemes fall into this class, as the authors pointed out in [9], DSS [22] may not be partitioned.[1]

Recently, Teranishi et al. [30] proposed two trapdoor-hash-function-based conversions which can convert *any* (standard) signature scheme to a strongly unforgeable one. The first conversion works by modeling the hash function (used in the trapdoor commitment) as a *random oracle* [5], while the second one works in the standard model, and uses a trapdoor commitment scheme with two trapdoors. With the knowledge of any one of the trapdoors, the simulator can simulate the game for the forger. Independently and concurrently, Steinfeld, Pieprzyk and Wang [29] proposed another similar transformation technique based on trapdoor hash functions. The idea is to use two trapdoor hash functions and apply the '*hash-then-switch*' method to protect the entire signature (rather than only part of it) from modification. They showed that any valid forgery against strong unforgeability would contradict either the existential unforgeability of the original scheme or the collision-resistance of the underlying trapdoor hash functions.

In all the transformations above, additional public and private key components for the underlying trapdoor hash functions have to be added into the public and private keys of the original signature scheme, respectively. Furthermore, it is not known if their techniques can be applied to signature schemes in other cryptographic settings, for example, in certificateless cryptography [1].

Earlier in [15], Goldreich showed the existence of strongly unforgeable signature schemes based on one-way functions. First, a *strong* one-time signature scheme is constructed from a one-time signature scheme (that follows the '*one-way function paradigm*' [14,15], which will also be introduced in Sec. 5). The construction is based on universal one-way hash functions [21,15] which in turn can be constructed from one-way functions. Then, by applying the 'authentication-tree' method [15], a strongly unforgeable signature scheme can be constructed. However, this is only a theoretical construction for the feasibility, and thus is inefficient.

Interestingly and independently of our work, Bellare and Shoup [6] propose a construction, which is quite similar with ours, to transform existentially unforgeable signature schemes into strongly unforgeable ones. Their transformation employs a *two-tier signature* [6] scheme rather than a one-time signature. Thus, the key structure of the original signature scheme is also changed by adding the key pair of the underlying *two-tier signature* scheme ds into it, if the primary key of ds is not empty.

[1] Readers may also refer to [29] for some additional discussions about this.

Paper organization. In next section, we review the definitions of unforgeable and strongly unforgeable signature schemes and the respective definitions for one-time signature schemes. Our generic transformation technique is proposed and shown to be secure in Sec. 3. In Sec. 4, the generic transformation is extended to certificateless signatures and identity-based signatures, and extensions to other settings are discussed. In Sec. 5, we propose a method to convert any one-time signature scheme following the one-way function paradigm into a strong one-time unforgeable one, and discuss its efficiency. In Sec. 6, we show how to use our generic transformation to construct an efficient *on-line/off-line* signature scheme, and conclude the paper.

2 Preliminaries

A signature scheme SIG consists of three (probabilistic) polynomial-time algorithms, KG, Sign and Vrfy, which are key generation, signature generation and verification, respectively. *Existential unforgeability against adaptive chosen message attacks* [16] for SIG can be defined using the following game called **Game-General**:

Setup: A public/private key pair $(pk, sk) \leftarrow \text{KG}(1^k)$ is generated and adversary $\mathcal{A}$ is given the public key pk.

Query: $\mathcal{A}$ runs for time t and issues q signing queries to a signing oracle in an adaptive manner, that is, for each i, $1 \le i \le q$, $\mathcal{A}$ chooses a message $m^{(i)}$ based on the message-signature pairs that $\mathcal{A}$ has already seen, and obtains in return a signature $\sigma^{(i)}$ on $m^{(i)}$ from the signing oracle (i.e., $\sigma^{(i)} = \text{Sign}(sk, m^{(i)})$).

Forge: $\mathcal{A}$ outputs a forgery (m^*, σ^*) and halts. $\mathcal{A}$ wins if
- σ^* is a valid signature on message m^* under the public key pk, i.e., $\text{Vrfy}(pk, \sigma^*, m^*) = 1$; and
- m^* has never been queried, i.e., $m^* \notin \{m^{(1)}, m^{(2)}, \cdots, m^{(q)}\}$.

Definition 1 (Unforgeability). *A signature scheme SIG = (KG, Sign, Vrfy) is (t, q, ε)-existentially unforgeable against adaptive chosen message attacks (or* ***unforgeable****, in short), if any adversary with run-time t wins in* ***Game-General*** *with probability at most ε after issuing at most q signing queries.*

One of the restrictions for adversary $\mathcal{A}$ in **Game-General** is that the forging message m^* must be new and has not been signed. We can relax this restriction to obtain the notion of ***strong*** *existential unforgeability against adaptive chosen message attacks*, such that $\mathcal{A}$ forges a new valid signature on a message that could have been signed previously. We refer to this new game as **Game-Strong** which is defined as follows.

The **Setup** and **Query** phases are the same as in **Game-General**.

Forge: $\mathcal{A}$ outputs a forgery (m^*, σ^*) and halts. $\mathcal{A}$ wins if
- σ^* is a valid, i.e., $\text{Vrfy}(pk, \sigma^*, m^*) = 1$; and
- $(m^*, \sigma^*) \notin \{ (m^{(i)}, \sigma^{(i)}) \}_{i \in \{1,2,\cdots,q\}}$.

Definition 2 (Strong Unforgeability). *A signature scheme SIG = (KG, Sign, Vrfy) is (t, q, ε)-strongly existentially unforgeable against adaptive chosen message attacks (or **strongly unforgeable**, in short), if any adversary with run-time t wins in **Game-Strong** with probability at most ε after issuing at most q signing queries.*

In our generic transformation proposed later in this paper, one of the primitives we use is the ***strong*** *one-time signature*. Informally, a strong one-time signature scheme is a signature scheme, but each private key is used only once for signature generation. We require that given a (one-time) public key, the adversary is only allowed to make *at most one* signing query before producing a forgery on a message that could have been queried previously. Formally, we define the following game called **Game-StrongOneTime**.

The **Setup** and **Forge** phases are the same as in **Game-Strong**.
Query: same as in **Game-Strong**, except that $q = 1$.

Definition 3 (Strong One-Time Unforgeability). *A signature scheme SIG = (KG, Sign, Vrfy) is a (t, ε)-*strong one-time signature *scheme, if any adversary with run-time t wins **Game-StrongOneTime** with probability at most ε.*

Similarly, a one-time signature (rather than strong) can be defined by strengthening the restriction for $\mathcal{A}$ so that the forgery must contain a new message which has not been signed previously.

3 Our Generic Transformation

In this section, we describe our generic transformation which converts *any* unforgeable signature scheme to a *strongly unforgeable* one. This transformation can be considered as a sequential composition of the original (standard) signature and a strong one-time signature. First, we use the original signature scheme to generate a "certificate" on a freshly generated one-time public key. Then, we use the strong one-time signature scheme to generate a signature on some message and the "certificate". Below are the details.

Let $\mathrm{SIG}' = (\mathrm{KG}', \mathrm{Sign}', \mathrm{Vrfy}')$ be a signature scheme that is unforgeable (Def. 1). Let $\mathrm{SIG}_{OT} = (\mathrm{KG}_{OT}, \mathrm{Sign}_{OT}, \mathrm{Vrfy}_{OT})$ be a strong one-time signature scheme (Def. 3). The transformation is described in Fig. 1, and we have the following theorem:

Theorem 1. *The generic transformation described in Fig. 1 is a (t, q, ε)-strongly unforgeable scheme (Def. 2), provided that SIG' is a $(t, q, \varepsilon/2)$-unforgeable signature scheme (Def. 1) and SIG_{OT} is a $(t, \varepsilon/2q)$-strong one-time signature scheme (Def. 3).*

Proof. Suppose there exists an adversary $\mathcal{A}$ in **Game-Strong** that runs for time t, issues at most q signing queries[2] and breaks the strong unforgeability

[2] W.l.o.g., we assume that $\mathcal{A}$ makes exactly q distinct signing queries.

KG: Generate a public/private key pair $(pk', sk') \leftarrow \mathrm{KG}'(1^k)$, and set public key $pk = pk'$ and private key $sk = sk'$.

Sign: On input private key sk and a message m, the following steps are carried out and a signature σ is generated.

$$\begin{aligned} (vk_{OT}, sk_{OT}) &\leftarrow \mathrm{KG}_{OT}(1^k) \\ \sigma_1 &\leftarrow \mathrm{Sign}'(sk, vk_{OT}) \\ \sigma_2 &\leftarrow \mathrm{Sign}_{OT}(sk_{OT}, m \| \sigma_1) \\ \sigma &\leftarrow (\sigma_1, \sigma_2, vk_{OT}) \end{aligned}$$

Vrfy: On input public key pk, message m and signature $\sigma = (\sigma_1, \sigma_2, vk_{OT})$, $b_1 \wedge b_2$ is returned where $b_1 \leftarrow \mathrm{Vrfy}'(pk, \sigma_1, vk_{OT})$ and $b_2 \leftarrow \mathrm{Vrfy}_{OT}(vk_{OT}, \sigma_2, m \| \sigma_1)$.

Fig. 1. Our Generic Transformation to Strongly Unforgeable Signatures

(Def. 2) of the generic transformation with probability at least ε. We show how to construct adversaries $\mathcal{B}$ and $\mathcal{C}$ that break the strong one-time unforgeability (Def. 3) of SIG_{OT} and the existential unforgeability (Def. 1) of SIG', respectively, such that either $\mathcal{B}$ wins in **Game-StrongOneTime** with probability at least $\varepsilon/2q$ or $\mathcal{C}$ wins in **Game-General** with probability at least $\varepsilon/2$, and both of them run for time slightly greater than t.

Let (m^*, σ^*) be the forgery of $\mathcal{A}$, where $\sigma^* = (\sigma_1^*, \sigma_2^*, vk_{OT}^*)$. For $i = 1, 2, \cdots, q$, let $m^{(i)}$ be the i-th (distinct) query message of $\mathcal{A}$ and $\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, vk_{OT}^{(i)})$ the corresponding signature. We define two events, E_1 and E_2. E_1 is that (m^*, σ^*) is valid and $vk_{OT}^* = vk_{OT}^{(i)}$ for some i $(1 \leq i \leq q)$. E_2 is that (m^*, σ^*) is valid and $vk_{OT}^* \neq vk_{OT}^{(i)}$ for all $1 \leq i \leq q$. As $\Pr[\mathrm{E}_1] + \Pr[\mathrm{E}_2] = \Pr[\mathcal{A} \text{ wins}]$, if $\mathcal{A}$ wins in **Game-Strong**, it must be that either event E_1 or event E_2 occurs. Since $\mathcal{A}$ wins with probability ε, it follows that one of the two events occurs with probability at least $\varepsilon/2$. In the simulations below, $\mathcal{A}$ will be run by each of the adversaries $\mathcal{B}$ and $\mathcal{C}$ which we will construct. If E_1 (respectively, E_2) occurs with probability $\varepsilon/2$, then $\mathcal{B}$ breaks the strong one-time unforgeability of SIG_{OT} with probability $\varepsilon/2q$ (respectively, $\mathcal{C}$ breaks the existential unforgeability of SIG' with probability $\varepsilon/2$).

***Adversary* $\mathcal{B}$.** Given a challenge one-time public key vk_{OT}, which is a random instance in the corresponding key space, and a (one-time) signing oracle $\mathrm{OSign}_{vk_{OT}}$, adversary $\mathcal{B}$ proceeds as below to attack against the strong one-time unforgeability of SIG_{OT}:

Setup: $\mathcal{B}$ runs $\mathrm{KG}(1^k)$ to generate a key pair (pk, sk) for the generic transformation, selects uniformly at random i from $\{1, 2, \cdots, q\}$, and runs $\mathcal{A}$ on input the public key pk.

Query: When $\mathcal{A}$ issues the j-th ($j \neq i$) signing query, $\mathcal{B}$ simulates the signing oracle as if the answer is generated by the real signer. That is, $\mathcal{B}$ responds as follows:

- Run $\mathrm{KG}_{OT}(1^k)$ to generate a one-time key pair $(vk_{OT}^{(j)}, sk_{OT}^{(j)})$;
- Compute $\sigma_1^{(j)} \leftarrow \mathrm{Sign}'(sk, vk_{OT}^{(j)})$;
- Compute $\sigma_2^{(j)} \leftarrow \mathrm{Sign}_{OT}(sk_{OT}^{(j)}, m^{(j)} \| \sigma_1^{(j)})$;
- Return $\sigma^{(j)} \leftarrow (\sigma_1^{(j)}, \sigma_2^{(j)}, vk_{OT}^{(j)})$ to $\mathcal{A}$.

When $\mathcal{A}$ issues the i-th signing query, $\mathcal{B}$ responds as follows:

- Set $vk_{OT}^{(i)} = vk_{OT}$ and compute $\sigma_1^{(i)} \leftarrow \mathrm{Sign}'(sk, vk_{OT}^{(i)})$;
- Obtain a signature $\sigma_2^{(i)}$ on $m^{(i)} \| \sigma_1^{(i)}$ by querying the one-time signing oracle $\mathrm{OSign}_{vk_{OT}}$.
- Return $\sigma^{(i)} \leftarrow (\sigma_1^{(i)}, \sigma_2^{(i)}, vk_{OT}^{(i)})$ to $\mathcal{A}$.

Forge: After $\mathcal{A}$ outputs a forgery (m^*, σ^*) where $\sigma^* = (\sigma_1^*, \sigma_2^*, vk_{OT}^*)$, $\mathcal{B}$ outputs $((m^* \| \sigma_1^*), \sigma_2^*)$ as its forgery for SIG_{OT}.

Since $\mathcal{B}$'s run is essentially a run of $\mathcal{A}$, if $\mathcal{A}$ runs for time t, so does $\mathcal{B}$. Also, $\mathcal{B}$ perfectly simulates the signing oracle for $\mathcal{A}$ as $\mathcal{B}$ follows exactly the signing process except when answering the i-th query. For the i-th query, $\mathcal{B}$ makes a black-box access to its one-time signing oracle $\mathrm{OSign}_{vk_{OT}}$ and the oracle's answer is indistinguishable from those signatures generated by a real signer with respect to the same one-time public key vk_{OT}. Thus, $\mathcal{A}$'s view is identical to that in a real attack (i.e. an exact simulation of **Game-Strong**) and is independent of the choice of i. This implies that $\mathcal{A}$ will succeed with the same probability as in a real attack.

Now we analyze the validity of $\mathcal{B}$'s output under the conditions that event E_1 occurs and $\mathcal{B}$'s guess of i is correct (i.e. $vk_{OT}^* = vk_{OT}^{(i)} = vk_{OT}$). If $(m^* \| \sigma_1^*) \neq (m^{(i)} \| \sigma_1^{(i)})$, by the validity of (m^*, σ^*), we have that $\mathrm{Vrfy}_{OT}(vk_{OT}^*, \sigma_2^*, m^* \| \sigma_1^*) = 1$, hence, $((m^* \| \sigma_1^*), \sigma_2^*)$ is certainly a valid forgery for SIG_{OT}. Then we come to the case that $(m^* \| \sigma_1^*) = (m^{(i)} \| \sigma_1^{(i)})$. Due to the validity of (m^*, σ^*), it must be that $\sigma_2^* \neq \sigma_2^{(i)}$. Therefore, $((m^* \| \sigma_1^*), \sigma_2^*)$ is also a valid forgery for SIG_{OT}, which contradicts the strong unforgeability of SIG_{OT}.

The probability that the choice of i is exactly the one such that $vk_{OT}^* = vk_{OT}^{(i)}$ is $1/q$. Therefore, if event E_1 occurs with probability at least $\varepsilon/2$, $\mathcal{B}$ which runs for time t breaks the security of SIG_{OT} with probability at least $\varepsilon/2q$.

Adversary $\mathcal{C}$. Given a public key pk' of SIG', which is chosen from the output space of $\mathrm{KG}'(1^k)$ at random, and a signing oracle $\mathrm{OSign}_{pk'}$, adversary $\mathcal{C}$ proceeds as below to attack against the existential unforgeability of SIG'.

Setup: $\mathcal{C}$ sets $pk = pk'$, and runs $\mathcal{A}$ on input public key pk. Note that $\mathcal{C}$ does not know the corresponding private key sk.

Query: When $\mathcal{A}$ issues a signing query on some message m, $\mathcal{C}$ simulates the answer as follows:

- Run $\text{KG}_{OT}(1^k)$ to generate a one-time key pair (vk_{OT}, sk_{OT});
- Query the signing oracle $\text{OSign}_{pk'}$ for a signature σ_1 on vk_{OT}, and compute $\sigma_2 \leftarrow \text{Sign}_{OT}(sk_{OT}, m\|\sigma_1)$;
- Return $\sigma \leftarrow (\sigma_1, \sigma_2, vk_{OT})$.

Forge: After $\mathcal{A}$ outputs a forgery (m^*, σ^*) where $\sigma^* = (\sigma_1^*, \sigma_2^*, vk_{OT}^*)$, $\mathcal{C}$ outputs (vk_{OT}^*, σ_1^*) as its forgery for SIG′.

If event E_2 occurs, vk_{OT}^* is a new one-time public key which has not been used by $\mathcal{C}$ in any of the previous queries to its signing oracle $\text{OSign}_{pk'}$. By the validity of $(m^*, (\sigma_1^*, \sigma_2^*, vk_{OT}^*))$ under the public key pk, we have $\text{Vrfy}'(pk, vk_{OT}^*) = 1$. Therefore, (vk_{OT}^*, σ_1^*) is a valid forgery for SIG′.

Since $\mathcal{C}$'s run is essentially a run of $\mathcal{A}$, if $\mathcal{A}$ runs for time t, so does $\mathcal{C}$. Also, $\mathcal{C}$ issues only one signing query to its own oracle $\text{OSign}_{pk'}$ when answering a signing query issued by $\mathcal{A}$, if $\mathcal{A}$ issues q signing queries, so does $\mathcal{C}$. Furthermore, $\mathcal{C}$ perfectly simulates the signing oracle for $\mathcal{A}$ because $\mathcal{C}$ simply follows the signing procedure with the only exception that $\mathcal{C}$ uses its signing oracle $\text{OSign}_{pk'}$ to generate σ_1, and the oracle's output is perfectly indistinguishable from signatures generated by real signers of SIG′ with respect to the same public key. Therefore, $\mathcal{A}$ will succeed with the same probability as that in a real attack. If event E_2 occurs with probability $\varepsilon/2$, $\mathcal{C}$ breaks the existential unforgeability of SIG′ with probability $\varepsilon/2$ as well.

This concludes that if $\mathcal{A}$ (t, q, ε)-breaks the strong unforgeability of SIG, either $\mathcal{B}$ $(t, \varepsilon/2q)$-breaks the strong one-time unforgeability of SIG_{OT}, or $\mathcal{C}$ $(t, q, \varepsilon/2)$-breaks the existential unforgeability of SIG′. □

The efficiency of the generic transformation depends very much on that of the underlying strong one-time signature scheme SIG_{OT}. As we can see, the generic transformation adds one key generation and one signing operation of SIG_{OT} onto the original signing process of SIG′, and one verification operation of SIG_{OT} onto the original verification process. According to [14,24], SIG_{OT} can usually be implemented with very efficient key generation, signing and verifying processes, and short signatures. In addition, the two verification operations of the generic transformation, one for checking σ_1 and the other for σ_2, can be carried out in parallel, that may also be used to improve efficiency. In the next section, we show that the generic transformation can also be extended to transform signatures in other settings such as certificateless signature, identity-based signature and several others, to strongly unforgeable ones. To the best of our knowledge, it is not known if this extent of versatility can also be supported by comparable methods such as [9,30,29].

4 Extensions to Other Cryptographic Settings

In the above, we show how to transform an unforgeable signature scheme to a strongly unforgeable one, under the conventional public key infrastructure. That is, the public key of an entity is assumed to be publicly known, for example due to the presence of a certificate issued by a Certification Authority. In this section, we show that the generic transformation technique can be extended directly for converting signature schemes in other cryptographic settings to their strongly unforgeable ones in some similar context.

4.1 Certificateless Signature and ID-Based Signature

Certificateless cryptography, introduced by Al-Riyami and Paterson [1], is intended to solve the key escrow issue that is inherent in ID-based cryptography. In the following, we adopt the simplified five-algorithm definition of [17] for specifying a certificateless signature scheme. These five algorithms are: master key generation, KG, user partial key generation, PartialKeyGen, user secret key generation, UserKeyGen, signature generation, Sign and verification, Vrfy. In a certificateless cryptosystem, there is a key generation center, KGC, which runs KG to generate its master key pair, and is responsible for generating users' partial key by running PartialKeyGen. Each user in the system have their own (independent) public/private key pair, generated by running UserKeyGen. Two games are considered for the security of a certificateless signature: Game-I and Game-II. Adversary $\mathcal{A}_I$ in Game-I models malicious users, which can compromise user secret key, replace user public key, but cannot get master secret key nor user partial key. Adversary $\mathcal{A}_{II}$ in Game-II models a dishonest KGC which knows master secret key and partial keys of all users but cannot get access to user secret key nor replace user public key. Informally, a certificateless signature is said to be $(t, q_s, q_o, \varepsilon)$-existentially unforgeable (in Game-I, or Game-II) if no adversary with run-time t issuing at most q_s signing queries and at most q_o other oracle queries (such as CreateUser, RevealPartialKey and others specified in the games) succeeds in forging a signature on a new identity-message pair (ID^*, m^*) with probability at least ε. For detailed definitions of security games, we refer readers to [17]. A certificateless signature is said to be *strongly unforgeable* (in Game-I, or Game-II) if it can prevent forgery of new signatures on identity-message pairs which could have been signed previously.

To the best of our knowledge, no certificateless signature scheme in the literature has formally been considered about the strong unforgeability. Also note that the generic composition of certificateless signature scheme from a standard signature scheme and an ID-based signature scheme proposed in [17] does not ensure strong unforgeability even if we assume that both of the underlying primitives are strongly unforgeable. It remains open to construct a generic composition of strongly unforgeable certificateless signature scheme. In the following, we show that the generic transformation technique proposed in Sec. 3 can be used directly to solve this problem.

Let KG_{CL}, $\text{PartialKeyGen}_{CL}$, UserKeyGen_{CL}, Sign_{CL} and Vrfy_{CL} constitute a certificateless signature scheme SIG_{CL}. The transformation is described as below:

KG: $(mpk, msk) \leftarrow \text{KG}_{CL}(1^k)$.
PartialKeyGen: $partialkey[ID] \leftarrow \text{PartialKeyGen}_{CL}(msk, ID)$.
UserKeyGen: $(upk[ID], usk[ID]) \leftarrow \text{UserKeyGen}_{CL}(mpk, ID)$.
Sign: For a message m and identity ID, a signature σ is generated:

$$\begin{aligned} (vk_{OT}, sk_{OT}) &\leftarrow \text{KG}_{OT}(1^k) \\ \sigma_1 &\leftarrow \text{Sign}_{CL}(usk[ID], partialkey[ID], vk_{OT}) \\ \sigma_2 &\leftarrow \text{Sign}_{OT}(sk_{OT}, m\|ID\|\sigma_1) \\ \sigma &\leftarrow (\sigma_1, \sigma_2, vk_{OT}) \end{aligned}$$

Vrfy: Given master public key mpk, message m, identity ID, user public key $upk[ID]$ and signature $\sigma = (\sigma_1, \sigma_2, vk_{OT})$, $b_1 \wedge b_2$ is returned, where $b_1 \leftarrow \text{Vrfy}_{CL}(mpk, ID, upk[ID], \sigma_1, vk_{OT})$ and $b_2 \leftarrow \text{Vrfy}_{OT}(vk_{OT}, \sigma_2, m\|ID\|\sigma_1)$.

Theorem 2. *The certificateless signature scheme described above is $(t, q_s, q_o, \varepsilon)$-strongly unforgeable in Game-I (respectively, Game-II) if* SIG_{CL} *is $(t, q_s, q_o, \varepsilon/2)$-existentially unforgeable in Game-I (respectively, Game-II), and* SIG_{OT} *is a $(t, \varepsilon/2q_s)$-strong one-time signature scheme, where q_s and q_o are the maximum numbers of signing queries and all the other oracle queries, respectively.*

Similar to the proof of Theorem 1, to prove the strong unforgeability of the generic transformation in Game-I (respectively, Game-II), we distinguish between two events: (1) the forgery of $\mathcal{A}_I$ is valid and the one-time public key in the forgery appears in some previous answer of the signing queries; (2) the forgery is valid but the one-time public key in the forgery is new. For the first event, we can construct an efficient adversary $\mathcal{B}_{CL}$ to break the strong one-time unforgeability of SIG_{OT}. Note that, with the knowledge of master secret key msk, $\mathcal{B}_{CL}$ can answer queries to all the other oracles (i.e., CreateUser, RevealPartialKey, RevealSecretKey and ReplaceKey) besides the Signing oracle, and it can issue a signing query to its own oracle in this case. For the second event, we can construct an efficient adversary $\mathcal{C}_{CL}$ to break the existential unforgeability of SIG_{CL}. Detailed proof is similar to that of Theorem 1, so we omit it here.

In the generic transformation above, we include identity ID in the message when generating σ_2. This allows us to follow the two-event approach in the proof of Theorem 1 and therefore simplifies the proof for this theorem.

An ID-based signature scheme, introduced by Shamir [27], comprises four efficient algorithms, master key generation, KG, user secret key generation, Extract, signature generation, Sign and verification, Vrfy. Such a scheme is said to be $(t, q_s, q_e, \varepsilon)$-existentially unforgeable against adaptive chosen message and identity attacks if no adversary, which runs for time t and issues at most q_e Extract queries and at most q_s Signing queries, succeeds in forging a signature on a new identity-message pair with probability at least ε. Readers may refer to [4] for details.

Let SIG_{IBS}=(KG_{IBS}, $\mathrm{Extract}_{IBS}$, Sign_{IBS}, Vrfy_{IBS}) be an ID-based signature scheme. We can apply the same transformation technique described above to convert SIG_{IBS} to a strongly unforgeable one (i.e. preventing adversaries from forging new signatures on identity-message pairs that could have been signed previously). We omit the details of the transformation as it can easily be obtained from the above.

Similarly, we have the following theorem:

Theorem 3. *The new signature scheme obtained from the generic transformation in the setting of ID-based cryptography is $(t, q_s, q_e, \varepsilon)$-strongly existentially unforgeable against adaptive chosen message and identity attacks, provided that SIG_{IBS} is an ID-based signature scheme that is $(t, q_s, q_e, \varepsilon/2)$-existentially unforgeable against adaptive chosen message and identity attacks, and SIG_{OT} is a $(t, \varepsilon/2q_s)$-strong one-time signature scheme, where q_e and q_s are the maximum numbers of the Extract queries and Signing queries, respectively.*

The proof is similar to that of Theorem 1 and is omitted here.

4.2 Other Signatures

In our generic transformation and its extensions to certificateless and ID-based settings, we can see that our technique makes no modification on the internal of the original signature scheme, but uses it as a *black-box* to sign a freshly-generated one-time public key. This does not rely on any additional property of the original scheme except the existential unforgeability. Besides, our transformation does not modify the public/private key pair nor information concerning users' identities. Therefore, after describing the generic transformation in Sec. 3, the extensions to certificateless signature and ID-based signature become straightforward. We believe that this generic transformation technique can also be applied to other types of signature schemes, such as group signature [11], ring signature [25], proxy signature [20] and some others. We leave this as our further studies.

5 Strong One-Time Signature

The security of our generic transformation relies on the existence of strong one-time signature schemes. In this section, we show how to transform any one-time signature scheme which follows the 'one-way function paradigm' [14,15] to a strong one-time version. We also evaluate the performance of an instantiation which is based on a scheme by Reyzin and Reyzin [24].

5.1 From One-Time to Strong One-Time

Since the introduction of one-time signature [23,19], there have been many schemes of this type proposed, and many of them follow the *one-way function paradigm* [14,15]. Let $f : \{0,1\}^k \rightarrow \{0,1\}^\kappa$ be a one-way function. Informally, a scheme that follows the '*one-way function paradigm*' has the private key composed of a set of random elements from the domain of f, and the public key composed of evaluations of those private key elements using f. To generate a signature, a subset of private key elements is chosen in accordance with the message, and the subset is considered to be the signature, which can then be verified through evaluations of f and comparison with the corresponding elements in the public key. Below is an example from [15].

KG: On input 1^k, randomly select 2ℓ strings of length k, $s_1^0, s_1^1, \cdots, s_\ell^0, s_\ell^1$, where $\ell = \ell(k)$ for some polynomial $\ell : \mathbb{N} \rightarrow \mathbb{N}$, and compute $v_i^b = f(s_i^b)$, for $b = 0, 1$ and $i = 1, 2, \cdots, \ell$. The public key vk_{OT} is $((v_1^0, v_1^1), \cdots, (v_\ell^0, v_\ell^1))$ and the private key sk_{OT} is $((s_1^0, s_1^1), \cdots, (s_\ell^0, s_\ell^1))$.

Sign: For an ℓ-bit message $m = b_1 b_2 \cdots b_\ell$ where $b_i \in \{0, 1\}$ for $i = 1, \cdots, \ell$, the signature σ is $(s_1^{b_1}, \cdots, s_\ell^{b_\ell})$.

Vrfy: For message $m = b_1 b_2 \cdots b_\ell$ and signature $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_\ell)$, if $v_i^{b_i} = f(\sigma_i)$ for all $i = 1, 2, \cdots, \ell$, output 1; otherwise, output 0.

It is easy to show that the scheme above is a one-time signature scheme as any forgery on a new message would lead to the inversion of f. However, it is not ensured that no forgery can be made on a message that has already been signed.

To transform a one-time signature that follows the one-way function paradigm to a strong one-time signature (in the sense of Def. 3), a method is proposed in [15], which is based on Universal One-Way Hash Functions (UOWHF, in short) [21]. Although UOWHF can be constructed directly from one-way functions, the resulting strong one-time signature scheme suffers from a much larger public key, which includes the description of 2ℓ randomly selected UOWHFs in addition to the one-way-function evaluations of the 2ℓ private key elements.

To solve this problem, we propose another method. Our method is to replace f with a randomly selected collision-resistant hash function h. In this conversion, only the description of *one* (collision-resistant) hash function is added into the public key vk_{OT} rather than that of 2ℓ UOWHFs as in the method of [21]. On the security of our conversion, as the minimal assumption currently known for constructing a collision-resistant hash function is the existence of claw-free permutations, which is stronger than that of the existence of one-way functions, it follows that our generic transformation proposed in Sec. 3 is also based on the existence of claw-free permutations. In general, we use signature schemes to sign arbitrary-length messages. The standard technique, so-called '*hash-and-sign*' paradigm [12], we can use it to apply a collision-resistant hash function to the message and then sign the resulting hash value. Therefore, the existence of claw-free permutations is generally an assumption for the security of the original unforgeable signature schemes already.

The following theorem shows that our method yields a strong one-time signature scheme.

Theorem 4. *Let SIG'_{OT} be a secure one-time signature scheme that follows the one-way function paradigm, and let SIG_{OT} be the resulting scheme by replacing the one-way function f with a randomly selected hash function h. Then SIG_{OT} is strongly unforgeable against (adaptive) chosen one-message attack in the sense of Def. 3, provided that h is collison-resistant and preimage-resistant (or, one-way).*

Proof (Sketch). First, as the collision resistance of a hash function implies one-wayness (please also refer to the remark below), if we view the hash function h as a one-way function, then the new scheme SIG_{OT} is equivalent to SIG'_{OT}, thus is also unforgeable against one-time chosen message attacks (i.e., SIG_{OT} is also a one-time signature scheme).

Let $\mathcal{A}$ be an adversary which runs for time t and breaks the strong one-time unforgeability (in the sense of Def. 3) of SIG_{OT} with probability at least ε. Note that in general, if messages are ℓ bits long, by no means the signatures must have exactly ℓ private key components. Hence, we use another notation d to denote the number of private key components in a signature of SIG_{OT}. For example, in HORS [24], the value of d is much less than ℓ. Now, suppose $m = b_1 b_2 \cdots b_\ell$ is the ℓ-bit message in the query of $\mathcal{A}$ and $\sigma = (\sigma_1, \cdots, \sigma_d)$ is the signature on m answered by the Signing oracle, where $\sigma_i \in \{0,1\}^k$, for $i = 1, \cdots, d$. Let (m^*, σ^*) be $\mathcal{A}$'s forgery, where $m^* = b_1^* b_2^* \cdots b_\ell^*$ and $\sigma^* = (\sigma_1^*, \cdots, \sigma_d^*)$. Then either of the following events would occur with probability at least $\varepsilon/2$:

1. If $m^* \neq m$, there exists at least one i such that $b_i^* \neq b_i$. This would lead to the break of the preimage-resistance of function h.
2. If $m^* = m$, it must hold that $\sigma^* \neq \sigma$, which implies that there exists at least one i $(1 \leq i \leq d)$ such that $\sigma_i^* \neq \sigma_i$. Such a pair forms a collision for h. □

Remark: It is worthwhile to notice the relation between collision-resistance and one-wayness. Suppose h is a hash function compressing k-bit strings into κ-bit strings. According to [26], ε-collision-resistance of h implies $(2\varepsilon + 2^{\kappa-k})$-one-wayness. This implies that the input length k should be larger than the output length κ by a sufficient margin for ensuring the one-wayness of h.

5.2 An Instantiation of Strong One-Time Signature

Most of the current constructions of one-time signature follow the one-way function paradigm and are very efficient (with regard to time) as they do not carry out any public key cryptographic operation. One of the most efficient one-time signature schemes that follows the one-way function paradigm is the HORS proposed by Reyzin and Reyzin [24].

HORS is in fact an r-time signature scheme. A single public key can be used for r times, in contrast to only one time in a one-time signature scheme. The security of HORS relies on the existence of *Subset Resilient* functions [24]. For $r > 1$, realizing such functions using only conventional complexity-theoretic assumptions without random oracles is still an open problem. Fortunately, in our case, each key pair of HORS only needs to be used once (i.e. $r = 1$), and so collision-resistant hash families are enough for realizing the subset resilient functions.

Let k be the security parameter. Two auxiliary parameters, t and d, are chosen such that $d \cdot \log t = \ell$, where ℓ is the output length of a collision-resilient hash function $Hash$. The private key contains essentially t strings of k bits (along with the value of d) and the public key contains the evaluations of these t strings using a one-way function $f : \{0,1\}^k \rightarrow \{0,1\}^\kappa$. The signing process requires just one $Hash$ operation and produces a signature which is a sequence of d out of t strings of the private key. The verification process requires only d evaluations of f and one of $Hash$.

According to Theorem 4, to convert HORS into a strong one-time signature scheme, we can replace f with a randomly chosen collision-resistant (and one-way) hash function h. As suggested in [24], the security parameter k is set 80, and the output length κ of h is 160-bit. According to the remark at the end of Theorem 4, the one-wayness of h may not be ensured by collision-resistance in this case. Hence we suggest that a collision-resistant and one-way hash function h should be used. For example, we employ a collision-resistant hash function $h : \{0,1\}^{240} \rightarrow \{0,1\}^{160}$, that is, setting k to 240. We refer to the resulting scheme as '*strong HORS*'.

Public Key Size and Signature Size. In our generic transformation, signature size depends on the public key size and signature size of the underlying strong one-time scheme, but not on the private key size. Hence in the following, we only evaluate the public key size and signature size of *strong HORS*. As we can see, the public key is $t \cdot \kappa$-bit-long and the signature is $d \cdot k$-bit-long. These sizes could be large in practice, for example when $t = 256$ and $d = 20$ (as suggested in [24]). We note that almost all the current one-time signature schemes suffer from this drawback, and we believe that improving the signature size of a one-time signature scheme is of independent interest.

Remark: Also note that any strongly unforgeable signature scheme (in the sense of Def. 2) is also a strong one-time signature scheme (Def. 3). Hence the drawback mentioned above can easily be solved by using a strongly unforgeable signature scheme that has short public key and signature, such as the one in [7], to instantiate SIG_{OT} in our generic transformation. We should note that the tradeoff is on the computational efficiency.

6 Concluding Remarks

On-line/Off-line. As mentioned before, the signature generation of our generic transformation can be considered as a sequential composition of the original signature scheme and a strong one-time signature scheme. In the first phase, a one-time public key is freshly generated and signed to create a 'certificate'. In the second phase, the message and the 'certificate' are then signed using the strong one-time signature scheme. The first phase is independent of the message and therefore, can be used directly as the off-line part of an *on-line/off-line* signature scheme [14]. This phase can also be carried out for multiple times, each time, a triple $(vk_{OT}, sk_{OT}, \sigma_1)$ is generated and stored. When a message m is to be signed on-line, an unused triple $(vk_{OT}, sk_{OT}, \sigma_1)$ is selected and a signature σ_2 on $m\|\sigma_1$ is generated under sk_{OT} using the strong one-time signing algorithm. This yields a very efficient on-line operation. For example, the signing process of (strong) HORS [24] is essentially one hash evaluation.

In this paper, we proposed a universal and generic transformation which converts *any* unforgeable signature scheme into a strongly unforgeable one. It is universal in the sense that it can also be applied to signatures in other settings such as ID-based signature, certificateless signature and several others. Our technique does not add any additional parameter into the original public/private key pair and makes no change to the internals of the original signature scheme. On the conversion to strong one-time signature schemes that follow the one-way function paradigm, our method is efficient and does not introduce any additional security assumption in general. Due to the limitation of currently available one-time signature schemes, our generic transformation could have a large signature size when implemented, which is the main drawback. However, by choosing proper parameters and instantiations, our technique can be very efficient for practical use (Sec. 5.2). Finally, thanks to the efficient key generation, signing and verification processes of the strong one-time signature scheme, these make our transformation much more efficient than the comparable transformation techniques [15,9,29,30,6].

Based on the results in this paper, we conclude with the following two remarks, which have the theoretical and practical interests, respectively.

1. If one-way function exists, there exists a generic transformation to a strongly unforgeable signature scheme.
2. If we admit a non-standard assumption, such as Subset Intractability [24], or the existence of collision-free hash functions, then we can obtain a very efficient generic transformation to a strongly unforgeable signature scheme.

Acknowledgements

We are grateful to Man Ho Au, Ron Steinfeld and all the anonymous reviewers for their invaluable comments. We also thank the reviewers for referring us to the works of [30,6], and thank Mihir Bellare for sending us his PKC07 paper.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In Proc. ASIACRYPT 2003, pages 452473. Springer-Verlag, 2003. LNCS 2894.
2. J. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In Proc. EUROCRYPT 2002, pages 83107. Springer-Verlag, 2002. LNCS 2332.

3. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Proc. CRYPTO 2000, pages 255270. Springer-Verlag, 2000. LNCS 1880.
4. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identi cation and signature schemes. In Proc. EUROCRYPT 2004, pages 268 286. Springer-Verlag, 2004.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In First ACM Conference on Computer and Communications Security, pages 6273, Fairfax, 1993. ACM.
6. M. Bellare and S. Shoup. Tow-tier signatures, strongly unforgeable signatures, and at-shamir without random oracles. To appear in PKC 2007.
7. D. Boneh and X. Boyen. Short signatures without random oracles. In Proc. EUROCRYPT 2004, pages 5673, 2004. LNCS 3027.
8. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In Proc. CRYPTO 2004, pages 4155, 2004. LNCS 3152.
9. D. Boneh, E. Shen, and B. Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In Proc. of PKC 2006, pages 229240. Springer- Verlag, 2006. LNCS 3958.
10. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In Proc. EUROCRYPT 2004, pages 207222. Springer-Verlag, 2004. LNCS 3027.
11. D. Chaum and E. V. Heyst. Group signatures. In Proc. EUROCRYPT 91, pages 257265. Springer-Verlag, 1991. LNCS 547.
12. I. Damgard. Collision free hash functions and public key signature schemes. In Proc. EUROCRYPT 87, pages 203216. Springer-Verlag, 1988. LNCS 304.
13. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. SIAM J. Computing, 30(2):391437, 2000.
14. S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. J. of Cryptology, 9(1), 1996.
15. O. Goldreich. Foundations of Cryptography, volume II, Basic Applications. Cambridge University Press, 2004.
16. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. SIAM J. Computing, 17(2):281308, Apr. 1988.
17. B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng. Key replacement attack against a generic construction of certificateless signature. In Information Security and Privacy: 11th Australasian Conference, ACISP 2006, pages 235246. Springer-Verlag, 2006. LNCS 4058.
18. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In Proc. CRYPTO 2003, pages 110125. Springer-Verlag, 2003. LNCS 2729.
19. L. Lamport. Constructing digital signatures from a one way function. Technical Report Technical Report CSL-98, SRI International, Oct. 1979.
20. K. U. M. Mambo and E. Okamoto. Proxy signature: Delegation of the power to sign messages. IEICE Trans. Fundamentals, E79-A(9):13381353, 1996.
21. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In Proc. of 21st ACM Symposium on the Theory of Computing, pages 3343, 1989.
22. National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186, Nov. 1994.
23. M. O. Rabin. Digitalized signatures. Foundations of Secure Computation, pages 155168, 1978.

24. L. Reyzin and N. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In Information Security and Privacy: 7th Australasian Conference, ACISP 2002, pages 144153. Springer-Verlag, 2002. LNCS 2384.
25. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In Proc. ASIACRYPT 2001, pages 552565. Springer-Verlag, 2001. LNCS 2248.
26. P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Proc. Fast Software Encryption 2004, pages 371388. Springer-Verlag, 2004. LNCS 3017.
27. A. Shamir. Identity-based cryptosystems and signature schemes. In Proc. CRYPTO 84, pages 4753. Springer, 1984. LNCS 196.
28. A. Shamir and Y. Tauman. Improved online/offine signature schemes. In Proc. CRYPTO 2001, pages 355367. Springer, 2001. LNCS 2139.
29. R. Steinfeld, J. Pieprzyk, and H. Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In CT-RSA 2007, pages 357371. Springer-Verlag, 2007. LNCS 4377.
30. I. Teranishi, T. Oyama, and W. Ogata. General conversion for obtaining strongly existentially unforgeable signatures. In Proc. Progress in Cryptology - INDOCRYPT 2006, pages 191205. Springer-Verlag, 2006. LNCS 4329.

Efficient Generic On-Line/Off-Line Signatures Without Key Exposure

Xiaofeng Chen[1,3], Fangguo Zhang[2,3], Willy Susilo[4], and Yi Mu[4]

[1] Department of Computer Science,
Sun Yat-sen University, Guangzhou 510275, P.R. China
isschxf@mail.sysu.edu.cn

[2] Department of Electronics and Communication Engineering,
Sun Yat-sen University, Guangzhou 510275, P.R. China
isszhfg@mail.sysu.edu.cn

[3] Guangdong Key Laboratory of Information Security Technology
Guangzhou 510275, P.R. China

[4] Centre for Computer and Information Security Research,
School of Computer Science and Software Engineering,
University of Wollongong, Australia
{wsusilo,ymu}@uow.edu.au

Abstract. The "hash-sign-switch" paradigm was firstly proposed by Shamir and Tauman with the aim to design an efficient on-line/off-line signature scheme. However, all existing on-line/off-line signature schemes based on Shamir-Tauman's paradigm suffer from the key exposure problem of chameleon hashing. That is, if the signer applies the same hash value more than once to obtain two signatures on two different messages, the recipient can obtain a hash collision and use it to recover the signer's trapdoor information. Therefore, the signer should pre-compute and store plenty of different chameleon hash values and the corresponding signatures on the hash values in the off-line phase, and send the collision and the signature for a certain hash value in the on-line phase. Hence, the computation and storage cost for the off-line phase and the communication cost for the on-line phase in Shamir-Tauman's signature scheme are still a little more overload.

In this paper, we first introduce a special double-trapdoor hash family based on the discrete logarithm assumption to solve this problem. We then apply the "hash-sign-switch" paradigm to propose a much more efficient generic on-line/off-line signature scheme. Additionally, we use a one-time trapdoor/hash key pair for each message signing, which prevents the recipient from recovering the trapdoor information of the signer and computing other collisions.

Keywords: On-line/off-line signatures, Chameleon hashing, Key exposure.

1 Introduction

The notion of on-line/off-line signatures was introduced by Even, Goldreich and Micali [10,11]. It performs the signature generating procedure in two phases.

The first phase is performed *off-line* (without knowing the signed message) and the second phase is performed *on-line* (after knowing the signed message). On-line/off-line signatures are particularly useful in smart card applications: The off-line phase is implemented either during the card manufacturing process or as a background computation whenever the card is connected to power, and the on-line phase uses the stored result of the off-line phase to sign actual messages. The on-line phase is typically very fast, and hence can be extended efficiently even on a weak processor.

Even, Goldreich and Micali proposed a general method for converting any signature scheme into an on-line/off-line signature scheme. However, the method is not practical because it increases the size of the signature by a quadratic factor. In Crypto 2001, Shamir and Tauman [22] used the so called "chameleon hash functions" to develop a new paradigm, named "hash-sign-switch", for designing much more efficient on-line/off-line signature schemes.

Chameleon hash functions, first introduced by Krawczyk and Rabin [16], are trapdoor one-way hash functions which prevent everyone except the holder of the trapdoor information from computing the collisions for a randomly given input. Chameleon hash functions were originally used to design chameleon signatures, which simultaneously provide non-repudiation and non-transferability for the signed message as undeniable signatures [7] do. In the chameleon signature schemes, the recipient is the holder of trapdoor information, while in case of on-line/off-line signatures, the signer is the holder of the trapdoor information. Therefore, in the off-line phase the signer generates a signature σ by using a provably secure signature scheme to sign the chameleon hash value $h(m', r')$ of a random message m' and a random auxiliary number r'. In the on-line phase, the signer computes a collision r of the chameleon hash function for the given message m such that $h(m, r) = h(m', r')$. The signature for the message m is the pair (σ, r).

In the Shamir-Tauman's on-line/off-line signature schemes, one limitation is that the signature for the different messages must use different chameleon hash values. Otherwise, if the signer uses the same hash value twice to obtain two signatures on two different messages, the recipient can obtain a hash collision and use it to recover the signer's trapdoor information, *i.e.*, the private key. To avoid this problem, the signer must compute and store plenty of different chameleon hash values and the corresponding signatures on the hash values in the off-line phase. Given a signed message in the on-line phase, the signer first chooses a one-time hash value, and then computes a hash collision for the hash value. He then sends the hash collision and the corresponding signature to the recipient. Hence, the computation and storage cost for the off-line phase and the communication cost for the on-line phase in Shamir-Tauman's signature scheme are still a little more overload.

In this paper, for the first time in the literature, we address this problem by introducing a double-trapdoor hash family based on the discrete logarithm assumption and then apply the "hash-sign-switch" paradigm to propose a much more efficient generic on-line/off-line signature scheme. In our signature scheme, the hash value and the corresponding signature are always identical and can be viewed as the public key of the signer. Hence, it is not required to compute and store them in the off-line phase. Additionally, we introduce the idea of *long-term* trapdoor and *one-time* trapdoor in our chameleon hash families, which is similar to the idea of *master* trapdoor and *specific* trapdoor in the multi-trapdoor commitment schemes [13]. The *one-time* trapdoor is used only once for each message signing in the on-line phase, which prevents the recipient from recovering the trapdoor information of the signer and computing other collisions.

In order to achieve the communication and computation advantages of our on-line/off-line signature scheme, we adopt elliptic curve cryptosystems [15,19] to present our double-trapdoor hash family. Certainly, we can design such a double-trapdoor hash family over other generic groups, *e.g.*, the subgroup of $\mathbb{Z}_p^*$. However, we argue that such a double-trapdoor hash family over generic groups is unsuitable for designing *efficient* generic on-line/off-line signature schemes. The reason is as follows: Since the "hash-sign-switch" paradigm is a generic method, it is required that any provably secure signature scheme $\mathcal{S}$ can be used to design the on-line/off-line signature scheme. However, only when the signature length of original signature scheme $\mathcal{S}$ is less than that of a group element, our proposed on-line/off-line signature scheme is superior to Shamir-Tauman's scheme in communication cost.[1] Currently, for any provably secure signature scheme, the signature length is more than 160 bits. Therefore, the elliptic curve cryptosystems seem to be the optimal choice. If we adopt other generic group such as the subgroup of $\mathbb{Z}_p^*$, many signature schemes including some short signature schemes [3,5] can not be used to design our on-line/off-line signature scheme. For more details, please refer to Section 5.2.

1.1 Related Works

As noted in [22], some signature schemes such as Fiat-Shamir, Schnorr, and ElGamal signature schemes [12,21,9] can be naturally partitioned into on-line and off-line phases. The reason is that the first step in these signature schemes does not depend on the given message, and can thus be carried out off-line. However, these are particular schemes with special structure and specific security assumptions rather than a general and provably secure conversion technique for arbitrary signature schemes. Shamir and Tauman introduced the "hash-sign-switch" method for simultaneously improving both the security and the real-time efficiency of any signature scheme by converting it into an efficient on-line/off-line signature scheme. Generally, a new chameleon hash family results in a new on-line/off-line signature scheme. Recently, some variants of on-line/off-line signature schemes [6,17] have been proposed based on Shamir-Tauman's general construction.

However, it seems that all existing on-line/off-line signature schemes based on Shamir-Tauman's paradigm suffer from the key exposure problem of chameleon

[1] In any case, our proposed scheme is no inferior to Shamir-Tauman's scheme in computation and storage cost.

hashing. This problem is firstly addressed by Ateniese and de Medeiros [1] in the original chameleon signature schemes. Chen *et al.* [8] proposed the first full construction of a chameleon hash function without key exposure. Later, Ateniese and de Medeiros presented several constructions of exposure-free chameleon hash functions based on different cryptographic assumptions [2]. However, to the best of our knowledge, there seems to be no existing work that solves the key exposure problem in the generic on-line/off-line signature schemes.

1.2 Organization

The rest of the paper is organized as follows: Some preliminaries are provided in Section 2. The new double-trapdoor chameleon hash family based on the discrete logarithm assumption is presented in Section 3. Our efficient generic on-line/off-line signature scheme is given in Section 4. The security and efficiency analysis of our scheme are given in Section 5. Finally, conclusions will be made in Section 6.

2 Preliminaries

In this section, we introduce the basic notion of chameleon hash family and Shamir-Tauman's "hash-sign-switch" paradigm [22].

2.1 Chameleon Hash Family

Definition 1. *(chameleon hash family) A chameleon hash family consists of a pair* $(\mathcal{I}, \mathcal{H})$*:*

- $\mathcal{I}$ *is a probabilistic polynomial-time key generation algorithm that on input* 1^k*, outputs a pair* (HK, TK) *such that the sizes of* HK, TK *are polynomially related to* k*.*
- $\mathcal{H}$ *is a family of randomized hash functions. Every hash function in* $\mathcal{H}$ *is associated with a hash key* HK*, and is applied to a message from a space* $\mathcal{M}$ *and a random element from a finite space* $\mathcal{R}$*. The output of the hash function* H_{HK} *does not depend on* TK*.*

A chameleon hash family $(\mathcal{I}, \mathcal{H})$ has the following properties:

1. *Efficiency:* Given a hash key HK and a pair $(m, r) \in \mathcal{M} \times \mathcal{R}$, $H_{HK}(m, r)$ is computable in polynomial time.
2. *Collision resistance:* There is no probabilistic polynomial time algorithm $\mathcal{A}$ that on input HK outputs, with a probability which is not negligible, two pairs $(m_1, r_1), (m_2, r_2) \in \mathcal{M} \times \mathcal{R}$ that satisfy $m_1 \neq m_2$ and $H_{HK}(m_1, r_1) = H_{HK}(m_2, r_2)$ (the probability is over HK, where $(HK, TK) \leftarrow \mathcal{I}(1^k)$, and over the random coin tosses of algorithm $\mathcal{A}$).
3. *Trapdoor collisions:* There exists a probabilistic polynomial time algorithm that given a pair $(HK, TK) \leftarrow \mathcal{I}(1^k)$, a pair $(m_1, r_1) \in \mathcal{M} \times \mathcal{R}$, and an additional message $m_2 \in \mathcal{M}$, outputs a value $r_2 \in \mathcal{R}$ such that:
   - $H_{HK}(m_1, r_1) = H_{HK}(m_2, r_2)$.
   - If r_1 is uniformly distributed in $\mathcal{R}$ then the distribution of r_2 is computationally indistinguishable from uniform in $\mathcal{R}$.

2.2 Shamir-Tauman's "Hash-Sign-Switch" Paradigm

Shamir and Tauman introduced the following "hash-sign-switch" paradigm to get a generic on-line/off-line signature scheme.

- **System Parameters Generation:** Let $(\mathcal{I}, \mathcal{H})$ be any trapdoor hash family and $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ be any provably secure signature scheme. The system parameters are $SP = \{(\mathcal{I}, \mathcal{H}), (\mathcal{G}, \mathcal{S}, \mathcal{V})\}$.
- **Key Generation Algorithm:**
  - On input 1^k, run the key generation algorithm of the original signature scheme $\mathcal{G}$ to obtain a signing/verification key pair (SK, VK).
  - On input 1^k, run the key generation algorithm of the trapdoor hash family $(\mathcal{I}, \mathcal{H})$ to obtain a hash/trapdoor key pair (HK, TK).

  The signing key is (SK, TK) and the verification key is (VK, HK).
- **The Signing Algorithm:**
  1. Off-line phase
     - Choose at random $(m_i, r_i) \in_R \mathcal{M} \times \mathcal{R}$, and compute the chameleon hash value $h_i = H_{HK}(m_i, r_i)$.
     - Run the signing algorithm $\mathcal{S}$ with the signing key SK to sign the message h_i. Let the output be $\sigma_i = \mathcal{S}_{SK}(h_i)$.
     - Store the pair (m_i, r_i), and the signature σ_i.
  2. On-line phase
     - For a given message m, retrieve from the memory a random pair (m_i, r_i) and the signature σ_i.
     - Compute $r \in \mathcal{R}$ such that $H_{HK}(m, r) = H_{HK}(m_i, r_i)$.
     - Send (r, σ_i) as the signature of the message m.
- **The Verification Algorithm:**
  - Compute $h_i = H_{HK}(m, r)$.
  - Verify that σ_i is indeed a signature of the hash value h_i with respect to the verification key VK.

In the following, we present Shamir-Tauman's "hash-sign-switch" paradigm with elliptic curve analogue of the chameleon hash family based on the discrete logarithm assumption [16,22], so that we can fairly compare it with our proposed signature scheme.

- **System Parameters Generation:** Let t be a prime power, and $E(\mathbb{F}_t)$ an elliptic curve over finite field $\mathbb{F}_t$. Let $\#E(\mathbb{F}_t)$ be the number of points of $E(\mathbb{F}_t)$, and P be a point of $E(\mathbb{F}_t)$ with prime order q where $q|\#E(\mathbb{F}_t)$. Denote $\mathbb{G}$ the subgroup generated by P. Let $(\mathcal{I}, \mathcal{H})$ be the trapdoor hash family based on the discrete logarithm assumption and $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ be any provably secure signature scheme. The system parameters are $SP = \{E, t, q, P, \mathbb{G}, (\mathcal{G}, \mathcal{S}, \mathcal{V})\}$.
- **Key Generation Algorithm:**
  - On input 1^k, run the key generation algorithm of the original signature scheme $\mathcal{G}$ to obtain the signing/verification key pair (SK, VK).

  - On input 1^k, run the key generation algorithm of the trapdoor hash family $(\mathcal{I}, \mathcal{H})$ to obtain the hash/trapdoor key pair $(Y = xP, x)$.

  The signing key is (SK, x) and the verification key is (VK, Y).[2]

– **The Signing Algorithm:**
  1. Off-line phase
     - Choose at random $(m_i, r_i) \in_R \mathcal{M} \times \mathcal{R}$, and computes the chameleon hash value $h_i = H_Y(m_i, r_i) = m_iP + r_iY$.
     - Run the signing algorithm $\mathcal{S}$ with the signing key SK to sign the message h_i. Let the output be $\sigma_i = \mathcal{S}_{SK}(h_i)$.
     - Store the pair (m_i, r_i), and the signature σ_i.
  2. On-line phase
     - For a given message m, retrieve from the memory x^{-1} and a random pair (m_i, r_i).
     - Compute $r = x^{-1}(m_i - m) + r_i \mod q$.
     - Send (r, σ_i) as the signature of the message m.

– **The Verification Algorithm:**
  - Compute $h_i = H_Y(m, r) = mP + rY$.
  - Verify that σ_i is indeed a signature of the hash value h_i with respect to the verification key VK.

3 A Double-Trapdoor Chameleon Hash Family

Chameleon hashing is very closely related to chameleon commitment schemes [4]. Gennaro [13] first introduced the notion of multi-trapdoor commitments. Ateniese and de Medeiros [2] observed that any stateless trapdoor commitment with two trapdoors may be adequate for designing a chameleon hash scheme without key exposure, which can be used to design a chameleon signature scheme. However, it seems that the current chameleon hash schemes without key exposure are not suitable for designing efficient on-line/off-line signature schemes. The reasons are twofold: Firstly, collision computation in these chameleon hash schemes usually requires the costly modular exponentiation operation. Secondly, though collision forgery will not reveal the signer's trapdoor information, it allows the verifier to compute other collisions for the same hash value.[3]

In this section, we first propose a new double-trapdoor chameleon hash family based on the discrete logarithm assumption as follows, which is a main ingredient for designing our efficient on-line/off-line signature scheme.

[2] The value of x^{-1} should be pre-computed and stored in order to decrease the computation cost in the on-line phase of the signature scheme.

[3] Note that this feature has some advantages in the chameleon signatures. For example, the signer can provide a different collision to hide the original signed message. While in the case of on-line/off-line signatures, it means that the verifier can universally forge a signature of the signer.

- **System Parameters Generation:** Let t be a prime power, and $E(\mathbb{F}_t)$ an elliptic curve over finite field $\mathbb{F}_t$. Let $\#E(\mathbb{F}_t)$ be the number of points of $E(\mathbb{F}_t)$, and P be a point of $E(\mathbb{F}_t)$ with prime order q where $q|\#E(\mathbb{F}_t)$. Denote by $\mathbb{G}$ the subgroup generated by P. Define a cryptographic secure hash function $f : \mathbb{Z}_q \times \mathbb{G} \to \mathbb{Z}_q$. Choose two random elements $k, x \in_R \mathbb{Z}_q^*$, and compute $K = kP, Y = xP$. The public hash key is $HK = (K, Y)$, and the private trapdoor key is $TK = (k, x)$.
- **The Hash Family:** Given the hash key HK, the proposed chameleon hash function $H_{HK} : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{G}$ is defined as follows:

$$H_{HK}(m, r) = f(m, K) \cdot K + rY.$$

Theorem 1. *The construction above is a chameleon hash family under the assumption of the discrete logarithm problem in G is intractable.*

Proof. We prove that the scheme satisfies the properties defined in Section 2.1.

1. *Efficiency:* Given the hash key HK and a pair $(m, r) \in \mathbb{Z}_q \times \mathbb{Z}_q$, $H_{HK}(m, r) = f(m, K) \cdot K + rY$ is computable in polynomial time.
2. *Collision resistance:* Assume to the contrary, that there exists a polynomial time algorithm $\mathcal{A}$ that on input HK outputs, with a probability which is not negligible, two pairs $(m_1, r_1), (m_2, r_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$ that satisfy $m_1 \neq m_2$ and $H_{HK}(m_1, r_1) = H_{HK}(m_2, r_2)$. Then, we can use $\mathcal{A}$ to solve the discrete logarithm problem in $\mathbb{G}$ as follows: For a randomly given instance (P, aP), choose a random integer $b \in_R \mathbb{Z}_q$ and define $K = aP$, and $Y = bP$. Therefore, if
$$f(m_1, aP) \cdot aP + r_1 Y = f(m_2, aP) \cdot aP + r_2 Y,$$
we can compute $a = (f(m_1, aP) - f(m_2, aP))^{-1}(r_2 - r_1)b \mod q$.
3. *Trapdoor collisions:* Assume that we are given the hash and trapdoor key pair (HK, TK), a pair $(m_1, r_1) \in \mathbb{Z}_q \times \mathbb{Z}_q$, and an additional message $m_2 \in \mathbb{Z}_q$, we want to find $r_2 \in \mathbb{Z}_q$ such that
$$f(m_1, kP) \cdot kP + r_1 Y = f(m_2, kP) \cdot kP + r_2 Y.$$
The value of r_2 can be computed in polynomial time as follows:
$$r_2 = r_1 + kx^{-1}(f(m_1, kP) - f(m_2, kP)) \mod q.$$
Also, if r_1 is uniformly distributed in $\mathcal{R}$ then the distribution of r_2 is computationally indistinguishable from uniform in $\mathcal{R}$. □

4 Our Efficient On-Line/Off-Line Signature Scheme

In this section, we apply the "hash-sign switch" paradigm to propose a much more efficient on-line/off-line signature scheme. We can adopt any provably secure digital signature scheme to design our on-line/off-line signature scheme, so it is a general construction. The main idea is that the hash value and the corresponding signature in the signature scheme are always identical and can be viewed as the public key of the signer. Hence, it is not required to compute and store them in the off-line phase. However, if we directly use the proposed double-trapdoor chameleon hash function to design the on-line/off-line signature scheme, the key exposure problem still arises.

We introduce the idea of *long-term* trapdoor and *one-time* trapdoor in our chameleon hash family. The *one-time* trapdoor is used **only once** for each message signing in the on-line phase, which prevents the recipient from recovering the trapdoor information of the signer and computing other collisions. The *long-term* trapdoor can be used repeatedly during its life span.

The proposed on-line/off-line signature scheme consists of the following efficient algorithms:

– **System Parameters Generation:** Let t be a prime power, and $E(\mathbb{F}_t)$ an elliptic curve over finite field $\mathbb{F}_t$. Let $\#E(\mathbb{F}_t)$ be the number of points of $E(\mathbb{F}_t)$, and P be a point of $E(\mathbb{F}_t)$ with prime order q where $q|\#E(\mathbb{F}_t)$. Denote $\mathbb{G}$ the subgroup generated by P. Define a cryptographic secure hash function $f : \mathbb{Z}_q \times \mathbb{G} \to \mathbb{Z}_q$. Given a hash key $HK = (K, Y)$, the chameleon hash function $H_{HK} : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{G}$ is defined as follows:

$$H_{HK}(m, r) = f(m, K) \cdot K + rY.$$

Let $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ be any provably secure signature scheme. The system parameters are $SP = \{E, t, q, P, \mathbb{G}, f, H_{HK}, (\mathcal{G}, \mathcal{S}, \mathcal{V})\}$.

– **Key Generation Algorithm:**
  - On input 1^k, run the key generation algorithm of the original signature scheme $\mathcal{G}$ to obtain the signing/verification key pair (SK, VK).
  - On input 1^k, run the key generation algorithm of the trapdoor hash family to obtain the *long-term* hash/trapdoor key pair, denote by $HK = Y = xP, TK = x$.
  - Choose at random $k^* \in_R \mathbb{Z}_q$, and compute the chameleon hash value $h = k^*Y$. Run the signing algorithm $\mathcal{S}$ with the signing key SK to sign the message h. Let the output be $\sigma = \mathcal{S}_{SK}(h)$.

  The signing key is (SK, x, k^*) and the verification key is (VK, Y, σ).

– **The Signing Algorithm:**
  1. Off-line phase
     - Choose at random $k_i \in_R \mathbb{Z}_q$, and computes $k_i x^{-1} \mod q$ and $k_i P$.
     - Store the *one-time* trapdoor/hash key pair $(k_i x^{-1}, k_i P)$.
  2. On-line phase
     - For a given signed message m_i, retrieve from the memory a random pair $(k_i x^{-1}, k_i P)$.
     - Compute $r_i = k^* - f(m_i, k_i P) k_i x^{-1} \mod q$.
     - Send $(r_i, k_i P)$ as the signature of the message m_i.

- **The Verification Algorithm:**
  - Compute $h = f(m_i, k_iP)k_iP + r_iY$ by using the *one-time* hash key k_iP and the *long-term* hash key Y.
  - Verify that σ is indeed a signature of the hash value h with respect to the verification key VK.

Note that

$$\begin{aligned} h = &\ f(m_i, k_iP)k_iP + r_iY \\ = &\ f(m_i, k_iP)k_iP + (k^* - f(m_i, k_iP)k_ix^{-1})Y \\ = &\ k^*Y \end{aligned}$$

The proposed scheme satisfies the property of completeness.

Remark 1. We argue that the value of x^{-1} should be pre-computed and stored in both our scheme and Shamir-Tauman's scheme.

Note that $r_i = k^* - f(m_i, k_iP)k_ix^{-1} \mod q$, it also requires only 1 modular multiplication of $\mathbb{Z}_q$ in the on-line phase of our scheme since k_ix^{-1} is stored in the off-line phase.

Remark 2. Note that in our proposed on-line/off-line signature scheme, the hash key $K_i = k_iP$ is used only **once** for signing a message m_i, while the other hash key $Y = xP$ can be used repeatedly. This is why we named them the *one-time* hash key and the *long-term* hash key, respectively.

5 Analysis of the Proposed Schemes

5.1 Security

The most general known security notion of a signature scheme is security against existential forgery on adaptively chosen message attacks, which was firstly defined by Goldwasser, Micali and Rivest [14] as follows:

Definition 2. *A signature scheme $\Omega = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is existentially unforgeable under adaptive chosen message attacks if for any probabilistic polynomial time adversary $\mathcal{A}$ there exist no non-negligible probability ϵ such that*

$$\boldsymbol{Adv}(\mathcal{A}) = \Pr \left[\begin{array}{l} \langle pk, sk \rangle \leftarrow \mathsf{Gen}(1^l); \\ for\ \ i = 1, 2, \ldots, k; \\ m_i \leftarrow \mathcal{A}(pk, m_1, \sigma_1, \ldots, m_{i-1}, \sigma_{i-1}), \sigma_i \leftarrow \mathsf{Sign}(sk, m_i); \\ \langle m, \sigma \rangle \leftarrow \mathcal{A}(pk, m_1, \sigma_1, \ldots, m_k, \sigma_k); \\ m \notin \{m_1, \ldots, m_k\} \wedge \mathsf{Ver}(pk, m, \sigma) = accept \end{array} \right] \geq \epsilon.$$

Now we give the formal security proof of our on-line/off-line signature scheme. More precisely, we have the following theorem:

Theorem 2. *In the random oracle model, the resulting on-line/off-line signature scheme is existentially unforgeable against adaptive chosen message attacks, provided that the discrete logarithm problem in G is intractable.*

Proof. In our proposed on-line/off-line signature scheme, the corresponding signature σ on the chameleon hash value h is viewed as the public key of the signer. Therefore, a hash collision r and a one-time hash key kP are the real signature on the message m.

Suppose that $\mathcal{A}$ is a probabilistic algorithm that given a verification key (VK, HK, σ), forges a signature with respect to the proposed on-line/off-line signature scheme by an adaptively chosen message attack in time T with success probability ϵ. We denote respectively by q_H and q_S the number of queries that $\mathcal{A}$ can at most ask to the hash oracle and the signing oracle. Let $(m_i, K_i = k_iP)$ denote the input of i-th query to the hash oracle, and e_i denote the corresponding answer to it. Let m_j denote the j-th query to the signing oracle, and (r'_j, K'_j) denote the corresponding signatures produced by the signing oracle. Let (m, r, kP) denote the output of $\mathcal{A}$. Since the success probability of $\mathcal{A}$ is ϵ, it follows that

$$Pr[V_{VK}(h, \sigma) = 1 \wedge h = H_{HK,kP}(m, r) = H_{HK,k_iP}(m_i, r_i)] \geq \epsilon.$$

Then we can construct a probabilistic algorithm $\mathcal{M}$ to compute a for a randomly given instance (P, aP) where P is a generator of $\mathbb{G}$ as follows:

- Let (SK, VK) be the signing/verification key pair of the original signature scheme. Choose a random integer $b \in_R \mathbb{Z}_q$, and let $HK = Y = bP$. Define the chameleon hash value $h = b \cdot aP$. Run the signing algorithm S with the signing key SK to sign the message h. Let the output be $\sigma = S_{SK}(h)$. Publish the pair (VK, Y, σ).
- Maintain a list, called f-list, which is initially set to empty. If the i-th query (m_i, K_i) to the hash oracle f is not in the list, choose a random element $e_i \in_R \mathbb{Z}_q$ and respond it as the answer of i-th query. Then add (m_i, K_i, e_i) to the f-list.
- Let m_j denote the input of j-th query to the signing oracle, choose at random $(e'_j, r'_j) \in_R \mathbb{Z}_q \times \mathbb{Z}_q$ (Note that e'_j is not in the f-list) and define

$$K'_j = e'^{-1}_j(h - r'_jY),$$

respond e'_j as the hash oracle answer to the query (m_j, K'_j), and (K'_j, r'_j) as the signing oracle answer to the query m_j. Then add (m_j, K'_j, e'_j) to the f-list.

Suppose the output of $\mathcal{A}$ is (m, K, r). If $m \neq m_j$ for $j = 1, ..., q_S$ and $h = f(m, K)K + rY$, we say that $\mathcal{A}$ forges a signature (K, r) on the message m with respect to the proposed on-line/off-line signature scheme.

By replays of $\mathcal{A}$ with the same random tape but different choices of oracle f, as done in the Forking Lemma [20], we can obtain two valid signatures (m, K, r) and (m, K, r') with respect to different hash oracles f and f'.

Note that $h = f(m, K)K + rY$ and $h = f'(m, K)K + r'Y$, we can compute $a = (f'(m, K) - f(m, K))^{-1}(f'(m, K)r - f(m, K)r')$ as the discrete logarithm of aP with respect to the base P.

The success probability of $\mathcal{M}$ is also ϵ, and the running time of $\mathcal{M}$ is equal to the running time of the Forking Lemma which is bounded by $23Tq_R/\epsilon$ [20]. □

5.2 Efficiency

We compare the efficiency of our scheme with that of Shamir-Tauman's scheme given in Section 2.2. We denote by $C(\theta)$ the computation cost of operation θ, and by $|\lambda|$ the bits of λ. Also, we denote by M a scalar multiplication in $\mathbb{G}$, by SM a simultaneous scalar multiplication of the form $\lambda P + \mu Q$ in $\mathbb{G}$, and by m the modular multiplication in $\mathbb{Z}_q$. We omit other operations such as point addition and hash in both schemes.

Table 1 and Table 2 present the comparison of the computation cost, the storage cost, and the communication cost for each message signing between Shamir-Tauman's scheme and our scheme.

Table 1. Comparison of the computation cost

	Shamir-Tauman's scheme	Our scheme
Off-line phase	$1C(h) + 1C(\sigma)$ $= 1SM + 1C(\sigma)$	$1C(kP) + 1C(kx^{-1})$ $= 1M + 1m$
On-line phase	$1m$	$1m$

Table 2. Comparison of the storage and communication cost

	Shamir-Tauman's scheme	Our scheme
Storage off-line phase	$2\|q\| + 1\|\sigma\|$	$1\|q\| + 1\|t\| + 1$
Communication on-line phase	$1\|q\| + 1\|\sigma\|$	$1\|q\| + 1\|t\| + 1$

Since a 160-bit ECC key offers more or less the same level of security as a 1024-bit RSA key [18], we let $|q|$=160 in the following. Currently, for any secure signature scheme, the signature length $|\sigma| \geq |t| + 1$ since $|t|$ is about 160 (In the optimal case, we can choose an elliptic curve $E(\mathbb{F}_t)$ such that $\#E(\mathbb{F}_t)$ is just a 160-bit prime q. From Hasse theorem, we know that $|t| = |\#E(\mathbb{F}_t)| = 160$). Therefore, the proposed scheme is much superior to Shamir-Tauman's scheme in the computation cost of off-line phase, storage cost and communication cost, while the computation cost in the on-line phase is same. So, we argue that our signature scheme is more suitable for smart-card applications where both the computation and storage resources are limited.

Remark 3. However, if we adopt other generic group such as the subgroup of $\mathbb{Z}_p^*$ to present our double-trapdoor chameleon hash family and on-line/off-line signature scheme, the communication cost for our on-line/off-line signature scheme is $1|q| + 1|p|$. For most current signature schemes, the signature length $|\sigma| < |p|$ if we let $|p| = 1024$. So, our proposed on-line/off-line signature scheme is inferior to Shamir-Tauman's scheme in communication cost since $1|q|+1|p| > 1|q|+1|\sigma|$. This is the reason why we choose the elliptic curve cryptosystems.

6 Conclusions

On-line/off-line signatures are particularly useful in smart card applications. In this paper, we first introduce a special double-trapdoor chameleon hash family based on the discrete logarithm assumption and then apply the "hash-sign-switch" paradigm to propose a much more efficient generic on-line/off-line signature scheme. Compared with Shamir-Tauman's signature scheme, the advantages of our signature scheme are the lower computation and storage cost for the off-line phase, and the lower communication cost for the on-line phase.

Acknowledgement

We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by National Natural Science Foundation of China (No. 60503006, 60403007 and 60633030), Natural Science Foundation of Guangdong, China (No. 04205407), 973 Program (2006CB303104) and ARC Discovery Grant DP0663306.

References

1. G. Ateniese and B. de Medeiros, *Identity-based chameleon hash and applications*, Finacial Cryptography and Data Security-FC 2004, LNCS 3110, pp.164-180, Springer-Verlag, 2004.
2. G. Ateniese and B. de Medeiros, *On the key-exposure problem in chameleon hashes*, Proceeding of the 4th Conference on Security in Communication Networks-SCN 2004, LNCS 3352, pp.165-179, Springer-Verlag, 2005.
3. D. Boneh and X. Boyen, *Short signatures without random oracles*, Advances in Cryptology-Eurocrypt 2004, LNCS 3027, pp.56-73, pringer-Verlag, 2004.
4. G. Brassard, D. Chaum, and C. Crepeau, *Minimum disclosure proofs of knowledge*, Journal of Computer and System Sciences, 37(2), pp.156-189, 1988.
5. D. Boneh, B. Lynn, and H. Shacham, *Short signatures from the Weil pairings*, Advances in Cryptology-Asiacrypt 2001, LNCS 2248, pp.514-532, Springer-Verlag, 2001.
6. C. Crutchfield, D. Molnar, D. Turner, and D. Wagner, *Generic on-line/off-line threshold signatures*, Proceeding of the 9th International Conference on Theory and Practice in Public-Key Cryptography-PKC 2006, LNCS 3958, pp.58-74, Springer-Verlag, 2006.
7. D. Chaum and H. van Antwerpen, *Undeniable signatures*, Advances in Cryptology-Crypto 1989, LNCS 435, pp.212-216, Springer-Verlag, 1989.
8. X. Chen, F. Zhang, and K. Kim, *Chameleon hashing without key exposure*, Proceeding of the 7th International Information Security Conference-ISC 2004, LNCS 3225, pp.87-98, Springer-Verlag, 2004.
9. T. ElGamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, 31(4), pp.469-472, 1985.
10. S. Even, O. Goldreich, and S. Micali, *On-line/Off-line digital signatures*, Advances in Cryptology-Crypto 1989, LNCS 2442, pp.263-277, Springer-Verlag, 1989.

11. S. Even, O. Goldreich, and S. Micali, *On-line/Off-line digital signatures*, Journal of Cryptology, 9(1), pp.35-67, Springer-Verlag, 1996.
12. A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, Advances in Cryptology-Crypto 1986, LNCS 263, pp. 186-194, Springer-Verlag, 1986.
13. R. Gennaro, *Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks*, Advances in Cryptology-Crypto 2004, LNCS 3152, pp.220-236, Springer-Verlag, 2004.
14. S. Goldwasser, S. Micali, and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal on Computing, 17(2), pp.281-308, 1988.
15. N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, 48(177), pp.203-209, 1987.
16. H. Krawczyk and T. Rabin, *Chameleon hashing and signatures*, Proceeding of Network and Distributed System Security 2000, pp.143-154, 2000.
17. K. Kurosawa and K. Schmidt-Samoa, *New on-line/off-line signature schemes without random oracles*, Proceeding of the 9th International Conference on Theory and Practice in Public-Key Cryptography-PKC 2006, LNCS 3958, pp.330-346, Springer-Verlag, 2006.
18. A.K. Lenstra and E.R. Verheul, *Selecting cryptographic key sizes*, Journal of Cryptology, 14(4), pp.255-293, Springer-Verlag, 2001.
19. V. Miller, *Uses of elliptic curves in cryptography*, Advances in Cryptology-Crypto 1985, LNCS 218, pp.417-426, Springer-Verlag, 1986.
20. D. Pointcheval and J. Stern, *Security arguments for digital signatures and blind signatures*, Journal of Cryptography, 13(3), pp.361-396, Springer-Verlag, 2000.
21. C. P. Schnorr, *Efficient signature generation for smart cards*, Journal of Cryptology, 4(3), pp.239-252, Springer-Verlag, 1991.
22. A. Shamir and Y. Tauman, *Improved online/offline signature schemes*, Advances in Cryptology-Crypto 2001, LNCS 2139, pp.355-367, Springer-Verlag, 2001.

Merkle Signatures with Virtually Unlimited Signature Capacity

Johannes Buchmann[1], Erik Dahmen[1], Elena Klintsevich[1], Katsuyuki Okeya[2], and Camille Vuillaume[2]

[1] Technische Universität Darmstadt
Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
{buchmann,dahmen,klintsev}@cdc.informatik.tu-darmstadt.de
[2] Hitachi, Ltd., Systems Development Laboratory,
1099, Ohzenji, Asao-ku, Kawasaki-shi, Kanagawa-ken, 215-0013, Japan
{katsuyuki.okeya.ue,camille.vuillaume.ch}@hitachi.com

Abstract. We propose GMSS, a new variant of the Merkle signature scheme. GMSS is the first Merkle-type signature scheme that allows a *cryptographically unlimited* (2^{80}) number of documents to be signed with one key pair. Compared to recent improvements of the Merkle signature scheme, GMSS reduces the signature size as well as the signature generation cost.

Keywords: Merkle signatures, post-quantum cryptography, SSL.

1 Introduction

Digital signatures are one of the most important applications of public key cryptography. For example, they are an essential part of the SSL/TLS protocol for authenticating websites. If large scale quantum computers are built, all popular digital signature schemes like RSA, DSA and ECDSA will become insecure [12]. In addition, significant progresses have been made for solving the underlying number theoretic problems for RSA or DSA, and therefore, the key lengths required to provide sufficient security have been steadily increasing [10]. As a consequence, it is urgent to look for alternative signature schemes, thoroughly analyze and understand their security, and see how they behave in real-life applications.

A promising alternative to common digital signature schemes is the Merkle signature scheme (MSS) proposed by Merkle in [11]. The security of MSS solely relies on the existence of cryptographic hash functions. According to [7], the required properties of the hash function are one-wayness and collision resistance. Using MSS, it is possible to remove the requirement for number theoretic assumptions from digital signatures. In recent years, many improvements to the original MSS were proposed. The inefficient key generation and the resulting limited signature capacity of MSS is addressed in [3]. The authors proposed CMSS, a variant of MSS that increases the signature capacity from 2^{20} to 2^{40} and provides competitive timings compared to common signature schemes. Other important contributions are efficient tree traversal algorithms [8,13,14], which play a crucial role for fast signature generation.

First of all, it is unclear whether a signature capacity of 2^{40} is sufficient for practical applications. Consider for example webservers using SSL/TLS or electronic archives. Second, the original MSS as well as CMSS suffer from quite large signature sizes. Further, there may be time and space requirements in specific application environments, such as smart cards, that are not satisfied by current constructions.

In this paper we present the generalized Merkle signature scheme (GMSS). GMSS is a highly flexible variant of CMSS that can be adjusted to the requirements and constraints of a particular environment. GMSS drastically reduces the signing time by distributing the costs for one signature generation across several previous signatures and the key generation. This in turn makes it possible to choose parameters that provide smaller signatures. Using GMSS, it is possible to achieve a signature capacity of 2^{80} with competitive timings and reasonable signature sizes, i.e. 26.1 ms for signing, 18.1 ms for verifying, and 3,620 bytes for the signature. In case of a signature capacity of 2^{40} it is possible to achieve 26 ms for signing, 19.6 ms for verifying, and only 1,860 bytes for the signature. For both signature capacities, it is also possible to achieve signing and verification times of only 10 ms at the expense of larger signatures, i.e. 2,340 bytes for 2^{40} and 4,240 bytes for 2^{80}. We also propose a client-server protocol for webservers using SSL/TLS that minimizes the latency and improves resistance to denial of service attacks.

This paper is organized as follows: Section 2 introduces GMSS. Section 3 shows how to choose appropriate parameters and how to exchange signatures in the SSL/TLS protocol. Finally, Section 4 states our conclusion.

2 GMSS in Theory

This section at first shows the general construction of GMSS. Then the key generation, signature generation and verification are explained in detail and the costs and memory requirements are estimated.

2.1 General Construction

The basic construction of GMSS consists of a tree with T layers. The nodes of this tree are in turn Merkle trees [11]. A brief description of Merkle trees can be found in Appendix A. The height of all Merkle trees in a certain layer i is denoted by h_i. Trees in different layers may have different heights. Each Merkle tree in layer $i = 1, \ldots, T-1$ is parent to 2^{h_i} Merkle trees. $\mathcal{T}_{1,0}$ denotes the single Merkle tree in layer 1. The $2^{h_1+\ldots+h_{i-1}}$ Merkle trees in layers $i = 2, \ldots, T$ are denoted by $\mathcal{T}_{i,j}$, $j = 0, \ldots, 2^{h_1+\ldots+h_{i-1}} - 1$ according to their position from left to right.

Parents and children Merkle trees have the following relationship: the root of a child tree is signed with the one-time signature key corresponding to a certain leaf of his parent tree. In the following, when talking about leaves in the context of signing, we mean the corresponding one-time signature key. $\textsc{Root}_{\mathcal{T}}$ denotes the root of tree $\mathcal{T}$. $\textsc{Sig}_{\mathcal{T}}$ denotes the one-time signature of $\textsc{Root}_{\mathcal{T}}$, which is generated using leaf l of $\mathcal{T}$'s parent. Message digests d are signed using the leaves of the Merkle trees on the deepest layer T and we call their one-time signature $\textsc{Sig}_d$. Thanks to the layer hierarchy, the number of message digests that can be signed using one GMSS key pair is $S = 2^{h_1+\ldots+h_T}$. The general construction of GMSS is depicted in Figure 1.

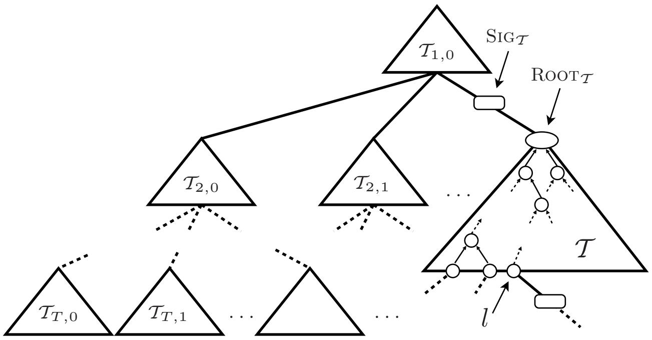


Fig. 1. General construction of GMSS

For any given signature $s \in \{0,\ldots,2^{h_1+\ldots+h_T}-1\}$, there is a unique path p from the Merkle tree on the lowest layer T, which is used to sign the digest, to the single Merkle tree on the top layer 1 ($\mathcal{T}_{1,0}$). This path involves one Merkle tree at each layer $i = 1,\ldots,T$. A GMSS signature of a message digest d contains the one-time signature $\textsc{Sig}_d$ of d and the one-time signatures $\textsc{Sig}_{\mathcal{T}}$ of the roots of all Merkle trees on path p, except for $\mathcal{T}_{1,0}$. For all trees $\mathcal{T}$ on path p, a GMSS signature also contains the authentication path $\textsc{Auth}_{\mathcal{T},l}$ of the leaf l that is used to sign either the child of $\mathcal{T}$, or the digest d. An authentication path is defined as the sequence of the siblings of all nodes on the path from leaf l to the root of $\mathcal{T}$. GMSS uses the Winternitz one-time signature scheme [4,11] for signing digests d and $\textsc{Root}_{\mathcal{T}}$ (see Appendix B for for a brief introduction to the Winternitz one-time signature scheme). w_i denotes the Winternitz parameter used in layer $i = 1,\ldots,T$. Different layers are allowed to use different Winternitz parameters. GMSS is specified by a GMSS parameter set

$$\mathcal{P} = \big(T,(h_1,\ldots,h_T),(w_1,\ldots,w_T)\big).$$

Remark 1. The Merkle variant CMSS proposed in [3] is a special case of GMSS. CMSS uses only two layers, where both layers use the same Winternitz parameter and all trees in both layers have the same height, i.e. $\mathcal{P} = \big(2,(h,h),(w,w)\big)$.

During the key generation, GMSS computes $\textsc{Sig}_{\mathcal{T}}$ and $\textsc{Auth}_{\mathcal{T},l}$ for the Merkle trees on the path p used by the first signature (Section 2.3). $\textsc{Sig}_{\mathcal{T}}$ and $\textsc{Auth}_{\mathcal{T},l}$

required by succeeding signatures are precomputed (Section 2.4). Since those values change less frequently for upper layers, the precomputation can be distributed over many steps. On the one hand, this results in a significant improvement of the signing speed. On the other hand, this enables us to choose large Winternitz parameters w_i, which results in smaller signatures. In Section 3.1, we formulate this trade-off as an optimization problem to find an optimal parameter set.

2.2 Winternitz One-Time Signature Key Generation

First of all, we describe our strategy for generating random data required by one-time signature keys. Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a cryptographic hash function with output length n bits. We adopt the approach for the generation of the Winternitz OTS signature keys proposed in [3] and use a pseudo random number generator (PRNG) $f : \{0,1\}^n \rightarrow \{0,1\}^n \times \{0,1\}^n$, $\textsc{Seed}_{\text{in}} \mapsto (\textsc{Seed}_{\text{out}}, \textsc{Rand})$ for generating secrets. In the following, we call c_{HASH} the cost evaluating one hash (in our case, the input size is small and fixed) and c_{PRNG} the cost for calling the PRNG. To assure interoperability we selected the PRNG described in [6], Appendix 3.1, which requires one single call to the hash function H.

$$\textsc{Rand} \leftarrow H(\textsc{Seed}_{\text{in}}), \textsc{Seed}_{\text{out}} \leftarrow (1 + \textsc{Seed}_{\text{in}} + \textsc{Rand}) \bmod 2^n \tag{1}$$

We use an initial seed $\textsc{Seed}_{\mathcal{T}_{i,j},l}$ to generate the seed for the lth Winternitz OTS signature key of Merkle tree $\mathcal{T}_{i,j}$, i.e.

$$(\textsc{Seed}_{\mathcal{T}_{i,j},l+1}, \textsc{Seed}_{\text{OTS}}) \leftarrow f(\textsc{Seed}_{\mathcal{T}_{i,j},l})$$
$$(\textsc{Seed}_{\text{OTS}}, x_k) \leftarrow f(\textsc{Seed}_{\text{OTS}}), k = 1, \ldots, t_{w_i}$$

and $t_{w_i} = \lceil n/w_i \rceil + \lceil (\lfloor \log_2(\lceil n/w_i \rceil) \rfloor + 1 + w_i) / w_i \rceil$. The lth one-time signature key and the lth leaf of Merkle tree $\mathcal{T}_{i,j}$ are given as $X = (x_1, \ldots, x_{t_{w_i}})$ and $Y = H\big(H^{2^{w_i}-1}(x_1) \| \ldots \| H^{2^{w_i}-1}(x_{t_w})\big)$, respectively. Note that Y is also the Winternitz one-time verification key that corresponds to X. $H^k(x)$ denotes the hash function applied k times and $\|$ the concatenation of two strings. The updated seed $\textsc{Seed}_{\mathcal{T}_{i,j},l+1}$ is stored and used to generate the $(l+1)$th signature key. If the current signature key is associated with the last leaf of tree $\mathcal{T}_{i,j}$, i.e. $l = 2^{h_i} - 1$, the updated seed is used as initial seed for the next Merkle tree $\mathcal{T}_{i,j+1}$, i.e. $(\textsc{Seed}_{\mathcal{T}_{i,j+1},0}, \textsc{Seed}_{\text{OTS}}) \leftarrow f(\textsc{Seed}_{\mathcal{T}_{i,j},2^{h_i}-1})$.

2.3 GMSS Key Generation

Next, we explain how to generate a GMSS keypair, establish the size of the public and private keys and the cost for computing them. From the parameter set $\mathcal{P}$ and initial seeds $\textsc{Seed}_{\mathcal{T}_{1,0},0}, \ldots, \textsc{Seed}_{\mathcal{T}_{T,0},0}$, the key generation step computes the GMSS public key $\textsc{Root}_{\mathcal{T}_{1,0}}$ and the GMSS private key which consists of the following entries.

$$\left\{ \begin{array}{rr} \textsc{Seed}_{\mathcal{T}_{i,0},0}\ , i = 1, \ldots, T\ , & \textsc{Seed}_{\mathcal{T}_{i,2},0}\ , i = 2, \ldots, T \\ \textsc{Sig}_{\mathcal{T}_{i,0}}\ , i = 2, \ldots, T\ , & \textsc{Root}_{\mathcal{T}_{i,1}}\ , i = 2, \ldots, T \\ \textsc{Auth}_{\mathcal{T}_{i,0},0}\ , i = 1, \ldots, T\ , & \textsc{Auth}_{\mathcal{T}_{i,1},0}\ , i = 2, \ldots, T \end{array} \right. \tag{2}$$

At first, the key generation computes the root of the first tree in each layer $\text{Root}_{\mathcal{T}_{i,0}}, i = 1, \ldots, T$. This can be done efficiently using a classical algorithm also referred to as treehash [13], shown in Algorithm 1. This algorithm uses a stack $\mathcal{S}$ of nodes, where each node knows its height in the tree. In this paper, we use a slightly modified version of treehash, which allows us to easily distribute costs by incrementally computing the root. In Algorithm 1, the leaf l is the lth verification key, computed using $\text{Seed}_{\mathcal{T}_{i,j},l}$ as described above. For a tree of height h_i, Algorithm 1 must be called 2^{h_i} times, where the 2^{h_i} leaves are supplied in sequential order, from left to right. For each call of Algorithm 1, the inner while loop might compute from 0 to h_i iterations, but in total, the 2^{h_i} calls will result in exactly $2^{h_i} - 1$ iterations. After the 2^{h_i} calls, the stack $\mathcal{S}$ contains a single node, the root of the tree. Note that during the computation of the root $\text{Root}_{\mathcal{T}_{i,0}}$, the authentication path for the 0th leaf of tree $\mathcal{T}_{i,0}$, i.e. $\text{Auth}_{\mathcal{T}_{i,0},0}$ is generated for free, since all nodes of the tree are parsed by the algorithm.

Algorithm 1. Treehash

Input: Leaf l, stack $\mathcal{S}$

Output: updated stack $\mathcal{S}$

```
1. push $l$ to $\mathcal{S}$
2. **while** top two nodes of $\mathcal{S}$ have the same height **do**
   2.1. pop $n_1$ from $\mathcal{S}$; pop $n_2$ from $\mathcal{S}$
   2.2. push $H(n_2||n_1)$ to $\mathcal{S}$
3. **return** $\mathcal{S}$
```

Next, the roots $\text{Root}_{\mathcal{T}_{i,1}}$ and authentication paths $\text{Auth}_{\mathcal{T}_{i,1},0}$ of of the succeeding trees $\mathcal{T}_{i,1}$, $i = 2, \ldots, T$ are computed with Algorithm 1. As explained above, the initial seeds $\text{Seed}_{\mathcal{T}_{i,1},0}$ related to the trees $\mathcal{T}_{i,1}$ are now available. Finally, after generating the second tree in each layer, the seeds $\text{Seed}_{\mathcal{T}_{i,2},0}$ are available, which are stored as part of the private key to allow an efficient generation of trees $\mathcal{T}_{i,2}$ during the signing process. Note that $\text{Sig}_{\mathcal{T}_{i,0}}$, $i = 2, \ldots, T$ does not have to be computed explicitly. It is an intermediate value during the computation of the 0th leaf of tree $\mathcal{T}_{i-1,0}$, $i = 2, \ldots, T$.

Lemma 1 (Key Generation). *The total cost for the key generation is*

$$c_{\text{keygen}} = \sum_{i=1}^{T} c_{\text{tree}}(i) + \sum_{i=2}^{T} c_{\text{tree}}(i) \tag{3}$$

where $c_{\text{tree}}(i) = \left(2^{h_i}\left(t_{w_i}(2^{w_i}-1)+1\right) + 2^{h_i}-1\right) c_{\text{Hash}} + 2^{h_i}\left(t_{w_i}+1\right) c_{\text{Prng}}$. *The memory requirements for the keys are*

$$\begin{aligned} m_{\text{pubkey}} &= n \text{ bits} \\ m_{\text{privkey}} &= \left(\sum_{i=1}^{T}(h_i+1) + \sum_{i=2}^{T}(h_i + t_{w_{i-1}} + 2)\right) n \text{ bits.} \end{aligned} \tag{4}$$

Proof. A tree on layer i requires the computation of 2^{h_i} leaves. Each leaf computation requires $(2^{w_i}-1)\cdot t_{w_i}+1$ hash function evaluations and $t_{w_i}+1$ calls to the PRNG: one PRNG to obtain the seed and t_{w_i} to obtain the signature key. The 2^{h_i} applications of treehash require $2^{h_i}-1c_{\text{HASH}}$. Therefore, the total cost for one tree on layer i is given as $c_{\text{tree}}(i) = \left(2^{h_i}\left((2^{w_i}-1)\cdot t_{w_i}+1\right)+2^{h_i}-1\right)c_{\text{HASH}} + 2^{h_i}\left(t_{w_i}+1\right)c_{\text{PRNG}}$. Since we construct two trees on layers $i=2,\ldots T$ and one on layer $i=1$, the total cost for the key generation is given by Equation (3). The memory requirements of the keys depend on the output size of the hash function n. A root $\text{Root}_{\mathcal{T}_{i,j}}$ is a single hash value and requires n bits, which explains $m_{\text{pubkey}} = n$ bits. A seed $\text{Seed}_{\mathcal{T}_{i,j},l}$ requires n bits. A one-time signature $\text{Sig}_{\mathcal{T}_{i,j}}$ requires $n\cdot t_{w_{i-1}}$ bits. An authentication path requires $h_i\cdot n$ bits. For each layer $i=2,\ldots,T$, we store two seeds, two authentication paths, one root and one one-time signature. For layer $i=1$, we store one seed and one authentication path. Hence, the size of the pivate key is m_{privkey} as in Equation (4). □

2.4 Signature Generation

In the following, we discuss the signature generation stage. We introduce the components of a GMSS signature and estimate formulas for the signature size and costs. We also explain how the signature generation costs are distributed. For the sth GMSS signature ($s \in \{0,\ldots,2^{h_1+\ldots+h_T}-1\}$), let

$$\begin{cases} j_T = \lfloor s/2^{h_T}\rfloor, & l_T = s \bmod 2^{h_T} \\ j_i = \lfloor j_{i+1}/2^{h_i}\rfloor, & l_i = j_{i+1} \bmod 2^{h_i}, i=1,\ldots,T-1. \end{cases} \tag{5}$$

The path from the lowest tree $\mathcal{T}_{T,j_T}$ to the top tree $\mathcal{T}_{1,0}$ used by the sth signature is given as $(\mathcal{T}_{T,j_T},\mathcal{T}_{T-1,j_{T-1}},\ldots,\mathcal{T}_{2,j_2},\mathcal{T}_{1,0})$. The one-time signature Sig_d of the message digest d is generated using the l_Tth leaf of tree $\mathcal{T}_{T,j_T}$. The one-time signature $\text{Sig}_{\mathcal{T}_{i,j_i}}$ of the root of tree $\mathcal{T}_{i,j_i}$ is generated using the l_{i-1}th leaf of tree $\mathcal{T}_{i-1,j_{i-1}}$, $i=2,\ldots,T$. The sth GMSS signature consists of

1. The index s
2. The one-time signature Sig_d
3. The one-time signatures $\text{Sig}_{\mathcal{T}_{i,j_i}}$, $i=2,\ldots,T$
4. The authentication paths $\text{Auth}_{\mathcal{T}_{i,j_i},l_i}$, $i=1,\ldots,T$

Lemma 2 (Signature Size). *The size of a signature is*

$$m_{\text{signature}} = \sum_{i=1}^{T}(h_i + t_{w_i})\cdot n \text{ bits}. \tag{6}$$

Proof. A signature consists of T authentication paths ($h_i\cdot n$ bits) and T one-time signatures ($t_{w_i}\cdot n$ bits), one for each layer $i=1,\ldots,T$. Adding up yields $m_{\text{signature}}$ as shown by Equation (6) as the size of a signature. □

Following the framework of [5], we split the signature generation into two parts. The first part is the online part which computes Sig_d and outputs the signature.

The second part is the offline part that precomputes the authentication paths and one-time signatures of the roots required for upcoming signatures. In fact, the offline part performs an update of the private key and GMSS is therefore a key-evolving signature scheme [2]. Note that for the first signature $s = 0$ the offline part was done during the key generation.

Lemma 3 (Online Signature Cost). *The average cost for the online signing part is*

$$c_{\text{online}} = (2^{w_T} - 1)t_{w_T}/2 \cdot c_{\text{HASH}} + (t_{w_T} + 1)c_{\text{PRNG}}. \tag{7}$$

Proof. The generation of the one-time signature key requires one call to the PRNG to obtain the seed and t_{w_T} are necessary to obtain the secrets. The average signing costs of the Winternitz one-time signature scheme are $\big((2^{w_T} - 1)t_{w_T}\big)/2 \cdot c_{\text{HASH}}$. This leads to Equation (7). □

Next, we explain how to efficiently compute the offline signature part by distributing its cost. Our idea is based on the observation that trees in upper layers do not change frequently from one signature to the other. In the following, the values l_i, j_i correspond to the current signature s.

We begin with the precomputation of $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$, $i = 2, \dots, T$, which must be ready when the next signature uses tree $\mathcal{T}_{i,j_i+1}$. $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ is generated using the one-time signature key that corresponds to either the $(l_{i-1} + 1)$th leaf of tree $\mathcal{T}_{i-1,j_{i-1}}$ or the 0th leaf of tree $\mathcal{T}_{i-1,j_{i-1}+1}$. The latter case appears if $(l_{i-1}+1) = 0 \bmod 2^{h_{i-1}}$, i.e. we have to use the next tree in the next upper layer $i-1$. For now we assume that $\textsc{Root}_{\mathcal{T}_{i,j_i+1}}$ is known when tree $\mathcal{T}_{i,j_i}$ is used for the first time, i.e. $l_i = 0$. This certainly holds if $j_i = 0$, since $\textsc{Root}_{\mathcal{T}_{i,1}}$ was computed during the key generation. We distribute the computation of $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ over the 2^{h_i} leaves (or steps) of tree $\mathcal{T}_{i,j_i}$. If $l_i = 0$ we use $\textsc{Root}_{\mathcal{T}_{i,j_i+1}}$ and perform the initialization steps of the Winternitz one-time signature scheme. Then we compute $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ step-by-step each time we advance one leaf in tree $\mathcal{T}_{i,j_i}$. The generation of $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ is completed if $l_i = 2^{h_i} - 1$.

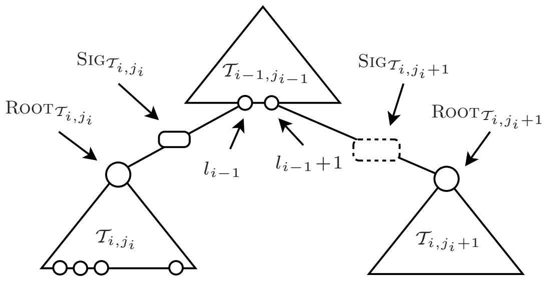
Lemma 4. *On average, we require*

$$c_{\text{sig}}(i) = \left\lceil \frac{(2^{w_{i-1}}-1)t_{w_{i-1}}}{2^{h_i+1}} \right\rceil c_{\text{HASH}} + \left\lceil \frac{t_{w_{i-1}}+1}{2^{h_i}} \right\rceil c_{\text{PRNG}} \tag{8}$$

operations each time we advance one leaf in $\mathcal{T}_{i,j_i}$ to compute $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$.

Proof. The one-time signature $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ is generated using the Winternitz parameter of layer $i-1$ (w_{i-1}), and on average requires $(2^{w_{i-1}} - 1)t_{w_{i-1}}/2$ hash evaluations and $t_{w_{i-1}} + 1$ calls to the PRNG, see Lemma 3. Since there are 2^{h_i} leaves in the tree on layer i, the computation of $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ can be distributed over 2^{h_i} steps which yields Equation (8) as costs per step. □

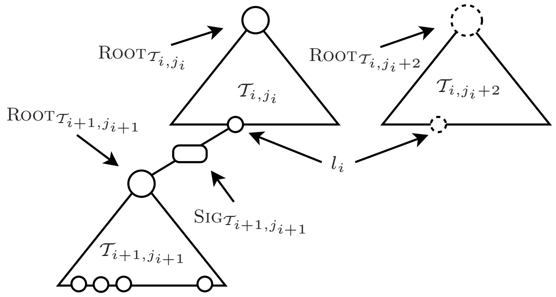
Above, we assumed that $\textsc{Root}_{\mathcal{T}_{i,j_i+1}}$ is known when we first use tree $\mathcal{T}_{i,j_i}$. Hence we must precompute $\textsc{Root}_{\mathcal{T}_{i,j_i+2}}$ while using tree $\mathcal{T}_{i,j_i}$, such that it is ready when we switch to tree $\mathcal{T}_{i,j_i+1}$ and want to start generating $\textsc{Sig}_{\mathcal{T}_{i,j_i+2}}$. This is



$\mathrm{SIG}_{\mathcal{T}_{i,j_i}}$
$\mathcal{T}_{i-1,j_{i-1}}$
$\mathrm{SIG}_{\mathcal{T}_{i,j_i+1}}$
$\mathrm{ROOT}_{\mathcal{T}_{i,j_i}}$
$\mathrm{ROOT}_{\mathcal{T}_{i,j_i+1}}$
l_{i-1}
$l_{i-1}+1$
$\mathcal{T}_{i,j_i}$
$\mathcal{T}_{i,j_i+1}$

Fig. 2. $\mathrm{SIG}_{\mathcal{T}_{i,j_i+1}}$ is precomputed from $\mathrm{ROOT}_{\mathcal{T}_{i,j_i+1}}$ while using tree $\mathcal{T}_{i,j_i}$

done by successively computing the leaves of tree $\mathcal{T}_{i,j_i+2}$ and passing them to Algorithm 1. While using the l_ith leaf of $\mathcal{T}_{i,j_i}$ we compute the l_ith leaf of $\mathcal{T}_{i,j_i+2}$. Its computation is distributed over the $2^{h_{i+1}}$ leaves (or steps) of $\mathcal{T}_{i+1,j_{i+1}}$, the current tree on the next lower layer $i+1$. Once the leaf is generated, it is passed to treehash which partially constructs $\mathrm{ROOT}_{\mathcal{T}_{i,j_i+2}}$. Since treehash must be called 2^{h_i} times to construct the root of a tree of height h_i, the construction of $\mathrm{ROOT}_{\mathcal{T}_{i,j_i+2}}$ is completed once we switch from tree $\mathcal{T}_{i,j_i}$ to tree $\mathcal{T}_{i,j_i+1}$. Note that $\mathrm{SEED}_{\mathcal{T}_{i,j_i+2},0}$, the seed required to compute the 0th leaf $\mathcal{T}_{i,j_i+2}$ was obtained during the generation of $\mathrm{ROOT}_{\mathcal{T}_{i,j_i+1}}$ and is part of the private key. If $j_i = 0$ the seed was computed during the key generation. For the lowest layer $i = T$ the computation of the leaves has to be done at once. We also store $\mathrm{AUTH}_{\mathcal{T}_{i,j_i+2},0}$ during the preparation of $\mathrm{ROOT}_{\mathcal{T}_{i,j_i+2}}$.



$\mathrm{ROOT}_{\mathcal{T}_{i,j_i}}$
$\mathrm{ROOT}_{\mathcal{T}_{i,j_i+2}}$
$\mathcal{T}_{i,j_i}$
$\mathcal{T}_{i,j_i+2}$
$\mathrm{ROOT}_{\mathcal{T}_{i+1,j_{i+1}}}$
l_i
$\mathrm{SIG}_{\mathcal{T}_{i+1,j_{i+1}}}$
$\mathcal{T}_{i+1,j_{i+1}}$

Fig. 3. Leaf l_i of tree $\mathcal{T}_{i,j_i+2}$ is precomputed while using tree $\mathcal{T}_{i+1,j_{i+1}}$

Lemma 5. *We require*

$$\begin{cases} c^1_{\mathrm{leaf}}(i) = \left\lceil \frac{(2^{w_i}-1)t_{w_i}+1}{2^{h_{i+1}}} \right\rceil c_{\mathrm{HASH}} + \left\lceil \frac{t_{w_i}+1}{2^{h_{i+1}}} \right\rceil c_{\mathrm{PRNG}} \\ c^2_{\mathrm{leaf}}(i) = h_i \cdot c_{\mathrm{HASH}} \ \textit{(at most)} \end{cases} \tag{9}$$

operations each time we advance one leaf in $\mathcal{T}_{i+1,j_{i+1}}$ *and* $\mathcal{T}_{i,j_i}$*, respectively, to compute* $\mathrm{ROOT}_{\mathcal{T}_{i,j_i+2}}$.

Proof. The generation of the l_ith leaf of tree $\mathcal{T}_{i,j_i+2}$ requires $((2^{w_i}-1)t_{w_i}+1)c_{\text{HASH}}$ and $(t_{w_i}+1)c_{\text{PRNG}}$. Since there are $2^{h_{i+1}}$ leaves in the tree on layer $i+1$, the computation of the l_ith leaf can be distributed over $2^{h_{i+1}}$ steps which yields $c^1_{\text{leaf}}(i)$. The while-loop of treehash requires at most h_i hash function evaluations which yields $c^2_{\text{leaf}}(i)$. □

Next, we describe the precomputation of $\text{AUTH}_{\mathcal{T}_{i,j_i},l_i+1}$, the authentication path of the next leaf of tree $\mathcal{T}_{i,j_i}$. We use an algorithm proposed by Szydlo in [13]. This algorithm uses h_i-1 instances of treehash to compute the upcoming authentication nodes. Given a leaf index l_i, Szydlo's algorithm firstly checks if a new instance of treehash must be generated. Then it spends at most h_i computational units, which are either hash function evaluations for the while-loop of treehash or leaf calculations. Again, the computation of the required leaves is distributed over the $2^{h_{i+1}}$ leaves of $\mathcal{T}_{i+1,j_{i+1}}$. When using the leaf $l_{i+1}=0$ of tree $\mathcal{T}_{i+1,j_{i+1}}$, we perform the initialization steps of Szyldo's algorithm and decide (1) if a new instance of treehash must be generated and (2) how many new leaves are required by the active treehash instances. Those leaves are computed step-by-step as explained above. If $l_{i+1}=2^{h_{i+1}}-1$ the calculation of the required leaves is completed and we pass them to Szydlo's algorithm which updates the treehash instances and outputs $\text{AUTH}_{\mathcal{T}_{i,j_i},l_i+1}$. Note that $\text{AUTH}_{\mathcal{T}_{i,j_i},0}$ is stored during the construction of $\text{ROOT}_{\mathcal{T}_{i,j_i}}$ and therefore already available if $l_i=0$. Also, $\text{AUTH}_{\mathcal{T}_{T,j_T},l_T+1}$ must be computed at once.

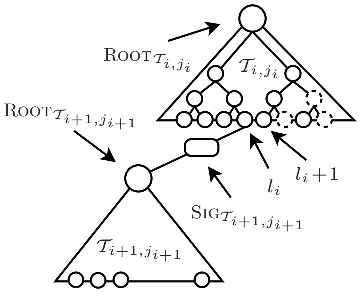

$\text{ROOT}_{\mathcal{T}_{i,j_i}}$
$\mathcal{T}_{i,j_i}$
$\text{ROOT}_{\mathcal{T}_{i+1,j_{i+1}}}$
l_i
l_i+1
$\text{SIG}_{\mathcal{T}_{i+1,j_{i+1}}}$
$\mathcal{T}_{i+1,j_{i+1}}$

Fig. 4. Leaves required for $\text{AUTH}_{\mathcal{T}_{i,j_i},l_i+1}$ are precomputed while using tree $\mathcal{T}_{i+1,j_{i+1}}$

Lemma 6. *We require at most*

$$\begin{cases} c^1_{\text{auth}}(i) = h_i \cdot c^1_{\text{leaf}}(i) + \left\lceil \frac{2^{h_i-2}}{2^{h_{i+1}}} \right\rceil c_{\text{PRNG}} \\ c^2_{\text{auth}}(i) = h_i \cdot c_{\text{HASH}} \end{cases} \tag{10}$$

operations each time we advance one leaf in $\mathcal{T}_{i+1,j_{i+1}}$ and $\mathcal{T}_{i,j_i}$, respectively, to compute $\text{AUTH}_{\mathcal{T}_{i,j_i},l_i+1}$.

Proof. Szydlo's algorithm initializes at most one instance of treehash in each iteration. We need at most 2^{h_i-2} calls to the PRNG to obtain the initial seed for the leaf required by this instance from the current seed. In the worst case, the active treehash instances require the computation of h_i leaves. The computation of those h_i leaves and the 2^{h_i-2} calls to the PRNG are distributed over the $2^{h_{i+1}}$ steps in the tree on layer $i+1$. This yields $c^1_{\text{auth}}(i)$. At most $h_i - 1$ hash evaluations are spend on the while-loops of the active treehash instances. One hash evaluation is required by the initialization steps of Szydlo's algorithm. This yields $c^2_{\text{auth}}(i)$. □

The following lemma considers the worst case costs of the offline part. It assumes that for the next signature, we have to advance one leaf in each tree $\mathcal{T}_{i,j_i}$, $i = 1,\dots,T-1$. Note that this is equivalent to advancing from tree $\mathcal{T}_{i,j_i}$ to $\mathcal{T}_{i,j_i+1}$ in all layers $i = 2,\dots,T$.

Lemma 7 (Offline Signature Cost and Memory). *The worst case costs and the memory for the offline part are*

$$\begin{aligned} c_{\text{offline}} &= \sum_{i=2}^{T} \left(c_{\text{sig}}(i) + c^1_{\text{leaf}}(i) + c^2_{\text{leaf}}(i)\right) + \sum_{i=1}^{T} \left(c^1_{\text{auth}}(i) + c^2_{\text{auth}}(i)\right) \\ m_{\text{offline}} &= \left(\sum_{i=2}^{T} (t_{w_{i-1}} + 4h_i) + 3h_1\right) \cdot n \text{ bits.} \end{aligned} \tag{11}$$

Proof. In the worst case, we have to advance one leaf in the current tree on all layers $i = 1,\dots,T-1$. The cost for this case are obtained by adding the costs for the precomputation of $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ and $\textsc{Root}_{\mathcal{T}_{i,j_i+2}}$ for all layers $i = 2,\dots,T$ and $\textsc{Auth}_{\mathcal{T}_{i,j_i},l_i+1}$ for all layers $i = 1,\dots,T$. This yields c_{offline}. During the offline part, we have to store $\textsc{Sig}_{\mathcal{T}_{i,j_i+1}}$ which requires $t_{w_{i-1}} \cdot n$ bits and the stack required by treehash to construct $\textsc{Root}_{\mathcal{T}_{i,j_i+2}}$ which requires $h_i \cdot n$ bits, $i = 2,\dots,T$. Further, we have to store $\textsc{Auth}_{\mathcal{T}_{i,j_i},l_i+1}$ and some temporary nodes required by Szydlo's algorithm which require $3h_i \cdot n$ bits, $i = 1,\dots,T$. This yields m_{offline}. □

2.5 Verification

The verification process of GMSS is essentially the same as for MSS and CMSS. The verifier knows the public key $\textsc{Root}_{\mathcal{T}_{1,0}}$ and the parameter set $\mathcal{P}$ used by the signer. At first, he verifies the one-time signature $\textsc{Sig}_d$ of the digest d using the Winternitz parameter w_T. Doing so, he obtains the verification key for this signature, i.e. leaf l_T of tree $\mathcal{T}_{T,j_T}$. Then, the verifier repeats the following steps for $i = T,\dots,2$. (1) use l_i and $\textsc{Auth}_{\mathcal{T}_{i,j_i},l_i}$ to compute $\textsc{Root}_{\mathcal{T}_{i,j_i}}$. (2) use $\textsc{Root}_{\mathcal{T}_{i,j_i}}$ and verify $\textsc{Sig}_{\mathcal{T}_{i,j_i}}$ and obtain l_{i-1}. Finally, the verifier uses l_1 and $\textsc{Auth}_{\mathcal{T}_{1,j_1},l_1}$ to obtain $\textsc{Root}_{\mathcal{T}_{1,0}}$. If $\textsc{Root}_{\mathcal{T}_{1,0}}$ matches the signers public key, the signature is accepted.

Lemma 8 (Verification Cost). *The average cost for the verification is*

$$c_{\text{verify}} = \sum_{i=1}^{T} \left((2^{w_i} - 1)t_{w_i}/2 + h_i\right) c_{\textsc{Hash}}. \tag{12}$$

Proof. On average, $\big((2^{w_i}-1)t_{w_i}\big)/2$ hash evaluations are required to verify an one-time signature. Using a leaf and an authentication path to construct a root requires h_i hash evaluations. Since there is a one-time signature and an authentication path for each layer the average costs for the verification are c_{verify}. □

3 GMSS in Practice

In this section, we give practical GMSS parameters that simultaneously allow for a large signature capacity, good efficiency and small bandwidth, and describe how to integrate GMSS in a protocol such as SSL with a client/server architecture.

3.1 Choosing $\mathcal{P}$

To find an optimal parameter set, we consider following optimization problem: given certain bounds on the signature capacity as well as the key generation, signature generation and verification time, find the parameter set which provides the smallest signatures. We formulated this optimization problem as mixed integer program (MIP) using Zimpl [9]. The constraints of this MIP are the equations for the key generation (3), signature generation (7),(11) and verification (12) time and the signature size (6). Note that the worst cast cost of Szydlo's algorithm for the authentication path computation shown in Equation (10) occurs only once per tree. To compute the parameter sets, we use the average costs of Szydlo's algorithm, which are

$$\Big(h_i/2\cdot\big((2^{w_i}-1)t_{w_i}+1\big)+h_i/2-1\Big)c_{\text{HASH}}+\Big(h_i+t_{w_i}\Big)c_{\text{PRNG}}$$

for a tree on layer i. To solve the MIP, we used the free solver SCIP [1]. Using different bounds for the signature generation and verification time, we obtained the following parameter sets $\mathcal{P}_k=\big(T,(h_1,\dots,h_T),(w_1,\dots,w_T)\big)$ that allow up to 2^k signatures.

$$\begin{array}{ll}\mathcal{P}_{40}=\big(2,(20,20),(10,5)\big) & \mathcal{P}_{80}=\big(4,(20,20,20,20),(8,8,8,5)\big)\\ \mathcal{P}'_{40}=\big(2,(20,20),(9,3)\big) & \mathcal{P}'_{80}=\big(4,(20,20,20,20),(7,7,7,3)\big)\end{array}$$

Table 1 shows timings (t) and memory requirements (m) for the parameter sets when using a 160 bit hash function. The size of the public key is $m_{\text{pubkey}}=20$

Table 1. Timings and memory requirements

	m_{offline}	m_{privkey}	$m_{\text{signature}}$	t_{keygen}	t_{sign}	t_{verify}
$\mathcal{P}_{40}$	3160 bytes	1640 bytes	1860 bytes	723 min	26.0 ms	19.6 ms
$\mathcal{P}'_{40}$	3200 bytes	1680 bytes	2340 bytes	390 min	10.7 ms	10.7 ms
$\mathcal{P}_{80}$	7320 bytes	4320 bytes	3620 bytes	1063 min	26.1 ms	18.1 ms
$\mathcal{P}'_{80}$	7500 bytes	4500 bytes	4240 bytes	592 min	10.1 ms	10.1 ms

bytes for all parameter sets. To get timings, we use the ratio $c_{\text{HASH}} = c_{\text{PRNG}} = 0.002$ ms, which we obtained using a Java implementation of SHA1 on a Pentium dualcore 1.8GHz.

This table clarifies the flexibility of GMSS. $\mathcal{P}_{40}$ and $\mathcal{P}_{80}$ provide the shortest possible signatures. In case of $\mathcal{P}_{40}$, the signature size is reduced more that 26% compared to what was stated in [3]. $\mathcal{P}'_{40}$ and $\mathcal{P}'_{80}$ provide very fast signature generation and verification times, at the expense of larger signatures.

Note that a large portion of the signature cost is required for the precomputation of the upcoming authentication paths. One possibility to circumvent this is to use a tree of small height (≤ 5) in the lowest layer and to store it completely in memory. Then the authentication paths can be obtained for free. In case of a 160 bit hash function, storing a tree of height five requires 1,260 bytes.

3.2 Message Flow and Application to SSL/TLS

Finally, we describe how signatures should be transmitted when the signer and the verifier are connected during the signing process as in case of the SSL/TLS protocol. Thanks to the online/offline strategy [5], the server has all signature parts ready from the beginning of the transaction, except for the one-time signature of the message digest SIG_d. The server delays the generation of the online signature SIG_d and sends only the first offline signature part $\text{SIG}_{\mathcal{T}_{T,j_T}}$ and $\text{ROOT}_{\mathcal{T}_{T,j_T}}$ to the client. Then, the client demonstrates his honesty by sending leaf l_{T-1} (the verification key of $\text{SIG}_{\mathcal{T}_{T,j_T}}$) back to the server. The client has to spend some computational effort to obtain l_{T-1}, namely he has to verify $\text{SIG}_{\mathcal{T}_{T,j_T}}$, while the server does not have to do anything. While the server is waiting for the client to send l_{T-1}, he can also start with next the offline part of the signing process. When the server receives the correct leaf l_{T-1}, he sends the remaining parts of the signature and starts to compute SIG_d. In the same time, the client can verify the remaining one-time signatures $\text{SIG}_{\mathcal{T}_{i,j_i}}$, $i = 2, \ldots, T-1$

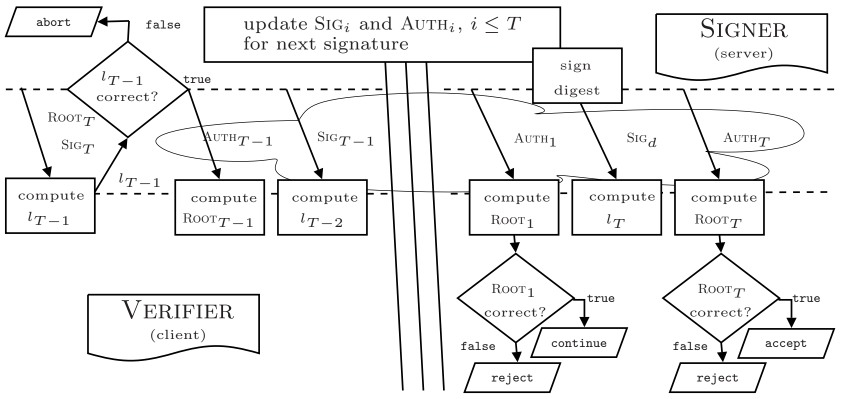


Fig. 5. Message Flow for SSL/TLS

that he receives from the server, and compares the root of the top tree to the server's public key. In the final step, the server sends $\textsc{Sig}_d$ to the client who verifies its correctness.

The overhead of our protocol for the server is just $2n$ bits memory to store $\textsc{Root}_{\mathcal{T}_{T,j_T}}$ and l_{T-1}. The benefits are as follows: it minimizes bandwidth and server-side calculations in case of DoS attacks, and optimizes the latency of the transaction. Indeed, the protocol can be stopped early in case of DoS. In addition, the signature generation, transmission and verification stages are performed concurrently.

4 Conclusion

We presented the generalized Merkle signature scheme (GMSS). GMSS is parameterized by the parameter set $\mathcal{P}$, that allows a great degree of freedom in choosing the signature capacity, the signature generation and verification timings, and the signature size. GMSS uses a scheduling strategy to precompute upcoming signatures. This drastically reduces the signature generation time and in turn allows to choose parameters that provide smaller signatures. For a signature capacity of 2^{40}, the signature size is 1,860 bytes, where signature generation and verification requires 26 ms and 19.6 ms, respectively. GMSS is the first Merkle-type signature scheme that maintains its efficiency even if the signature capacity *cryptographically unlimited*, i.e. 2^{80}. In that case, the signature size is 3,620 bytes, where signature generation and verification requires 26.1 ms and 18.1 ms, respectively. For both signature capacities $(2^{40}, 2^{80})$, it is also possible to find parameter sets such that the signature generation and verification time is only 10 ms. This makes GMSS a serious competitor to commonly used signature schemes such as RSA or ECDSA. Furthermore, GMSS does not rely on number theoretic problems and is not vulnerable to quantum computer attacks. Finally, we proposed a DoS-resilient protocol for SSL/TTL that minimizes the total latency of a signature exchange.

References

1. Tobias Achterberg. SCIP - a framework to integrate constraint and mixed integer programming. Technical Report 04-19, Zuse Institute Berlin, 2004.
2. Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In *Proc. Advances in Cryptology (Crypto'99)*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–238. Springer-Verlag, 1999.
3. Johannes Buchmann, Luis Carlos Coronado Garcia, Erik Dahmen, Martin Döring, and Elena Klintsevich. CMSS – an improved Merkle signature scheme. In *Proc. Progress in Cryptology (Indocrypt'06)*, volume 4329 of *Lecture Notes in Computer Science*, pages 349–363. Springer-Verlag, 2006.
4. Chris Dods, Nigel Smart, and Martijn Stam. Hash based digital signature schemes. In *Proc. Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 96–115. Springer-Verlag, 2005.

5. Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In *Proc. Advances in Cryptology (Crypto'89)*, volume 435 of *Lecture Notes in Computer Science*, pages 263–277. Springer-Verlag, 1990.
6. Digital signature standard (DSS). FIPS PUB 186-2, 2007. Available at `http://csrc.nist.gov/publications/fips/`.
7. Luis Carlos Coronado García. On the security and the efficiency of the Merkle signature scheme. Cryptology ePrint Archive, Report 2005/192, 2005. `http://eprint.iacr.org/`.
8. Markus Jakobsson, Tom Leighton, Silvio Micali, and Michael Szydlo. Fractal Merkle tree representation and traversal. In *Proc. Cryptographer's Track at RSA Conference (CT-RSA'03)*, volume 2612 of *Lecture Notes in Computer Science*, pages 314–326. Springer-Verlag, 2003.
9. Thorsten Koch. Rapid mathematical programming. Technical Report 04-58, Zuse Institute Berlin, 2004.
10. Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.
11. Ralph C. Merkle. A certified digital signature. In *Proc. Advances in Cryptology (Crypto'89)*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1989.
12. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Comput. Soc. Press, 1994.
13. Michael Szydlo. Merkle tree traversal in log space and time (preprint). Available at `http://www.szydlo.com/`.
14. Michael Szydlo. Merkle tree traversal in log space and time. In *Proc. Advances in Cryptology (Eurocrypt'04)*, volume 3027 of *Lecture Notes in Computer Science*, pages 541–554. Springer-Verlag, 2004.

A Merkle's Tree Authentication Scheme

The tree authentication scheme was proposed by Merkle in [11] for using multiple one-time signature instances with a single "master" public key. Merkle's tree authentication scheme in conjunction with a one-time signature scheme is referred to as the Merkle signature scheme (MSS).

Keypair Generation: The signer first generates 2^h one-time key pairs. The one-time verification keys form the leaves of a binary hash tree of height h, called Merkle tree. The value of an inner node is obtained by hashing the concatenation of the values of its two children. Iterating yields the root of the tree which is the MSS public key, and the private key consists of the 2^h one-time signature keys.

Signature Generation: To authenticate the s-th leaf, i.e. the s-th verification key, the s-th authentication path is required. This authentication path consists of the $h-1$ siblings of the $h-1$ nodes on the path from the sth leaf to the root. The s-th Merkle signature consists of four parts: first the index $s \in \{0, \ldots, 2^h - 1\}$ of the selected one-time signature; second, the one-time signature of data d generated with the s-th signature key; third, the s-th verification key; and fourth, the authentication path for the s-th verification key.

Verification: After verifying the one-time signature of d, the verifier has to validate the authenticity of the supplied verification key. Using the index s and the authentication path of the s-th leaf, he re-computes the path from the sth leaf to the root. To do so, he hashes the concatenation of the s-th leaf and first node in the authentication path to obtain the first node on the path to the root (the order for concatenating is decided according to the index s). By successively concatenating the hashed nodes and authentication nodes, the verifier can eventually recover the root itself. If the root matches the signer's public key, the signature is valid.

B The Winternitz One-Time Signature Scheme

The Winternitz OTS [11], described in detail in [4], is parameterized by w, which allows a trade-off between the signature cost and size.

Keypair Generation: The keypair generation produces t_w random values x_1, x_2, ..., x_{t_w}, where $t_w = \lceil n/w \rceil + \lceil (\lfloor \log_2(\lceil n/w \rceil) \rfloor + 1 + w)/w \rceil$. Then, the one-time signature key is $X = (x_1, \ldots, x_{t_w})$, and the one-time verification key $Y = H\big(H^{2^w-1}(x_1)\| \ldots \|H^{2^w-1}(x_{t_w})\big)$, where $H^k(x)$ denotes the hash function applied k times and $\|$ the concatenation of two strings. The cost for the key pair generation is $c_{\text{OTSkeygen}} = \big((2^w - 1)t_w + 1\big)c_{\text{HASH}} + t_w \cdot c_{\text{PRNG}}$.

Signature Generation: For an n-bit message digest d, the OTS is computed as follows. The binary representation of d (possibly padded) is divided into $\lceil n/w \rceil$ blocks of length w: $b_1, \ldots, b_{\lceil n/w \rceil}$. Next, the checksum $C = \sum_{k=1}^{\lceil n/w \rceil} 2^w - b_k$ is calculated. The binary representation of C (possibly padded) is again divided into blocks of length w: $b_{\lceil n/w \rceil + 1}, \ldots, b_{t_w}$. Finally, the signature of d is $\textsc{Sig} = (\sigma_1, \ldots, \sigma_{t_w})$, where $\sigma_k = H^{b_k}(x_k)$, $k = 1, \ldots, t_w$. The signature size is $t_w \cdot n$ bits and the average cost for signing is $c_{\text{OTSsign}} = (2^w - 1)t_w/2 \cdot c_{\text{HASH}}$.

Verification: Given the digest d, the signature $\textsc{Sig} = (\sigma_1, \ldots, \sigma_{t_w})$, and the verification key Y, the verifier computes $b_1, \ldots, b_{t_w}$ just like the signer and then calculates $y_k = H^{2^w-1-b_k}(\sigma_k)$, $k = 1, \ldots, t_w$. The signature is accepted if $H(y_1\| \ldots \|y_{t_w})$ equals the verification key Y. The average verification cost is exactly the same as the signature cost.

Midpoints Versus Endpoints: From Protocols to Firewalls*

Diana von Bidder-Senn[1], David Basin[1], and Germano Caronni[2]

[1] ETH Zürich, 8092 Zürich, Switzerland
[2] Google Inc., Mountain View
diana.bidder@inf.ethz.ch, basin@inf.ethz.ch, gec@acm.org

Abstract. Today's protocol specifications only define the behaviour of principals representing communication endpoints. But in addition to endpoints, networks contain midpoints, which are machines that observe or filter traffic between endpoints. In this paper, we explain why midpoints should handle protocols differently from endpoints and thus midpoint specifications are needed. With a case study, using the TCP protocol and three different firewalls as midpoints, we illustrate the consequences of the current lack of protocol specifications for midpoints, namely that the same protocol is implemented differently by the different firewalls. We then propose a solution to the problem: We give an algorithm that generates a midpoint automaton from specifications of endpoint automata. We prove that the resulting midpoint automata are correct in that they forward only those messages that could have resulted from protocol-conform endpoints. Finally, we illustrate the algorithm on the TCP protocol.

1 Introduction

Networks contain different kinds of principals. Some are communication endpoints, such as clients and servers, while others are midpoints (also called *middleboxes* [Gro02a]) that forward, filter, or, more generally, transform traffic. A midpoint that simply forwards traffic is straightforward to implement. But as soon as stateful filtering comes into play, the midpoint must know the communication protocols used. This is TCP for packet filters and diverse application-level protocols for application-level firewalls. If a midpoint does not know enough about the protocols it filters, there exist ways to bypass a security policy. A prominent example is sending file-sharing traffic over http when using packet filters.

Protocol specifications are normally written for endpoints. Starting from such specifications, it is not clear how a midpoint should enforce the protocol-conform execution by the endpoints, as it can neither observe nor correctly track the protocol states of endpoints (see Section 3 for more details on this problem). Another problem is that filtering midpoints need to be as secure (i.e. as strict)

* This work was partially supported by armasuisse. It represents the views of the authors.

as possible. However, they should also be user-friendly (and therefore not overly strict). This leads to different interpretations on how a midpoint should handle a given protocol.

The implications of the lack of protocol specifications for midpoints are that manufacturers of devices acting as midpoints have no guidelines on how they should implement a protocol. In practice, midpoint manufacturers implement the same protocol differently, based on their own interpretation of how the midpoint should handle the endpoint data. This implementation is then incrementally adapted based on practical experience. To show how this looks in practice, we present the TCP automata of three different firewalls in Section 4 and analyse their differences.

As a solution to this problem, we show how to systematically generate midpoint specifications from endpoint specifications. We propose an algorithm that, given the protocol automata for the endpoints, generates a protocol automaton for the midpoint. Roughly speaking, the algorithm tracks all possible endpoint states at each point in time, taking into account messages in transit and possible network behaviour. We prove that the midpoint automata constructed forward only those messages that could have resulted from protocol-conform endpoints. Overall, our contributions are an analysis of why different protocol specifications are needed for midpoints than for endpoints, what the implications of the lack of such specifications are, and a solution for this problem. We use the TCP protocol as an example.

The remainder of this paper is organised as follows. In Section 2, we briefly describe background and related work. We then, in Section 3, explain why midpoints are different from endpoints and therefore need their own protocol specifications, before presenting, in Section 4, the results of our case study. In Section 5, we present an algorithm to generate a midpoint automaton from endpoint automata and prove it correct. Finally we conclude and report on future work in Section 6.

2 Background and Related Work

We will use the TCP protocol [ISI81] for our examples, which we briefly summarise here. TCP is a connection-oriented protocol, which is used by applications to transport data to communication partners in a reliable way. TCP itself uses IP to transport the data and just adds a header for flow control, reliability, and multiplexing purposes. As TCP is connection-oriented, there is an initiation — called a *three-way-handshake* (see Figure 1) — and a tear down (see Figure 2) for each connection. With the use of sequence numbers and acknowledgements, TCP ensures that (a copy of) every packet reaches its destination.

For protocol specifications, we use Mealy machines (automata) [Mea55]. A Mealy machine is a six-tuple $M = (Q, \Sigma, \Gamma, \delta, \lambda, q_1)$, where $Q = \{q_1, q_2, ..., q_{|Q|}\}$ is a finite set of *states*; $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_{|\Sigma|}\}$ is a finite *input alphabet*;

Alice		message sent		Bob
CLOSED				LISTEN
SYN-SENT	—	SYN	→	
	←	SYN & ACK	—	SYN-RECEIVED
ESTABLISHED	—	ACK	→	ESTABLISHED

Fig. 1. TCP three-way-handshake

Alice		message sent		Bob
	—	FIN & ACK	→	
	←	FIN & ACK	—	
	—	ACK	→	

Fig. 2. TCP connection tear down

$\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_{|\Gamma|}\}$ is a finite *output alphabet*; $\delta : Q \times \Sigma \to Q$ is the *transition function*; $\lambda : Q \times \Sigma \to \Gamma$ is the *output function*; and $q_1 \in Q$ is the *initial state*. Note that we write $q_i \xrightarrow{x/y} q_j$ to denote that $\delta(q_i, x) = q_j$ and $\lambda(q_i, x) = y$.

Despite the fact that midpoint automata are central for the construction of firewalls and other security gateways, the problem we address has only partly been identified before [Gro00, Gro02b]. To our knowledge, we are the first to present a formal treatment of the problem and to provide an approach for solving it. The closest related work is [BCMG01], which describes how to build a monitor to find out if a system correctly implements an endpoint specification. The authors report on the same problems as ours in determining the state of an endpoint. This problem arises as packets can be reordered or lost between the monitor and the endpoint. They propose several monitoring algorithms. Unfortunately, their algorithm that takes arbitrary reordering and loss into account is very inefficient (the authors call it brute-force) and their refinements are too constrained to be useful in our setting. In principle, their solution could be used to solve our problem, by using one monitor per endpoint and attaching the output of one monitor to the input of the other. As their monitors are inefficient, this is not a practical solution since firewalls should execute very efficiently, i.e., make each decision with minimal overhead.[1] Note that we also solve a different problem than they do: We do not care if the endpoints correctly implement a protocol. Our goal is to only let messages arising from correct protocol runs pass the firewall, independently of how the endpoints create them.

Related areas are firewall testing [MWZ05, ASH03], test-case generation for Mealy machines [Gil61, Cho78, SD88, FvBK$^+$91, CVI89] and the testing of TCP endpoint automata [BFN$^+$06, Pax97]. In firewall testing a midpoint is tested. The starting point of previous work has been the firewall rulesets but not the underlying automata. The area of test case generation for Mealy machines is

[1] This is not a problem with our generated automata. The generation takes time but their execution is very fast.

related, as these methods can be used to generate test cases for the midpoint automata resulting from our algorithm.

3 The Source of the Problem

In this section, we explain why midpoints are different from endpoints and why they thus need a different protocol specification. The problem arises with the filtering midpoints. These base their decisions — basically drop or forward — on the protocol states the endpoints are in. Unfortunately the two endpoints of a connection can be in different states and not all of these states are observable by the midpoint.

Consider the following example: the TCP connection initiation (three-way-handshake) shown in Figure 1. Imagine the second packet gets lost after being forwarded by the midpoint (scenario 1). Alice is now in the state `SYN-SENT`, whereas Bob is in state `SYN-RECEIVED`. To the midpoint, this situation looks the same as the situation where the second packet reaches Alice, but the third packet gets lost before being received by the midpoint (scenario 2).

Given that the midpoint cannot differentiate between these scenarios, in what state should the midpoint be? And how should it react upon receiving a SYN packet from Alice? In scenario 1, the SYN is a retransmission and should be forwarded. Whereas in scenario 2, a SYN does not conform with a correct protocol execution (Alice should not send SYN packets in state `ESTABLISHED`) and should therefore be dropped.

The endpoints see the situation differently. Alice can clearly distinguish between scenario 1, where she would repeat her SYN, and scenario 2, where she would repeat her ACK. Bob cannot distinguish between the scenarios (or at least not until he has seen Alice's reaction), but he does not need to: he would repeat his SYN&ACK in any case.

Another scenario is possible: Alice may have crashed and the SYN at hand could represent a new connection initiation (with the same source port as before). This scenario can also happen later on in a connection. What should the midpoint then do? Should it forward the SYN and risk damage to Bob? Should it just block the packet and hinder Alice from communicating with Bob? Should it send a RST in Bob's name? Should it also send a RST in Alice's name to Bob? All these questions must be answered when giving a midpoint specification of TCP.

In this example, we see that one reason why midpoints cannot always track the protocol states of the endpoints lies in packet loss. But packet loss is only part of the problem. Another reason lies in the fact that certain endpoint constructs lead to ambiguity from the midpoint's perspective. These are:

1. Multiple transitions with the same output:

   Consider two transitions $q_i \xrightarrow{-/a} q_j$ and $q_i \xrightarrow{b/a} q_k$. Assume that the midpoint has previously forwarded b to an endpoint with these two transitions and afterwards receives an a. It cannot know from which transition this a originated.

2. Transitions without output:
   In this case, a midpoint cannot distinguish between the start state of the transition and the end state of the transition (at least until it has seen other, unambiguous output from the endpoint). A special case is hidden states, which are states where all incoming and outgoing transitions have no output. Such states can never be identified by a midpoint.
3. A packet can be sent in different states:
   This makes an unambiguous mapping between packets and states impossible. While this is not a source of tracking problems, it does make recovering from them difficult.

4 Case Study: Differences in Midpoints Based on TCP

In the last section, we explained why it is nontrivial to build a midpoint from endpoint specifications. In this section, we now show the implications of a lack of midpoint specifications by documenting the current state of affairs. For this, we took three commonly used firewalls — Checkpoint [Che], netfilter / iptables [ipt], and ISA Server [Isa] — and reverse engineered them, testing them against our (as there is no other) TCP midpoint specification given in [SBC05] and then analysing the results by hand.

As a result, we derived three distinct TCP automata (see [vBBC07] for details). Below we describe three of the differences:

A 'clean' three-way-handshake is not enforced. To initiate a TCP connection, a so called three-way-handshake is used (see Figure 1). So let us assume the firewall has accepted a SYN from an endpoint E_0 (Alice) to another endpoint E_1 (Bob). If there is now a SYN&ACK from E_1 to E_0, then everything works as expected: the packet should be let through and the firewall should enter the next state. If there is another SYN from E_0 to E_1, then this will be a retransmission (it could be that the first SYN was lost between the firewall and E_1) and should be allowed as well. If there is a RST from E_1 to E_0, then E_1 does not want this connection, the packet should be let through, and the TCP automaton initialised. All other packets make no sense at this time and therefore should be blocked. Unfortunately, in all of the tested firewalls, additional packets were let through. As one example, a FIN from E_1 to E_0 is allowed during connection initiation in netfilter. But there is no connection to be closed: If E_1 is not accepting the connection, then it would send a RST.

After a FIN, data from both sides is still accepted. If E_0 sends a FIN to E_1, then this means that E_0 wants to close the connection. After this FIN, E_1 is still allowed to send data, but E_0 is not (it makes not sense to send data after requesting to close the connection), except for the ACK belonging to E_1's FIN. As packets may not arrive in their correct ordering at the firewall, the firewall cannot just drop all packets from E_0 after having seen a FIN from B. But the firewall should just let through older packets (based on the sequence number), a retransmission of the FIN, and the ACK to E_1's FIN&ACK (after having received E_1's FIN&ACK). To accomplish this task, the firewall has to keep track of the sequence numbers. This appears not to be done and therefore too many packets are let through.

SYNs are accepted during already established connections. SYNs are only used for connection initiation. That means that if a connection is fully established, there will be no more legitimate SYNs (based on the sequence numbers) belonging to that connection. But netfilter and ISA Server accept SYNs (from the initiator of the connection) all the time. Checkpoint does block the SYNs, but always allows SYN & ACK, which is not much better.

These findings show that there is a lack of consensus, at best, and a general lack of understanding, at worst, about how TCP should be handled by a firewall. The result is that every vendor does something different. We would like to contribute a solution to this problem by showing, systematically, how to construct midpoint specifications from endpoint specifications.

5 Construction of a Midpoint Automaton from Endpoint Automata

In the preceding sections, we saw why there is a need for midpoint specifications. Basically there are two ways to construct such a specification: write it directly or generate it from the endpoint specifications. The first alternative has two major drawbacks: 1) the consistency with the endpoint specifications must somehow be assured and 2) it requires additional work for the protocol designers. The second alternative overcomes both problems.

5.1 Setting

We subsequently consider only two-party protocols, i.e. protocols for only two endpoints. This covers most network protocols.[2] Let E_0 and E_1 be the endpoints and M the midpoint through which communication passes. Communication takes place in the form of messages, where the endpoint specification of the communication protocol determines when an endpoint may send which kind of message. For every message arriving at the midpoint, the midpoint can either forward the message or drop it (Figure 3).

E0
m
M
m'
E1
E0
m
M
E1

Fig. 3. A message m from endpoint E_0 to endpoint E_1 is forwarded (left) or dropped (right) by the midpoint M

We write $X \rightarrow Y : m$ to express that the message m is sent from the endpoint X to the other endpoint Y, where $X \in \{E_0, E_1\}$, $Y \in \{E_0, E_1\}$, and $Y \neq X$. As

[2] It is straightforward to extend our approach to protocols with more endpoints.

there is a midpoint M between E_0 and E_1, every $X \to Y : m$ can be divided into the two parts $X \to M : m$ and $M \to Y : m'$. This makes explicit on which side of the midpoint a message is and also simplifies the specification of the actions of the midpoint: $m' = m$ if the midpoint forwards the message unaltered, $m = -$ (where $-$ signifies no external output) if the midpoint drops the message, and $m' \neq m$ if the midpoint alters the message before forwarding it. The messages may be altered, for example, when using Network Address Translation (NAT) in the firewall. For the sake of simplicity, we will not consider this case.

We will construct our midpoints to be *permissive* rather than *restrictive*. This means that our midpoint forwards messages if they could have resulted from protocol-conform endpoints. Thus our midpoint can possibly accept an incorrect message, but only in the cases where there is a scenario where this message could occur.

For the transport of the messages between the endpoints (via the midpoint), we assume a network that either (1) delivers messages, although not necessarily preserving the order, or (2) looses them.

5.2 Idea

Before giving the construction of a midpoint automaton from endpoint automata in Section 5.3, we first sketch the ideas behind our construction.

We model the global state of a system (the endpoints, midpoint, and network) at some time t as a state $st^t = (q_0^t, q_M^t, q_1^t, net^t)$, where q_i^t is the state of endpoint E_i at time t, q_M^t is the state of the midpoint M at time t, and net^t consists of all messages travelling between the endpoints at time t.

The midpoint M has to base its actions on the state of its environment. If M could observe all actions in the system, $q_M^t = (q_0^t, q_1^t, net^t)$ would hold at any time t, meaning that M always knows the exact states of the endpoints and the contents of the network. But midpoints generally cannot always determine the correct values of these components and hence we will let the state of our midpoint be a set of such triples, where each of these triples represents a possibly correct view of the system. Thus the triples of one state q_M^t are equivalent in the sense that they are not distinguishable by the midpoint with its current knowledge, i.e. based on the traffic it has previously observed.

To provide further intuition about the functioning of a midpoint, let us consider an example. Suppose the system starts in the global state $st^1 = (q_1, \{(q_1, q_1, net^1)\}, q_1, net^1)$.[3] Assume the following steps are taken:

1. M sends (forwards) a message x to E_0.
   To track the fact that the network now contains x, M must change its state to $q_M^2 = \{(q_1, q_1, net^2)\}$, where $net^2 = net^1 \cup \{M \to E_0 : x\}$. The global state is then $st^2 = (q_1, \{(q_1, q_1, net^2)\}, q_1, net^2)$.

[3] This means that E_0 and E_1 both are in their start states, the network content is net^1 and the midpoint knows about all this. Note that q_1 of E_0 and q_1 of E_1 are not the same as they do not belong to the same automaton.

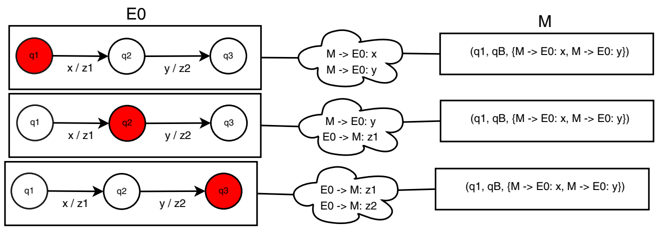


Fig. 4. Two consecutive endpoint transitions

2. x is received by E_0 and used as input to its automaton. Suppose, for E_0 in state q_1, there are only two transitions: $q_1 \xrightarrow{x/y} q_2$ and $q_1 \xrightarrow{-/z} q_3$. As M does not know if and when E_0 makes a transition, it cannot directly act on this step and thus its state is wrong. The global state is $st^3 = (q_2, \{(q_1, q_1, net^2)\}, q_1, net^3)$, where $net^3 = net^2 \setminus \{M \to E_0 : x\} \cup \{E_0 \to M : y\}$.
3. A message y reaches M.
   To take a correct decision, M must compute what could have happened (all possible successor steps) since its last step. This is either:
   - nothing, i.e., (q_1, q_1, net^2),
   - E_0 consumes x, makes a transition to q_2, and outputs y: (q_2, q_1, net^3), or
   - E_0 makes a transition to q_3, and outputs z: $(q_3, q_1, net^2 \cup \{E_0 \to M : z\})$.

   Now, M can determine its reaction on y. If there is one or more triple having y in its net (a possibly correct scenario where y occurred), y is forwarded and M's next state will consist of all these matching triples with their nets updated: $q_M^4 = \{(q_2, q_1, net^4)\}$, where $net^4 = (net^3 \setminus \{E_0 \to M : y\}) \cup \{M \to E_1 : y\}$. The global state is $st^4 = (q_2, \{(q_2, q_1, net^4)\}, q_1, net^4)$.

In the example above, there was only one endpoint transition between two consecutive midpoint transitions. In such a case, tracking (computing all possible successor states) is not difficult. The situation is more complex if more than one endpoint transition can happen between two consecutive midpoint transitions. Figure 4 provides an example. Since communication need not be order preserving, after two consecutive endpoint transitions, the second message ($E_0 \to M : z2$) may reach the midpoint first. Thus it would not be enough if the midpoint computed only the next possible state (reachable in one step), but all possible successor states are needed, as only this would lead to $\{(q_1, q_B, \{M \to E_0 : x, M \to E_0 : y\}), (q_2, q_B, \{M \to E_0 : y, E_0 \to M : z1\}), (q_3, q_B, \{E_0 \to M : z1, E_0 \to M : z2\})\}$ and thus lead to the correct action — forwarding $z2$ — and the correct next state $\{(q_3, q_B, \{E_0 \to M : z1, M \to E_1 : z2\})\}$.

Let us return to our first example to illustrate why all possibly correct messages must be forwarded. Assume that we have the following sequence of actions:

1. M forwards a message x to E_0.
2. x is lost by the network.
3. E_0 takes the transition to state q_3.

4. An intruder sends message y (which is incorrect at q_3).
5. y reaches M.

For the midpoint, this scenario looks exactly the same as the one before. But here y is an incorrect message. As we never want to block correct messages (our decision for a *permissive* rather than *restrictive* midpoint), we have to accept y here (it could be the correct one from above).

5.3 Construction

We will now give the technical details of our construction of a midpoint automaton from endpoint automata. A two-party protocol p can be specified by two Mealy automata (one for each endpoint):

$$A_0 = (Q_0, \Sigma_0, \Gamma_0, \delta_0, \lambda_0, q_{0,1}) \qquad \text{and} \qquad A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, \lambda_1, q_{1,1}).$$

Note that $\Sigma_0 = \Gamma_1$ and $\Gamma_0 = \Sigma_1$ since these automata must be able to communicate with each other. Often even A_0 and A_1 are the same.

The network can be modelled as a multiset (also called bag) *net*, which stores all messages in transit between the midpoint and the endpoints. Specifically

$$net \subseteq \mathcal{M}(S) = \{x' \mid x \subseteq S, x' =_s x\},$$

where S is the set of messages allowed by the protocol and $=_s$ denotes set equality (the sets contain the same elements, ignoring repetition).

In our construction, *net* will be part of a state of a deterministic automaton. Since we cannot handle a network of infinite size we must forbid actions of the endpoints and the network that can put infinitely many packets into the network. Hence, for the endpoints, we forbid loops without input in their protocol automata. For the network, we do not model message duplication. These restrictions are not problematic as the former should not be present and the latter can easily be detected and handled on another layer. Thus, it suffices to consider

$$net \subseteq \mathcal{P}(S), \text{where } S = \{X \to Y : m | X, Y \in \{E_0, E_1, M\}, Y \neq X, m \in (\Sigma_0 \cup \Gamma_0)\}.$$

Before starting with the construction of the midpoint automaton, we need to define the actions that are possible in our system. These are either transitions by the endpoints or the midpoint, based on their automata, network loss, or a message inserted by an intruder.

Definition 1. *$st^{t+1} = (q_0^{t+1}, q_M^{t+1}, q_1^{t+1}, net^{t+1})$ is a* successor state *of $st^t = (q_0^t, q_M^t, q_1^t, net^t)$, denoted $st^t \vdash st^{t+1}$, if one of the following conditions holds.*

Midpoint transition: For any $msg \in net$ with $msg = (E_i \to M : m)$,

$$\begin{aligned} q_0^{t+1} &= q_0^t, \\ q_1^{t+1} &= q_1^t, \\ q_M^{t+1} &= \delta_M(q_M^t, msg), \\ net^{t+1} &= (net^t \setminus \{msg\}) \cup \{\lambda_M(q_M^t, msg)\}. \end{aligned} \tag{1a}$$

Correct endpoint transition: For any $msg \in net^t$ *with* $msg = (M \to E_i : m)$ *(the endpoint taking an input from the network) or for* $m = -$ *(no input)*

$$\begin{aligned} msg' &= \begin{cases} (E_i \to M : \lambda_i(q_i^t, m)) & if \lambda_i(q_i^t, m)) \neq -, \\ - & otherwise, \end{cases} \\ net^{t+1} &= (net^t \setminus \{msg\}) \cup \{msg'\}, \\ q_i^{t+1} &= \delta_i(q_i^t, m), \\ q_{1-i}^{t+1} &= q_{1-i}^t, \\ q_M^{t+1} &= q_M^t. \end{aligned} \tag{1b}$$

Incorrect transition:

$$\begin{aligned} msg' &\in (\Gamma_i \setminus \{\lambda_i(q_i^t, m) | (M \to E_i : m) \in net^t \ or \ m = -\}), \\ net^{t+1} &= net^t \cup \{msg'\}, \\ q_0^{t+1} &= q_0^t, \\ q_1^{t+1} &= q_1^t, \\ q_M^{t+1} &= q_M^t. \end{aligned} \tag{1c}$$

Network loss: for any $msg \in net^t$,

$$\begin{aligned} q_0^{t+1} &= q_0^t, \\ q_1^{t+1} &= q_1^t, \\ q_M^{t+1} &= q_M^t, \\ net^{t+1} &= net^t \setminus \{msg\}. \end{aligned} \tag{1d}$$

Note that as the network is modelled by a set, the permutation of messages is handled implicitly. Furthermore, note that an endpoint transition must not produce output. We denote empty output as '$-$'.

Definition 2. *Transitions* (1a), (1b) *and* (1d) *represent* correct transitions. *We denote a message resulting from a correct transition as a* correct message *and all other messages as* incorrect messages.

Definition 3. *A* correct trace *is a trace* $st^1 \vdash st^2 \vdash ... \vdash st^n$, *where every transition* $st^i \vdash st^{i+1}$, *with* $1 \leq i < n$, *is a correct transition.*

Definition 4. *The* message history *of a trace* $tr = st^1 \vdash st^2 \vdash ... \vdash st^n$ *is a sequence of messages* $m_1, m_2, ..., m_t$, *where* t *is the number of non-midpoint transitions in* tr *that produce output, and* m_i *is the output from the* i*th of these transitions.*

Definition 5. *The* midpoint message history *of a trace* $tr = st^1 \vdash st^2 \vdash ... \vdash st^n$ *is a sequence of messages* $m_1, m_2, ..., m_s$, *where* s *is the number of midpoint transitions in* tr, *and* m_i *is the input of* M *at its* i*th transition in* tr.

Definition 6. *Two* traces *are* midpoint equivalent *if they have the same midpoint message history.*

Definition 7. *Two* triples (a, c, d) *and* (e, g, h) *are* midpoint equivalent *if there exist two midpoint equivalent traces* tr_1 *and* tr_2 *with* $tr_1 = s^1 \vdash s^2 \vdash ... \vdash (a, b, c, d)$ *and* $tr_2 = st^1 \vdash st^2 \vdash ... \vdash (e, f, g, h)$.

To have a correctly functioning midpoint, two properties about a midpoint state q_M must be satisfied: 1) one of the triples in q_M is the correct one (Definitions 8 and 9) and 2) only possibly correct messages are forwarded (Definition 10).

Definition 8. q^t_M *is a* correct tracking at time t *if* q^t_M *is neither too small nor too large.* Not too small *means that* $(q^t_0, q^t_1, net^t) \in q^t_M$. Not too large *means that all* $q \in q^t_M$ *are midpoint equivalent to* (q^t_0, q^t_1, net^t).

Definition 9. M tracks endpoints correctly *if for every midpoint transition* $(q^{n-1}_0, q^{n-1}_M, q^{n-1}_1, net^{n-1}) \vdash (q^n_0, q^n_M, q^n_1, net^n)$ *in a trace,* q^n_M *is a correct tracking at time* n.

Definition 10. M computes outputs correctly *if for every trace* $tr = st^1 \vdash \ldots \vdash st^n$ *and every* t, $1 \leq t \leq n$ *we have:*

$$\lambda_M(q^t_M, E_i \to M : m) = \begin{cases} M \to E_j : m & \text{if } E_i \to M : m \text{ occurs in the message history of any trace } tr' \text{ which is midpoint equivalent to } tr, \\ - & \text{otherwise.} \end{cases}$$

where $j = 1 - i$.

As M cannot distinguish between tr and tr' in the above, it must forward all messages that occur in any of these traces, in order to avoid ever dropping a correct message.

Based on A_0 and A_1, we will now construct a Mealy automaton A_M for the handling of protocol p by the midpoint:

$$\begin{aligned} A_M &= (Q_M, \Sigma_M, \Gamma_M, \delta_M, \lambda_M, s_M) \\ Q_M &= \mathcal{P}(Q_0 \times Q_1 \times net) \\ \Sigma_M &= \{E_0 \to M : a \mid a \in (\Gamma_0 \setminus \{-\}\} \cup \{E_1 \to M : a \mid a \in (\Gamma_1 \setminus \{-\}\} \\ \Gamma_M &= \{M \to E_0 : a \mid a \in (\Sigma_0 \setminus \{-\}\} \cup \{M \to E_1 : a \mid a \in (\Sigma_1 \setminus \{-\}\} \cup \{-\} \\ q_{M,1} &= \{(q_{0,1}, q_{1,1}, \{\})\} \end{aligned}$$

Before we define the functions δ_M and λ_M, we first analyse the different possible scenarios. We do this with the help of Figure 5. There we describe the relationship between the actions of an endpoint E_0 (the situation for E_1 is analogous), the network, and the midpoint M. In particular, we consider how the four different types of transitions an endpoint can take ($x/-$, x/y, $-/y$, and $-/-$, for

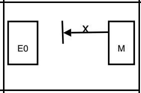
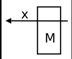
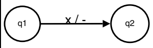
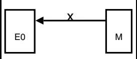
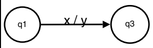
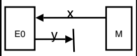
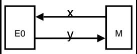
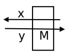
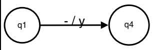
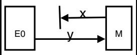
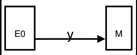
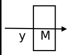
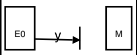
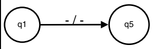
Endpoint E_0	Network	Midp.	correct midpoint transition	Eq.
-			$\delta_M(q_M, -) = \{(q_1, q_{E_1}, net_M \setminus \{M \to E_0 : x\})\}$ $\lambda_M(q_M, -) = -$	(2c)
			$\delta_M(q_M, -) = \{(q_2, q_{E_1}, net_M \setminus \{M \to E_0 : x\})\}$ $\lambda_M(q_M, -) = -$	(2d)
			$\delta_M(q_M, -) = \{(q_3, q_{E_1}, net_M \setminus \{M \to E_0 : x\})\}$ $\lambda_M(q_M, -) = -$	(2d) (2c)
			$\delta_M(q_M, E_0 \to E_1 : y) = \{(q_3, q_{E_1}, (net_M \setminus \{M \to E_0 : x\}) \cup \{M \to E_1 : y\})\}$ $\lambda_M(q_M, E_0 \to E_1 : y) = E_0 \to E_1 : y$	(2d)
			$\delta_M(q_M, E_0 \to E_1 : y) = \{(q_4, q_{E_1}, (net_M \setminus \{M \to E_0 : x\}) \cup \{M \to E_1 : y\})\}$ $\lambda(q_M, E_0 \to E_1 : y) = E_0 \to E_1 : y$	(2c) (2e)
			$\delta_M(q_M, E_0 \to E_1 : y) = \{(q_4, q_{E_1}, net_M \cup \{M \to E_1 : y\})\}$ $\lambda(q_M, E_0 \to E_1 : y) = E_0 \to E_1 : y$	(2e)
			$\delta_M(q_M, -) = \{(q_4, q_{E_1}, net_M)\}$ $\lambda_M(q_M, -) = -$	(2e) (2c)
			$\delta_M(q_M, -) = \{((q_5, q_{E_1}, net_M)\}$ $\lambda_M(q_M, -) = -$	(2e)

Fig. 5. A transition in an endpoint, from a midpoint's view

$x \in \Sigma_0$, $y \in \Gamma_0$) look from the endpoints', the network's, and the midpoint's respective point of view. These are shown in columns $1 - 3$, where one row represents one case. Note that one view (row) of one principal can belong to several views of another principal.

In the fourth column, the correct midpoint transition is shown. That is the transition the midpoint must take if it wants to correctly track the endpoint's state and the messages in the network.[4] For this we assume $q_M = \{(q_1, q_{E_1}, net_M)\}$ to be the state of the midpoint after its last transition (where applicable, this is forwarding x). Note that net_M contains all the messages in the network. Therefore a message has to be removed from net_M if it is no longer in the network, either because it was consumed by an endpoint or midpoint, or lost by the network.

As an example, let us explain the contents of the third row (in the first column, the third and the forth row coincide). Here a message x is forwarded by M to E_0 (3rd column), i.e. $M \to E_0 : x$. This message then reaches E_0 (2nd column), which uses it as input to its x/y-transition (1st column). After this transition, E_0 is in state q_3 (1st column) and a message y ($E_0 \to M : y$) has been inserted

[4] We will later give definitions of δ_M and λ_M that incorporate all these scenarios. The fifth column gives the number of the corresponding equations.

into the network (2nd column). This message is then lost by the network (2nd column). To correctly represent these actions, M has to change its state as given: E_0 is now in state q_3 and the *net* no longer contains x (4th column). Note that net_M does not contain y as its insertion is compensated by its removal.

With the help of Figure 5, we will now define the successor function. This function computes all direct successor states of a triple of the midpoint state (the transition function will then later choose some of these triples, based on its input). The figure illustrates why we sometimes have more than one possible successor state: the midpoint (3rd column) cannot distinguish all the scenarios (rows). Note that Figure 5 only considers one endpoint, whereas *succ* considers both endpoints (the equations (2d) and (2e) for E_0 correspond to the equations (2f) and (2g) for E_1).

$$\begin{aligned}
succ(q_M) = \bigcup_{q\in q_M} \bigcup_{(M\to E_0:m_1)\in net_M} \bigcup_{(M\to E_1:m_4)\in net_M} \bigcup_{msg\in net_M} \quad & \qquad (2a)\\
\{(q_0, q_1, net_M), & \qquad (2b)\\
(q_0, q_1, net_M \setminus \{msg\}), & \qquad (2c)\\
(\delta_0(q_0, m_1), q_1, (net_M \setminus \{M \to E_0 : m_1\}) \cup m_2) & \qquad (2d)\\
(\delta_0(q_0, -), q_1, net_M \cup m_3) & \qquad (2e)\\
(q_0, \delta_1(q_1, m_4), (net_M \setminus \{M \to E_1 : m_4\}) \cup m_5), & \qquad (2f)\\
(q_0, \delta_1(q_1, -), net_M \cup m_6)\} & \qquad (2g)
\end{aligned}$$

where

$$m_2 = \begin{cases} \{E_0 \to M\} : \lambda_0(q_0, m_1) & \lambda_0(q_0, m_1) \neq -, \\ \emptyset & \text{otherwise,} \end{cases} \qquad (2h)$$

$$m_3 = \begin{cases} \{E_0 \to M : \lambda_0(q_0, -)\} & \lambda_0(q_0, -) \neq -, \\ \emptyset & \text{otherwise,} \end{cases} \qquad (2i)$$

$$m_5 = \begin{cases} \{E_1 \to M : \lambda_1(q_1, m_4)\} & \lambda_1(q_1, m_4) \neq -, \\ \emptyset & \text{otherwise,} \end{cases} \qquad (2j)$$

$$m_6 = \begin{cases} \{E_1 \to M : \lambda_1(q_1, -)\} & \lambda_1(q_1, -) \neq -, \\ \emptyset & \text{otherwise.} \end{cases} \qquad (2k)$$

The function *succ* computes all the states that are reachable in one step by an endpoint or the network. Since we are interested in all possible successor states, we must compute the closure of *succ*, defined as

$$cl(succ(x)) = \bigcup_{i=0}^{\infty} succ^i(x). \qquad (3)$$

Observe that the closure is monotonic. It also has an upper bound, namely

$$cl(succ(q_M^t)) \subseteq \mathcal{P}(\{(q_0, q_1, n) | q_0 \in Q_0, q_1 \in Q_1, n \in net\}).$$

Hence, as Q_0, Q_1, and *net* are finite, $cl(succ(q_M^t))$ is also finite.

We now can define δ_M. The idea is to let our midpoint track all possible actions. We do this by first calculating the closure of all possible next states before actually executing a transition based on them.

$$\delta_M(q_M^t, m) = \bigcup_{(q_0, q_1, net_M) \in cl(succ(q_M^t))} \{(q_0, q_1, (net_M \backslash \{m\}) \cup \lambda_M(q_M, m))\} | m \in net_M\} \tag{4}$$

Note that $cl(succ(q_M^t))$ represents all possible successor states of q_M^t, whereas $\delta_M(q_M^t, m)$ only contains those successor states of q_M^t that can be reached with a message m. λ_M is now straightforward: If there is any triple where the input occurs, i.e. the message is correct in some midpoint-equivalent trace, the input is forwarded.

$$out((q_0, q_1, net_M), E_i \to M : y) = \begin{cases} (M \to E_j : y) & \text{if } \{E_i \to M : y\} \in net_M, \\ - & \text{otherwise.} \end{cases} \tag{5}$$

where $j = 1 - i$, and

$$\lambda_M(q_M, m) = \begin{cases} out(q, m) & \text{if there exists a } q \in cl(succ(q_M)) \text{ with } (out(q, m) \neq -), \\ - & \text{otherwise.} \end{cases} \tag{6}$$

Note that for each m, there is at most one non-empty (not '$-$') value for $out(q, m)$. Hence, λ_M is well-defined. This is due to the fact that our midpoint either drops or forwards a message. This would have to be revised for a midpoint that alters messages (e.g. a firewall performing Network Address Translation).

5.4 Correctness

As stated in Section 5.2, a correctly functioning midpoint must satisfy two properties: 1) one of its state triples is correct and 2) only possibly correct messages are forwarded. In this section, we prove that a midpoint, constructed as described in Section 5.3, satisfies the above properties.

Property I: one state triple is correct.
We prove the first property with the help of the following lemmas.

Lemma 1. *Given an M produced by our midpoint construction and a trace $st^1 \vdash st^2 \vdash ... \vdash st^n$, if there is a correct tracking at time $t1$, then the tracking after the next midpoint transition $st^{t2} \vdash st^{t2+1}$, $t2 \geq t1$, is also correct.*

Lemma 2. *Given an M produced by our midpoint construction and a trace $tr = st^1 \vdash st^2 \vdash \cdots \vdash st^n$, for any midpoint transition $st^{t2} \vdash st^{t2+1}$, there is a correct tracking at each time $t1$, with $1 \leq t1 \leq t2 < n$.*

Lemma 3. *Given an M produced by our midpoint construction, M tracks endpoints correctly (as defined in Definition 9).*

Lemma 3 follows from the other two. If for every midpoint transition there exists an earlier correct tracking (Lemma 2), then the tracking after the midpoint transition is correct (Lemma 1). Hence the tracking after every midpoint transition is correct.

Proof sketch for Lemma 1
We will only sketch the proof of the first lemma. The proof in its entirety can be found in [vBBC07].

Recall that δ_M consists of two parts: 1) compute all possible successor states using $cl(succ(q_M^{t1}))$, and 2) keep only the state triples that are feasible with respect to a given message. The second part directly reflects the definition of a midpoint transition. To prove the first part, we show that *succ* can track one non-midpoint transition correctly, and that taking the closure of *succ* computes any number of non-midpoint transitions correctly.

Proof of Lemma 2
We prove the lemma by induction on the midpoint transitions in a trace.

Base Case, the first midpoint transition in a trace.
The first state st^1 is a correct tracking at time 1:

$$st^1 = (q_0^1, q_M^1, q_1^1, net^1) = (q_{0,1}, (q_{0,1}, q_{1,1}, \{\}), q_{1,1}, \{\}).$$

The first midpoint transition cannot take place before $st^1 \vdash st^2$. By Lemma 1, this means that the tracking after the first midpoint transition is correct.

Step Case, the nth midpoint transition, $n > 1$.
By the induction hypothesis, the tracking is correct after the $(n-1)$th midpoint transition. By Lemma 1, this implies that the tracking is also correct after the nth midpoint transition. QED.

Property II: only possibly correct messages are forwarded.

Lemma 4. *M computes outputs correctly (as defined in Definition 10).*

We show the correctness of $\lambda_M(q_M^t, msg)$ in two steps:

1. $msg = (E_i \rightarrow M : m)$ was inserted by a correct transition.
   There is is a triple $q_{corr}^t \in q_M^t$ with $q_{corr}^t = (q_0^t, q_1^t, net^t)$ (see the proof of Lemma 1 in [vBBC07]). The output of this triple, defined by Equation (5), is correct, namely $msg' = (M \rightarrow E_j : m)$. For every other triple q, $out(q, msg)$ is either msg' or $-$. Thus, by Equation (6), $\lambda_M(q_M^t, E_i \rightarrow M : m)$ is correct.
2. msg was inserted by an incorrect transition.
   As seen above, there can only be a $(q_0, q_1, net) \in q_M^t$ with $msg \in net$ if this represents a possibly correct scenario. But, in this case, forwarding msg is correct.[5] QED.

[5] Note that in this case, the endpoints might not be able to continue their run of the protocol; the incorrect endpoint is only able to continue to send messages if they belong to a possibly correct scenario. This is the price of having a permissive firewall.

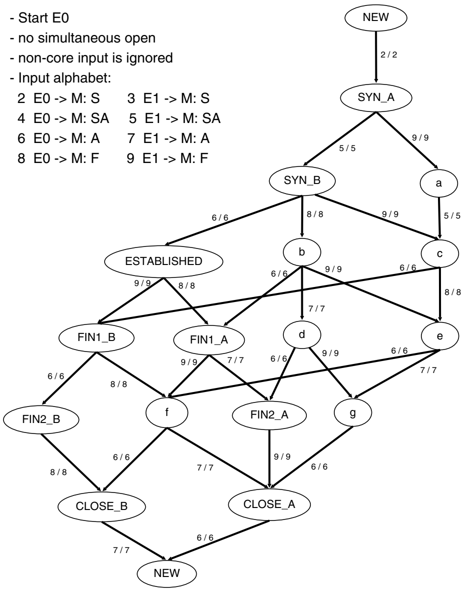


Fig. 6. Midpoint Automaton for TCP

5.5 Discussion

In Section 4, we analysed the TCP automata of several firewalls. We now compute the TCP midpoint automaton using the construction just presented. For endpoint automata, we use the automaton from the TCP specification for endpoints [ISI81, page 23].

The endpoint automaton in the TCP specification combines the initiator and the responder role. These roles are handled differently by firewalls, which distinguish between the networks outside and behind the firewall. Normally only one side is allowed to initiate a connection. Therefore we made two copies of the TCP endpoint automaton from the specification, one for each of the roles, which we adapt as follows. We chose E_0 to play the role of the *initiator*. Therefore we denote the state `CLOSED` as the start state of automaton A_0 and delete the state `LISTEN` from A_0. Furthermore, as we are only interested in one run of the protocol, we name the end state of A_0 `CLOSED2` (instead of `CLOSED`). To let E_1 play the role of the *responder*, we denote the state `LISTEN` as the start state of A_1 and delete the state `SYN-SENT` and the transitions from state `CLOSED` to state `LISTEN` from A_1. The resulting, minimised midpoint automaton can be found in Figure 6. Note that although this automaton represents only a subset of the TCP protocol this is not a limitation of our algorithm. Our algorithm can handle sequence numbers and the like if they are part of an endpoint automaton.

As expected, in each state of the midpoint, there is considerable uncertainty about the exact state of the endpoints. This is reflected in the fact that some midpoint states consist of over 60 triples. Despite this, the automaton is of manageable complexity, in particular the number of outgoing transitions per state is small (1 – 3). The multiple transitions reflect the (limited) ways that messages can be sent independently by the endpoints and how they can be reordered by the network.

Note too that our midpoint automaton has 7 more states (a – g) than our reverse-engineered TCP automata. This reflects the additional complexity necessary to properly track possible network events. Let us illustrate this with an example. In our midpoint automaton, it can clearly be seen that, to get from state `SYN_B` to state `FIN1_B`, two messages — an ACK from E_0 and a FIN from E_1 — are needed. These messages are independent and thus can arrive in either order at the firewall. If we look how actual firewalls handle this, we see that the intended order of sending the ACK before the FIN leads to the same result, but that the opposite order ends in state `ESTABLISHED`, leaving us without an explanation why the FIN needs to be allowed in state `SYN_B`.

Our construction builds permissive midpoint automata. This reflects our decision not to penalise protocol-conform endpoints for actions of the environment (here the network). But it is a simple matter to modify the approach to construct restrictive midpoint automata. These can be built by stopping — i.e. dropping everything from then on — at states that consist of more than one triple. But building a restrictive automaton makes little sense with current protocols: It requires dropping more or less everything, as there will be uncertainty already after a few packets.

As discussed in Section 3, a midpoint may send (spoofed) messages to the endpoints to tear down (reject) an unwanted connection. Note that the decision whether such a message is sent is part of the midpoint's policy, not of the automaton. Therefore, similar to instantiating a new instance of the midpoint automaton after receiving the first packet of a connection, the sending of such messages should result in the deletion of the corresponding instance of the midpoint automaton.

6 Conclusion and Future Work

In this paper, we have shown why midpoints must behave, and hence be specified, differently from endpoints. Furthermore we have given an algorithm to generate midpoint automata from endpoint automata. Our solution should be of interest to at least two groups: those building midpoints and those analysing (e.g. testing) them. Both groups will benefit from having a general method to systematically construct midpoint specifications from those for endpoints.

The construction presented has two minor limitations: it requires that the endpoint automata do not have loops without input and it does not take duplication in the network into account. The first point is unproblematic, as loops without input should not be present since these would enable one endpoint to loop infinitely without communicating (only "talking" not "listening") with the other endpoint. We believe the problem of duplicates (or retransmissions) should be solved independent of protocol automata. The midpoint should remember the packets seen (unique id) and its decision, and then apply the same decision to duplicates received later. We intend to investigate if this solution is feasible.

We plan to use our algorithm in the area of firewall conformance testing. Namely, when testing a given firewall, we first determine the differences between the automaton implemented in the firewall and the generated midpoint automaton A_{M_p} for every protocol p.[6] Then the user can decide if the additional or missing transitions represent a problem. If not, we continue with the test of the policy, as described in [SBC05], based on the protocol automaton of the given midpoint. In this way, firewall conformance testing (and testing of other kinds of midpoints as well) is possible and we can give the users information (and control) on the strictness of their firewalls.

References

[ASH03] Ehab Al-Shaer and Hazem Hamed. Management and translation of filtering security policies. In *Proc. 38th Int. Conf. Communications (ICC 2003), IEEE*, pages 256–260, May 2003.

[BCMG01] Karthikeyan Bhargavan, Satish Chandra, Peter J. McCann, and Carl A. Gunter. What packets may come: automata for network monitoring. In *POPL*, pages 206–219, 2001.

[BFN$^+$06] Steve Bishop, Matthew Fairbairn, Michael Norrish, Peter Sewell, Michael Smith, and Keith Wansbrough. Engineering with logic: HOL specification and symbolic-evaluation testing for TCP implementations. In *POPL*, pages 55–66. ACM, 2006.

[Cho78] Tsun S. Chow. Testing software design modeled by finite-state machines. In *IEEE Transactions on Software Engineering*, volume SE-4, pages 178–187, May 1978.

[CVI89] Wendy Y. L. Chan, Son T. Vuong, and M. Robert Ito. An improved protocol test generation procedure based on UIOS. In *SIGCOMM*, pages 283–294, 1989.

[ipt] Harald Welte et al. netfilter/iptables (ip_conntrack 2.1). `http://www.netfilter.org/`.

[FvBK$^+$91] Susumu Fujiwara, Gregor von Bochmann, Ferhat Khendek, Mokhtar Amalou, and Abderrazak Ghedamsi. Test selection based on finite state models. volume 17, pages 591–603, 1991.

[Gil61] A. Gill. State-identification experiments in finite automata. In *Information and Control, vol. 4*, pages 132 – 154, 1961.

[Gro00] Network Working Group. RFC 2979: Behavior of and requirements for internet firewalls, October 2000.

[Gro02a] Network Working Group. RFC 3234: Middleboxes: Taxonomy and issues, February 2002.

[Gro02b] Network Working Group. RFC 3360: Inappropriate tcp resets considered harmful, August 2002.

[ISI81] University of Southern California Information Sciences Institute. RFC 793: Transmission control protocol, September 1981.

[6] In the ideal case there are no differences at all or any differences are at least given (and justified) by the midpoint vendor.

[Che] Checkpoint Software Technologies Ltd. Checkpoint R55W. http://www.checkpoint.com/.

[Mea55] G.H. Mealy. Method for synthesizing sequential circuits. In *Bell System Technical Journal*, volume 34, pages 1045–1079, 1955.

[Isa] Microsoft. ISA server v4.0.2161.50. http://www.microsoft.com/isaserver/default.mspx.

[MWZ05] A. Mayer, A. Wool, and E. Ziskind. Offline firewall analysis. In *International Journal of Information Security*, pages 125–144, 2005.

[Pax97] Vern Paxson. Automated packet trace analysis of TCP implementations. In *SIGCOMM*, pages 167–179, 1997.

[SBC05] Diana Senn, David Basin, and Germano Caronni. Firewall conformance testing. In *Lecture Notes in Computer Science*, volume 3502, pages 226–241, May 2005.

[SD88] Krishan Sabnani and Anton Dahbura. A protocol test generation procedure. In *Computer Networks and ISDN Systems 15*, pages 285–297, 1988.

[vBBC07] Diana von Bidder, David Basin, and Germano Caronni. Midpoints versus endpoints: From protocols to firewalls. Technical report 552, ETH Zürich, Department of Computer Science, March 2007.

An Adversary Aware and Intrusion Detection Aware Attack Model Ranking Scheme*

Liang Lu, Rei Safavi-Naini, Jeffrey Horton, and Willy Susilo

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering
University of Wollongong, Australia
{ll97,rei,jeffh,wsusilo}@uow.edu.au

Abstract. A successful computer system intrusion is often resulted from an attacker combining exploits of individual vulnerability. This can be modelled by attack models and attack graphs to provide a global view on system security against attacker's goal. However, as the size and complexity of attack models and attack graphs usually greatly exceeds human ability to visualize, understand and analyze, a scheme is required to identify important portions of attack models and attack graphs. Mehta *et al.* proposed to rank states of an attack model by the probability of an adversary reaching a state by a sequence of exploiting individual vulnerabilities in a previous scheme. Important portions can hence be identified by ranks of states. However, Mehta *et al.*'s ranking scheme is based on the PageRank algorithm which models a web surfing scenario, but has not considered much on the dissimilarity between web surfing scenarios and computer system intrusion scenarios. In this paper, we extend Mehta *et al.*'s scheme by taking into consideration dissimilarity between web surfing scenarios and computer system intrusion scenarios. We experiment with the same network model used in Mehta *et al.*'s scheme and have the results compared. The experiments yielded promising results that demonstrated consistent ranks amongst varying parameters modelled by our ranking scheme.

1 Introduction

A large computer system often consists of multiple platforms, runs different software packages and has complex connections to other systems. Despite the best efforts by system designers and architects, there will still exist vulnerabilities resulting from bugs or design flaws allowing an adversary (attacker) to gain a level of access to systems or information not desired by the system owners. The act of taking advantage of an individual vulnerability is referred to as an "atomic attack" or an "exploit". A vulnerability is often exploited by a piece of software, a chunk of data, or sequence of commands, resulting in unintended or unanticipated behavior of the system such as gaining control of a computer system or

* This work is partially supported by Cooperative Research Center - Smart Internet Technology (CRC-SIT), Australia.

allowing privilege escalation. Although an exploit may have only insignificant impact on the system by itself, an adversary may be able to construct a system *intrusion* that combines several atomic attacks, each taking the adversary from one system state to another, until he reaches the final goal. Therefore, to evaluate the security level of a large computer system, an administrator must not only take into account the effects of exploiting each individual vulnerability, but also consider global intrusion scenario where an adversary combines several exploits to compromise the system.

There has been considerable amount of work on modelling multi-stage attacks by combination of individual vulnerabilities [7] [12] [5]. Recently, Sheyner *et al.* proposed using attack models and attack graphs to provide a global view of system security against exploiting the combination of vulnerabilities [16] [7]. An attack model is a graph that consists of a set of nodes and edges, where each node represents a reachable system state and each edge represents an atomic attack that takes the system from one state to another. An attack graph is a subgraph of the attack model and contains only nodes in the paths that eventually reach a state where the system is considered compromised. With attack models and graphs, a global view on multi-stage attacks by combination of individual vulnerabilities can be obtained by administrators to assist in the implementation of effective security measures.

However, as the size and complexity of attack models/graphs usually greatly exceeds human ability to visualize, understand and analyze, a scheme is required to identify important portions of attack models/graphs. An effective method is to rank states in attack models/graphs based on factors like the probability of an intruder reaching the state. Important portions of attack models/graphs can hence be identified by ranks of their states.

Mehta *et al.* [16] propose to rank states of an attack model by the probability of an adversary reaching a state by a sequence of atomic attacks. The ranking algorithm is based on the PageRank algorithm used by Google to measure importance of web pages on the World Wide Web. Given a system to be analyzed, first an attack model formally describing the system is constructed. Then the ranking algorithm is applied to rank states of the obtained attack model. Meanwhile, an attack graph is generated using the attack graph generation tool [10], and is "projected" on the ranked attack model to obtain a ranked attack graph. The ranked states of an attack graph provide various security analysis for the system, such as measuring security of the system, evaluating effectiveness of counter-measures, and identifying important portion for visual analysis of the system.

Despite the similarity between ranking web pages and ranking system states, differences not considered by Mehta's scheme exist between the two scenarios. The World Wide Web model adopted by PageRank assumes that a random surfer has universally equal probabilities of following one of the links in a current page to the next page, and correspondingly Mehta's ranking scheme assumes that an attacker has equal probability of remaining undetected at all states of an attack model. However, the likelihood of an attacker remaining undetected at a state so as to exploit a vulnerability that takes the system to another state could be considered to be influenced by the number of steps required to reach the state from the starting position. Consider a scenario where a network has implemented some sort of defense-in-depth as an example, the more steps an attack has taken, the more likely that he is discovered. Therefore, we assume the probability of an attacker remaining undetected at a state decreases with number of steps required to reach that state. The decreasing rate is not universal amongst all computer systems but determined by each system's intrusion detection ability, which affects state transitions in attack models and should be considered when ranking system states.

Moreover, the random transition model adopted by PageRank assumed that a WWW surfer navigates to the next page by selecting one of the available succeeding pages at random with equal probabilities. This assumption fits well with intrusions that use a brute force probe-scan approach. However, an adversary may exploit vulnerabilities based on metrics such as cost, age, evaluation on probability of success and being detected. In this case, vulnerabilities are not selected at random and different vulnerabilities have different probabilities of being exploited. The behavior of an adversary selectively exploiting vulnerabilities has considerable effect on his chance to reach the final goal, and therefore should be considered when ranking system states of attack models and graphs.

1.1 Our Contribution

In this paper, we propose a ranking scheme that addresses problems stated above. The proposed ranking scheme is adjusted from Mehta *et al.*'s scheme, but has the advantage of modelling variation in intrusion detection abilities amongst computer systems, and non-uniform distribution in probability that each vulnerability is exploited. First, in addition to modelling vulnerabilities in a system that could be exploited to have system states changed, our ranking scheme also models intrusion detection ability of computer systems defined as the system's effort to detect and prevent such state transitions by intruders exploiting vulnerabilities. Secondly, we provide an instantiation of the biasing idea in [1] by modelling adversaries' behavior in exploiting vulnerabilities probabilistically based on certain metrics as stated above but not by brute-force probing. With the proposed ranking scheme, evaluation on system intrusion detection ability or adversaries' ability in relation to probabilistically exploit vulnerabilities, when available from for example empirical data or log statistics, can be used to obtain more accurate ranks of computer system states modelled by attack models and attack graphs. The proposed scheme can also be applied to other areas such as network research or system design, e.g. determining minimum system intrusion detection strength required to protect against best effort by an adversary.

To evaluate the effectiveness of the proposed ranking scheme, we implemented a prototype of the scheme in Java. We experiment with the network example used by Mehta *et al.* [16] and have the results compared with their scheme. The experiments yielded promising results that demonstrated consistent ranks amongst varying parameters modelled by the proposed ranking scheme.

1.2 Related Work

Techniques for quantitative security measurement have been a strong focus in the research community [9] [11] [4] [5]. Dacier *et al.* [9] model a computer system as a *Privilege Graphs* exhibiting security vulnerabilities and convert it into a Markov chain corresponding to all possible successful attack scenarios. The Markov chain is then used to compute MTTF (mean time to failure) of the system, used as the quantitative measure of the security level of a system. Time and effort required by each type of attack is estimated from empirical and statistical data. Phillips *et al.* [5] present a framework for evaluating the most likely attack paths in the attack graph generated by an ad hoc algorithm. The framework requires attacker profiles and attack templates in order to compute the likelihood of each type of attack. Madan *et al* [4] proposed an approach to quantifying various security related attributes of a computer system such as system availability, MTTF, and probabilities of system failure. Quantification of security related attributes is by solving the Semi-Markov Process (SMP) model describing state transitions in the system. However, the proposed approach requires availability of a wide range of ad hoc model parameters, restricting the approach feasible only for systems of a small scale. Another related work presented in [11] provides a quantitative analysis of attacker behavior based on empirical data collected from intrusion experiments.

1.3 Paper Organisation

The rest of the paper is organized as follows. Section 2 provides a brief review on relevant background knowledge. The proposed ranking scheme is presented in Section 3. Section 4 provides implementation details and experimental results demonstrating effectiveness of the proposed scheme, and Section 5 concludes the paper.

2 Background and Preliminaries

2.1 Attack Models and Attack Graph

Sheyner *et al.* first formally defined the concept of *Attack Model* and *Attack Graph* [10]. An *Attack Model* is a formal description of security related attributes of the attacker, the defender and the modelled system using graph representation. Nodes represent the states of the system, such as the attacker's privilege level on individual system components. Transitions correspond to actions taken by the attacker which lead to a change in the state of the system. The starting state of the model denotes the state of the system where no damage has occurred and the attacker is looking for an entry point to enter the system. As an example, if we consider the case of a computer network attack model, a state represents the state of the attacker, the running services, access privileges, network connectivity and trust relations. The transitions correspond to actions of the attacker such as exploiting vulnerabilities to obtain elevated privileges on the computer system. Formally,

Definition 1. [10] *Let AP be a set of atomic propositions. An Attack Model is a finite automaton $M = (S, \tau, s_0, l)$, where S is a set of states in relation to a computer system, $\tau \subseteq S \times S$ is the transition relation, $s_0 \in S$ is the initial state, and $l : S \to 2^{AP}$ is a labelling of states with the set of propositions true in that state.*

The negation of an attacker's goal in relation to an attack model can be used as *security properties* that the system must satisfy in a secure state. An example of a security property in computer networks would be "the intruder cannot gain root access on the database server". States in an attack model where the *security properties* are not satisfied are called *error states*. Given an attack model and the attacker's goal, an *Attack Graph* is a subgraph of the attack model which contains only paths leading to one of the error states, and states on such paths. Formally,

Definition 2. [10] *Let AP be a set of atomic propositions. An Attack Graph is a finite automaton $G = (S, \tau, s_0, S_s, l)$, where S is a set of states in relation to a computer system, $\tau \subseteq S \times S$ is the transition relation, $s_0 \in S$ is the initial state, $S_s \subseteq S$ is the set of error states in relation to the security properties specified for the system, and $l : S \to 2^{AP}$ is a labelling of states with the set of propositions true in that state.*

Given an attack model and the associated security properties, model checking techniques can be used to generate attack graphs automatically [14].

2.2 Web Graph and PageRank

Web Graph and Notations. For a web model consisting of N nodes (pages), notations used in the our discussion are defined in Table 1. Consider the web graph shown in Figure 1 as an example. Node 1 has three out links (node 2, 3 and 4) and hence $h_1 = 3$. Node 4 is pointed to by two nodes (node 1 and 3) and hence $pa[4] = \{\text{node 1, node 3}\}$. Equation 1 represents the transition matrix W in relation to the web graph. It is noticeable that not all nodes have out links, and we refer to these nodes as *dangling nodes*. When a web surfer reaches at a web page that has no out links, he is often assumed to select a random page to continue surfing [13] [2]. To model this, a *dangling node* in a web graph is typically assumed to be pointing to all other nodes with equal probabilities.

$$\mathbf{W} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0 \\ \frac{1}{3}&0&0&0&0&0&0&0&0 \\ \frac{1}{3}&0&0&0&0&0&0&0&0 \\ \frac{1}{3}&0&\frac{1}{3}&0&0&0&0&0&0 \\ 0&1&0&\frac{1}{2}&0&0&0&0&0 \\ 0&0&0&0&\frac{1}{2}&0&0&0&0 \\ 0&0&0&0&\frac{1}{2}&0&1&0&0 \\ 0&0&\frac{1}{3}&\frac{1}{2}&0&0&0&0&0 \\ 0&0&\frac{1}{3}&0&0&0&0&0&0 \end{pmatrix} \tag{1}$$

Table 1. Web Model Notations

Notation	Meaning
h_j	Number of nodes (pages) pointed to by node (page) j
$pa[j]$	Set of nodes (pages) pointing to node (page) j.
d	Probability that a random surfer continues surfing by navigating to one of the pages linked by the current page, usually referred to as *damping factor*. Correspondingly, $1 - d$ represents the probability that a random surfer continues surfing and navigates to a random page
W	$W = \{w_{i,j}\}$ is a transition matrix such that $w_{i,j} = \frac{1}{h_j}$ if there is a link from node j to node i, otherwise $w_{i,j} = 0$. An important property of W is that $\forall j$, $\sum_{i=1}^{N} w_{i,j} = 1$.
Π_N	[1, . . ., 1]', i.e. transpose of the N-dimension unit vector

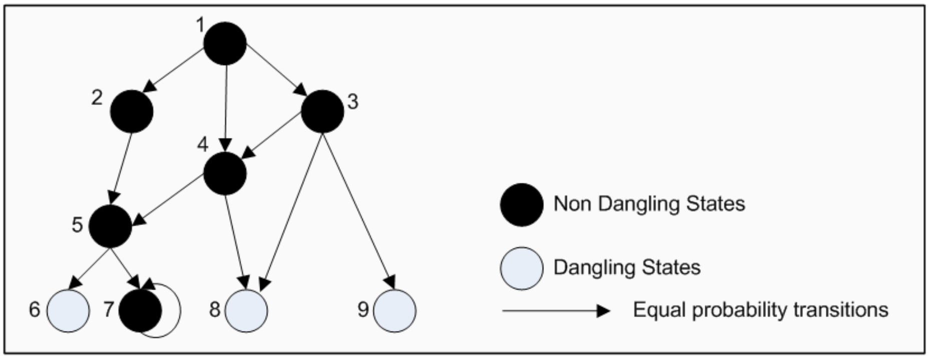


Fig. 1. An example of web graph

PageRank Algorithm. PageRank [13] is the algorithm used by Google to determine the relative importance of web pages on the World Wide Web. PageRank is based on the behavior model of a random surfer in a web graph. It assumes that a random surfer starts at a random page and keeps clicking on links and eventually gets bored and starts on another random page. To capture the notion that a random surfer might get bored and restart from another random page, a *damping factor* d is introduced, where $0 < d < 1$. The transition probability from a state is divided into two parts: d and 1 - d. The d mass is divided equally among the state's successors. Random transitions are added from that state to all other states with the residual probability 1 - d equally divided amongst them, modelling that if a random surfer arrives at a dangling page where no links are available, he is assumed to pick another page at random and continue surfing from that page. The computed rank of a page is the probability of a random surfer reaching that page. That is, consider a web graph with N pages linked to each other by hyperlinks, the PageRank x_p of page (node) p is defined as the probability of the random surfer reaching p, formally

$$x_p = d \sum_{q \in pa[p]} \frac{x_q}{h_q} + \frac{1-d}{N} \tag{2}$$

When stacking all the x_p into a vector $\mathbf{x}$, it can be represented as

$$\mathbf{x} = dW\mathbf{x} + \frac{1}{N}(1-d)\Pi_N \tag{3}$$

Using iterative expression, Equation 3 can be represented as

$$\mathbf{x}(t) = dW\mathbf{x}(t-1) + \frac{1}{N}(1-d)\Pi_N \tag{4}$$

The computation of PageRank can be considered a Markov Process, as can be seen from Equation 4. It has been proved that after multiple iterations, Equation 4 will reach a stationary state where each x_p represents the probability of the random surfer reaching page p [8].

2.3 Mehta et al's Ranking Scheme

Given an attack model $M = (S, \tau, s_0, l)$, the transition probability from each state is divided into d and 1-d, modelling respectively that an attacker is discovered and isolated from the system, or that the attacker remains undetected and proceeds to the next state with his intrusion. Similar to PageRank, the rank of a state in an attack model is defined to be the probability of the system being taken to that state by a sequence of exploits. The ranks of all states are computed using the method for computing PageRank described in Section 2.2. Breadth first search starting from the initial system state s_0 is then performed for each atomic attack in τ to construct the transition matrix W. The only adjustment from PageRank, where a transition from each state pointing to all other states with probability 1-d equally divided amongst all other states, is that a transition from each state pointing back to the initial state with probability 1-d is added to model the situation where an attacker is discovered and has to restart the intrusion from the initial state.

3 Modelling Adversary and Intrusion Detection Capability in Ranking Attack Models

Recall the discussion in Section 1. Unlike the web graph model adopted by PageRank where the probability that a random surfer follows a link to the next page is state independent, the likelihood of an attacker remaining undetected at a state so as to exploit a vulnerability that takes the system to another state could be considered to be influenced by the number of steps required to reach the state from the starting position. Therefore we assume the probability of an attacker remaining undetected at a state decreases with number of steps required to reach that state. The decreasing rate is not universal amongst all computer systems but system specific as determined by each system's intrusion detection ability. Another important dissimilarity between a web surfing scenario and a system intrusion scenario is that exploits taking a computer system from one state to another may be "selected" not at random but based on the adversary's evaluation on metrics such as cost, effort, probability of success and being detected, whereas links taking a web surfer to the next page is always selected at random with equal probabilities. These factors affect an adversary's chance to reach his final goal, and therefore should be considered when ranking states of attack models and graphs.

In this section, we propose an adversary aware and intrusion detection aware ranking scheme that addresses problems stated above. Being adversary aware, the proposed scheme considers how an adversary selectively exploiting vulnerabilities affect his chance to compromise the system. Being intrusion detection aware, the proposed scheme considers system intrusion detection ability and how it affects an adversary's chance to reach his final goal.

3.1 Web Graph Adjustment

The transition model of web graph needs to be adjusted to provide a more accurate simulation of computer system state transitions in relation to system intrusion scenario. As in Mehta *et al.*'s ranking scheme, we add a transition from each state pointing back to the initial state with probability 1-d, modelling the situation where an intrusion is detected and needs to be restarted from initial state. Furthermore, our ranking scheme differs from Mehta *et al.*'s scheme in that

1. We assume that the probability of an attacker remaining undetected at a state decreases with the number of steps required to reach that state, which in an attack model can be represented as length of the intrusion path to reach that state. The decreasing rate is determined by each system's intrusion detection ability. In general, well-protected systems such as systems implementing "defense-in-depth" have better ability to detect intrusions at earlier stages and can be simulated with greater decreasing rates. As it is difficult to predict the actual intrusion path, we simplify the situation by assuming that at each state s_j the probability of an attacker remaining undetected decreases at a rate proportional to $l(s_0, s_j)$, length of the shortest path between state s_j and initial state s_0. That is, the probability of an attacker remaining undetected at state s_j exponentially decays with length of the shortest path from s_0 to s_j. Consequently, transition probability from each state s_j is divided into d_j and 1-d_j representing respectively the situation where an attacker remains undetected and is able to take the system to another state, or where the attacker is discovered and has to restart the intrusion. d_j is the value of d exponentially decaying with $l(s_0, s_j)$ where d is the usual *damping factor*.

2. In a system intrusion scenario, it is more likely that an adversary has the ability to prioritize and exploit "promising" vulnerabilities based on his past experience and knowledge, other than probing the target network with brute-force attack. This is modelled in our ranking scheme by assigning a separate probability to each type of exploit. We divide each d_j among state s_j's successors according to the probability that each type of exploit is selected to take s_j to one of its successors. The probability distribution can be obtained from empirical data or other sources [12]. By doing so, that the adversary probes the system with brute-force attack can be modelled by assigning equal probabilities to all exploits. Similarly, intrusions by an experienced attacker who exploits vulnerabilities selectively to maximize his chance of success can be modelled by assigning higher probabilities to critical exploits.
3. We add a transition from each dangling state pointing back to the initial state s_0 with probability 1, modelling the situation that an adversary has come to a state where he cannot proceed with the intrusion and has to restart from initial state.

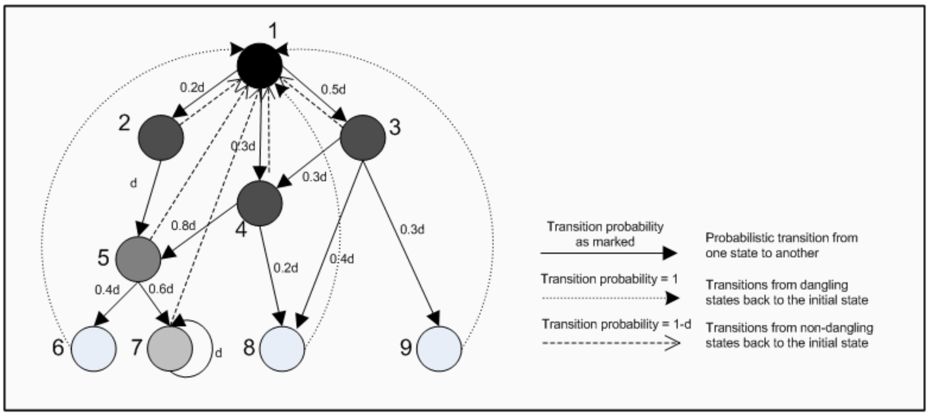

1
2
3
4
5
6
7
8
9
0.2d
0.5d
0.3d
d
0.3d
0.8d
0.3d
0.4d
0.2d
0.4d
0.6d
d
Transition probability as marked
Probabilistic transition from one state to another
Transition probability = 1
Transitions from dangling states back to the initial state
Transition probability = 1-d
Transitions from non-dangling states back to the initial state

Fig. 2. Transitions in attack models

Consider the graph illustrated in Figure 1 as an example. Assume we have some empirical data that enables us to estimate that whenever the system is in s_1, on average it will take the transition to s_2, s_3 and s_4 2, 5 and 3 times, respectively, out of ten. We can then place probabilities 0.2, 0.5 and 0.3 on these transitions. Similarly, assume that the empirical data enables us to place probability 0.3, 0.4 and 0.3 to the transitions taking s_3 to s_4, s_8 and s_9 respectively, probability 0.8 and 0.2 to the transitions taking s_4 to s_5 and s_8 respectively, and probability probability 0.4 and 0.6 to the transitions taking s_5 to s_6 and s_7 respectively. Figure 2 illustrates the graph with adjusted transition model from web graphs, which is a more accurate simulation of computer system state transition in an intrusion scenario. The intensity of color for each state s_j visualize the probability d_j that an intrusion is not detected at that state.

3.2 Transition Matrix Construction

To rank an attack model $M = (S, \tau, s_0, l)$, we need to construct the *transition matrix* $\overline{W} = \overline{w}_{ij}$, the matrix representation of state transitions in an attack model, where $\overline{w}_{ij}$ is the probability of the system being taken to state s_j from state s_i. Let $\tau(s_j \rightarrow s_i)$ denote the proportion between the number of exploits that take the system from s_i to s_j and the total number of exploits applicable to s_i and $l(s_i, s_j)$ denote the length of the shortest path between s_i and s_j, a concrete algorithm for constructing $\overline{W}$ is presented in Algorithm 1. Depth-first-search or model checker such as NuSMV [1] is first used to construct the $N \times N$ *adjacency matrix AM* where N is the number of reachable states in M, such that $AM[i, j] = 1$ if state s_j is one of the successor states of state s_i, otherwise $AM[i, j] = 0$. Then the *transition matrix* $\overline{W}$ is constructed following the above stated adjustment to state transitions in web graphs.

Reconsider the web graph illustrated in Figure 1 as a example. We now construct the *transition matrix* $\overline{W}$ according to the adjustment illustrated in Figure 2 using Algorithm 1. The generated $\overline{W}$ is shown in Equation 5 where each $d_i = d \times e^{-\lambda l(s_1, s_i)}$, d being the usual damping factor used in PageRank.

$$\overline{\mathbf{W}} = \begin{pmatrix} 0 & 1-d_2 & 1-d_3 & 1-d_4 & 1-d_5 & 1 & 1-d_7 & 1 & 1 \\ 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0 & 0.3d_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0.8d_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4d_5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.6d_5 & 0 & d_7 & 0 & 0 \\ 0 & 0 & 0.4d_3 & 0.2d_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.3d_3 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5}$$

3.3 Ranking Attack Models

Following Mehta *et al.*'s definition, we define the rank for each state s_j in an attack model as the probability that s_j is reached from the initial state s_0. This can be recursively represented as

$$x_p = \sum_{q \in pa[p]} \overline{w}_{qp} \times x_q \tag{6}$$

When stacking all x_p into one vector $\mathbf{x}$, Equation 6 can be represented as

$$\mathbf{x} = \overline{W}\mathbf{x} \tag{7}$$

$\mathbf{x}$ in Equation 7 can be computed by multiple iterations through the following equation until a stationary state is reached.

$$\mathbf{x(t)} = \overline{W}\mathbf{x(t\text{-}1)} \tag{8}$$

Algorithm 1: Generate$\overline{W}(M)$

```
/* The function generates the adjusted transition matrix $\overline{W}$ from the
   given attack model $M$.                                                  */
/* Input: $M = (S, \tau, s_1, L)$: the attack model where $s_1$ is the initial
   state of $M$.                                                            */
/* Output: $\overline{W} = \overline{w}_{ij}$, where $\overline{w}_{ij}$ represents the probability of an
   adversary exploiting the vulnerability that takes the system from
   state $s_j$ to state $s_i$.                                              */
begin
    $AM$ = Construct_Adjacency_Matrix_From_Model($M$)
    /* Set the probabilities of transitions to and from the initial
       state                                                                */
    for $i = 1$ to $N$ do
        if $AM[1, i] = 1$ then
            $\overline{w}_{i1} = \tau(s_1 \to s_i)$
        else
            $\overline{w}_{i1} = 0$
        if $\forall j, AM[i, j] = 0$ then
            $\overline{w}_{1i} = 1$
        else
            $\overline{w}_{1i} = 1 - d \times e^{-\lambda l(s_1, s_i)}$
    /* Set the probabilities of transitions to and from other states
       */
    for $i = 2$ to $N$ do
        for $j = 2$ to $N$ do
            if $AM[j, i] = 0$ then
                $\overline{w}_{ij} = 0$
            else
                $\overline{w}_{ij} = d \times \tau(s_j \to s_i) \times e^{-\lambda l(s_1, s_j)}$
end
```

If Equation 8 reaches a stationary state, i.e. $x(t) = x(t-1)$, after a long run of computation, all states in attack graph can be ranked. However, Equation 8 may or may not reach a stationary state after a long run of computation. Moreover, the result after multiple iterations may not be interesting (for example, the stationary state $\lim_{t\to\infty} \mathbf{x(t)}$ may be a vector of all 0s). A detailed proof of **Theorem 1** is provided in Appendix A to justify that Equation 8 constructed as above can always reach a non-trivial stationary state after multiple iterations.

Theorem 1. *Equation 8 converges at a non-trivial vector x^* where $\sum_i x_i^* = 1$ after multiple iterations.*

Given an attack model and empirical data that enables us to evaluate probabilities of different vulnerabilities being exploited, we first construct the *transition* *matrix* $\overline{W}$ as presented in Algorithm 1. We then assign random initial value to the rank of each state, and run Equation 8 for multiple iterations until it reaches the stationary state, guaranteed to exist by **Theorem 1**.

4 Implementation and Experiments

To evaluate the effectiveness of the proposed ranking scheme, we developed a toolkit in Java that ranks attack models with the proposed scheme. We ran the toolkit on the network example used by Mehta *et al* [16] and have the results compared with their ranking scheme. In this section, we first provide implementation details of the toolkit, then present the network model and the experimental results.

4.1 Implementation

The implementation toolkit is developed in Java but relies on NuSMV [1] for model checking functionalities, such as generating the complete set of reachable states given an initial state and the set of allowed state transitions. We made a minor modification to the source code of NuSMV (see below) to achieve interaction with our Java-based implementation toolkit. In the following, we provide details on the architecture of our implementation toolkit and its interaction with the modified NuSMV.

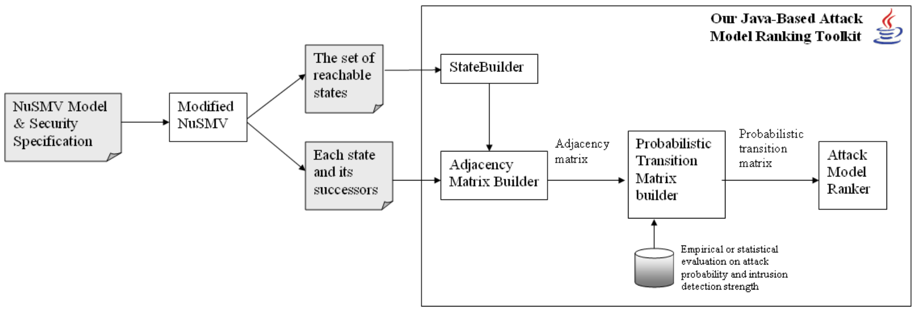

Our Java-Based Attack Model Ranking Toolkit
NuSMV Model & Security Specification
Modified NuSMV
The set of reachable states
Each state and its successors
StateBuilder
Adjacency Matrix Builder
Adjacency matrix
Probabilistic Transition Matrix builder
Probabilistic transition matrix
Attack Model Ranker
Empirical or statistical evaluation on attack probability and intrusion detection strength

Fig. 3. Toolkit Architecture

Toolkit Architecture. The architecture of our Java-based attack model ranking toolkit is illustrated in Figure 3. A network model along with the security specification written in NuSMV modelling language are fed to NuSMV. NuSMV then generates the complete set of reachable states S in the given model. We also modified NuSMV so that for each state $s \in S$ it generates the set of successors. The results generated as above are then saved as files, and feeded to the implementation toolkit to construct the adjacency matrix for states in the model. Combining the adjacency matrix, the empirical evaluation on the probability that each type of vulnerability is exploited, and the evaluation on the system's intrusion detection ability, the implementation toolkit generates the *transition matrix* $\overline{W}$ and ranks the states in the attack model as described in Section 3.

Toolkit Components

State Builder. With the NuSMV command *print_reachable_states -v*, we generate the set of reachable states from the specified system initial state which are saved to a text file. The State Builder then reads the set of reachable states into Java-specific representation from the generated text file.

Adjacency Matrix Builder. We modified NuSMV such that it generates and saves into a text file the successor states of a given state with the *-st* command line option. Iteratively using the *-st* option for each reachable state, the Adjacency Matrix Builder generates an $N \times N$ adjacency matrix AM where N is the number of states in the attack model such that $AM[i, j] = 1$ if state j is one of the successor states of state i, otherwise $AM[i, j] = 0$.

Transition Matrix Builder. Combining the adjacency matrix, the empirical evaluation on the probability that each type of vulnerability is exploited, and the evaluation on the system's intrusion detection ability, the Transition Matrix Builder follows Algorithm 1 to generate the *transition matrix* $\overline{W}$.

Attack Model Ranker. Given the *transition matrix* $\overline{W}$, the Attack Model Ranker computes the ranks for all reachable states in the attack model using Equation 8 iteratively until the stationary is reached. A non-trivial stationary state is guaranteed to exist after multiple iterations by **Theorem 1**.

4.2 The Network Model for Experiments

The network model used for our experiments is illustrated in Figure 4. There are two target hosts ip_1 and ip_2, and a firewall separating them from the rest of the Internet. As shown each host is running two of three possible services (ftp, sshd, database). We model the same 4 types of atomic attacks summarized in Table 2 as in [15] [16] for comparable results. A detailed explanation of each attack follows.

The intruder launches his attack starting from an external machine ip_a that lies outside the firewall. His eventual goal is to gain access to the database. For that, he needs root access on the database server ip_2.

Table 2. Atomic Attacks Modelled in the Sample Network

Attack	**Vulnerability Exploited**
▶ sshd buffer overflow	Some versions of ssh are vulnerable to buffer overflow
▶ ftp.rhosts	Exploiting the vulnerability resulting from a writable ftp home directory
▶ remote login	Remote trust relation between machines
▶ local buffer overflow	Some `setuid root` executables are vulnerable to buffer overflow

We construct a finite state model of the network such that each state represents the system state including trust relation, connectivity, and adversary privilege on each machine, and each state transition corresponds to a single atomic attack which takes the system from one state to another.

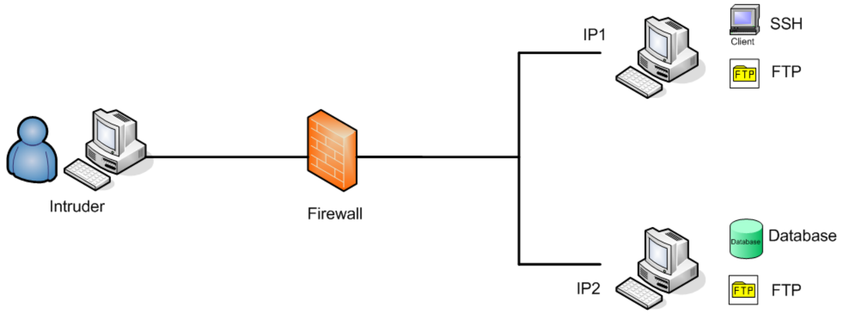


Fig. 4. Network

Connectivity and Trust Relation. Connectivity models the connection between two machines. We denote the connectivity by a binary relation $\mathit{Reachable} \subseteq \mathit{Host} \times \mathit{Host}$, where $\mathit{Reachable}(h_1, h_2) = 1$ if h_1 can connect to h_2, otherwise $\mathit{Reachable}(h_1, h_2) = 0$, i.e. either there is no physical link between h_1 and h_2, or the link is blocked by the firewall. Assuming the firewall policy is that the ftp server (ip_1) is publicly accessible while the database server (ip_2) can only be accessed internally, the connectivity relation is shown in Table 3. Similarly, we denote trust relation between machines by a binary relation $\mathit{Trust} \subseteq \mathit{Host} \times \mathit{Host}$, where $\mathit{Trust}(h_1, h_2) = 1$ if a user on h_1 can login to h_2 remotely without specifying a password, $\mathit{Trust}(h_1, h_2) = 0$ otherwise. The trust relation is summarized in Table 4.

The Adversary and Privilege. Privileges are $\{\mathit{none}, \mathit{user}, \mathit{root}\}$. There is an ordering of privileges: $\mathit{none} < \mathit{user} < \mathit{root}$. The adversary has root on ip_a and no privileges on other machines initially. We use the function $\mathit{plvl}_A(H) : \mathit{Hosts} \rightarrow \{\mathit{none}, \mathit{user}, \mathit{root}\}$ to denote the level of privilege that intruder A has on machine H.

Vulnerabilities and Atomic Attacks. We model the same 4 types of attacks as in [15] [16], each taking the modelled network from one state to another as described by the "effect" section of the attack. An attack is only applicable when both the network precondition and intruder precondition are satisfied. Throughout the following description, we denote source and target machine by S and T. To simplify the notations, we use ssh_H, ftp_H and local_H to denote the presence of a vulnerability by running ssh service, ftp service and a `setuid root` executable respectively on host H.

Table 3. Connectivity

Reachable	ip_a	ip_1	ip_2
ip_a	1	1	0
ip_1	1	1	1
ip_2	1	1	1

Table 4. Trust Relation

Trust	ip_a	ip_1	ip_2
ip_a	1	0	0
ip_1	0	1	1
ip_2	0	1	1

1. sshd buffer overflow: Some versions of ssh services are vulnerable to a buffer overflow attack that allows an intruder to obtain a root shell on the target machine. Formally,

```
attack sshd-buffer-overflow is
    intruder preconditions
        [User-level privileges on host S]
        $plvl_A(S) \geq user$
    network preconditions
        [Host T is running a vulnerable version of ssh service]
        $ssh_T$
        [Host T is reachable from S]
        $Reachable(S, T) = 1$
    intruder effects
        [Root-level privileges on host T]
        $plvl_A(T) = root$
end
```

2. ftp .rhosts: With a writable home directory and an executable command shell assigned to anonymous ftp users, an intruder can modify the .rhosts file in the ftp home directory, so as to create a remote login trust relationship between his machine and the target machine. Formally,

```
attack ftp-rhosts is
    intruder preconditions
        [User-level privileges on host S]
        $plvl_A(S) \geq user$
    network preconditions
        [Host T is running a ftp service in a writable directory,
        which gives a user shell to ftp users]
        $ftp_T$
        [Host T is reachable from S]
        $Reachable(S, T) = 1$
    network effects
        [Trust relation between the intruder's machine and the target]
        $Trust(S, T) = 1$
end
```

3. remote login: Using an existing remote login trust relationship between two machines, the intruder can login from his machine to the target and obtain a user shell without supplying a password. Although remote login is usually considered a legitimate operation by regular users, it is however an atomic attack from an intruder's viewpoint. Formally,

```
attack remote-login is
    intruder preconditions
        [User-level privileges on host S]
        $plvl_A(S) \geq user$
    network preconditions
        [Host T trusts S]
        $Trust(S, T) = 1$
        [Host T is reachable from S]
        $Reachable(S, T) = 1$
    intruder effects
        [User-level privileges on host T]
        $plvl_A(T) = user$
end
```

4. local buffer overflow: The attacker exploits a buffer overflow vulnerability in a `setuid root` executable to gain root access. Formally,

```
attack local-buffer-overflow is
    intruder preconditions
        [User-level privileges on host T]
        $plvl_A(T) \geq user$
    network preconditions
        [Host T runs a vulnerable version of a setuid root executable]
        $local_T$
    intruder effects
        [Root-level privileges on host T]
        $plvl_A(T) = root$
end
```

4.3 Experimental Results Analysis and Evaluation

Let the security property be "intruder cannot gain root access on ip_2". We ran our attack model ranking toolkit presented in Section 4.1, and visualized the results with the *graphViz* package [3]. Figure 5 illustrates the result obtained as such. For each state, the intensity of color is proportional to the rank of that state. Any path in the graph from the root node to a leaf node represents a sequence of exploits with which the intruder can achieve his final goal. It can be seen that *local buffer overflow* and *remote login* are critical exploits as each path from the root node to a leaf node has exploited them at least once.

After fixing either *local buffer overflow* or *remote login*, NuSMV asserts security property "intruder cannot gain root access on ip_2" to be true. On the other hand, *ftp.rhost* and *ssh buffer over flow* are non-critical exploits as an intruder can still reach his final goal without either of them.

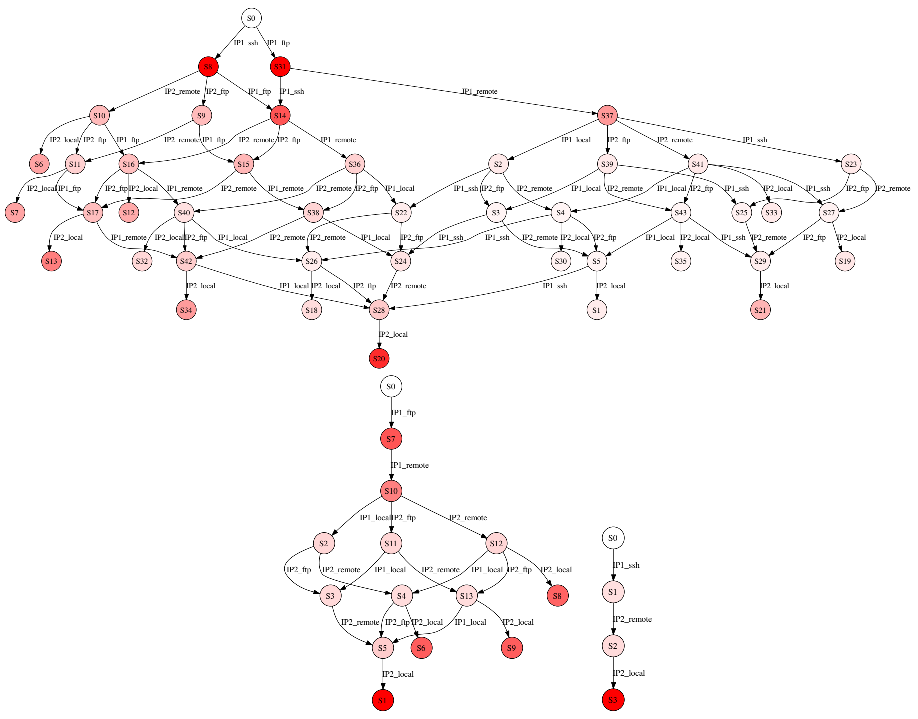


Fig. 5. Comparison of Ranked Attack Models. (a) The complete ranked attack model (b) Attack Model after fixing up the SSH vulnerability (c) Attack Model after fixing FTP vulnerability.

To investigate how an attacker selectively exploiting vulnerabilities affects his chance of compromising the system, we vary the probability assigned to each type of exploit and have other exploits divide the remaining probability equally. Changes to the ranks of states resulting from varying the probabilities of the exploits reflects how an attacker selectively exploiting vulnerabilities affects his chance to compromise the system. We also set the rate by which probability of an intrusion remaining undetected decays with the shortest path to 0, so that changes to the rank of a state are the result only of varying the probabilities of the exploits. The experimental result is plotted in Figure 6 where the Y-axis represents the total rank of error states, i.e. the probability of an adversary reaching his goal. It can be seen that the total rank of error states increases as the attacker prioritize critical exploits *local buffer overflow* and *remote login*, modelled by assigning higher probabilities to the two attacks. Similarly, the adversary's chance to succeed decreases as he prioritize non-critical exploits *ftp.rhosts* and *sshd buffer overflow*. In general, our scheme produces a higher rank when the attacker prioritize critical exploits and hence has better chance to succeed. The rank produced by our scheme joins the rank by Mehta's scheme at the equal probability point, i.e. where all exploits are assigned equal probabilities.

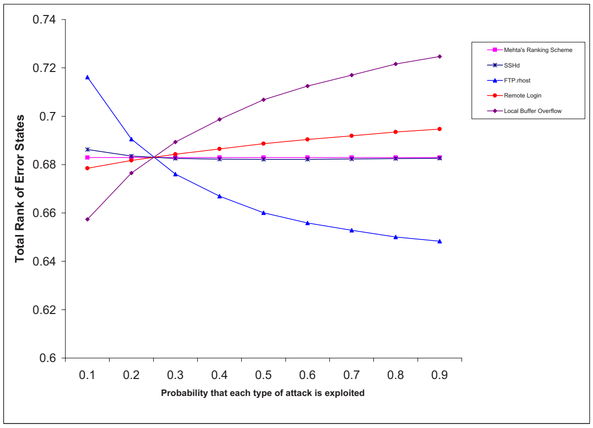


Fig. 6. Rank varies with attack probabilities

To investigate the effect that probability of an intrusion remaining undetected decays at a rate proportional to length of the shortest path from initial state, we vary the decaying rate λ while assigning equal probabilities to all exploits. The experimental result is plotted in Figure 7. It can be seen that the total rank of error states increases as the decaying rate decreases. This corresponds to the fact that an attacker has less chance of success on well-protected systems such as systems implementing "defense-in-depth" which at each step of the intrusion and thus on the whole has a higher probability of being able to discover and thwart the intrusion. The ranks produced by our ranking scheme consistently remain lower than the rank by Mehta's scheme, resulting from the decaying of probability that an adversary remains undetected and is able to proceed.

Figure 8 plots the experimental result by the overall effect of various decaying rates and varying probability assigned to each type of exploit (still other exploits divide remaining probability equally). It can be seen that ranking of system states is dominated by decaying of probability that intrusion remains undetected. Variation in probability assigned to each type of exploit only affects ranking of states to a minor extent. It can also be see that, the greater the decaying rate is, the less variation in probability assigned to each type of exploit affects ranking of states. The result reveals that deployment of well-protected system offsets experienced intruder's strategy in selectively exploiting vulnerabilities to maximize his chance of success, lowering his chance of success to no more than that of brute-force type of attack.

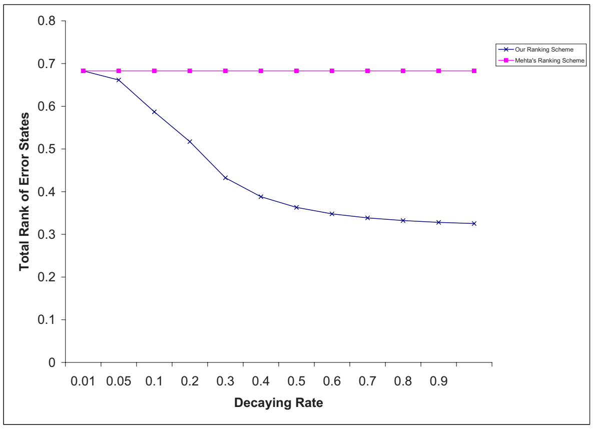


Fig. 7. Rank varies with decaying rate

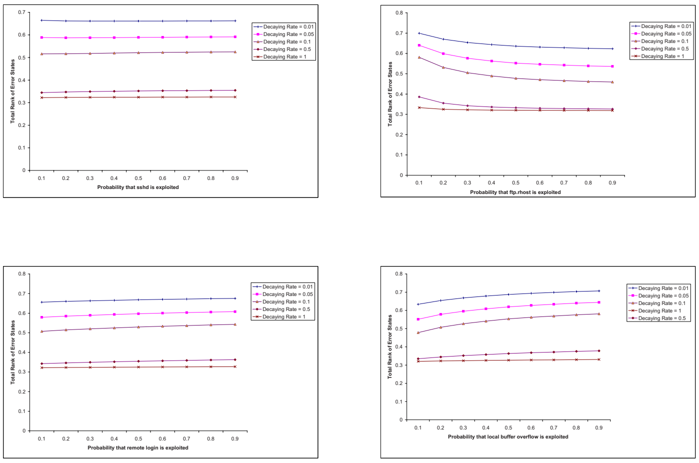


Fig. 8. Rank varies with decaying rate and attack probabilities

The experimental results and analysis presented above demonstrates the advantage and application of the proposed ranking scheme. Firstly, it considers the effect on ranking of system states by an intruder selectively exploiting vulnerabilities to maximize his chance of success. Secondly, it is able to model the effect on ranking of system states by system intrusion detection ability that aims at thwarting exploits of vulnerabilities that take the system to another state where the intruder gains an elevated privilege. Therefore, the proposed ranking scheme can rank attack models more accurately, and provide more realistic evaluation on the probability that a system is in a compromised state.

Intuitively, the probability of a system being in a compromised state increases with the probability that an intruder is able to prioritize critical exploits, and with weakening system intrusion detection ability; however, our ranking scheme provides a quantitative measure for the increase. The proposed scheme can also assist network researchers and architects in network design and analysis, e.g. determining the minimum intrusion detection strength required to thwart the best effort in selectively exploiting vulnerabilities by intruders.

5 Conclusion

As the size and complexity of attack models/graphs usually greatly exceed human ability to visualize, understand and analyze, ranking of states is often required to identify important portions of attack models/graphs. Mehta *et al* proposed a ranking scheme based on the PageRank algorithm used by Google to measure importance of web pages on World Wide Web. We extend their scheme by modelling an attacker selectively exploiting vulnerabilities to maximize his chance of compromising the system, and intrusion detection ability of computer systems detecting and preventing attackers to exploit system vulnerabilities. With the proposed ranking scheme, evaluation on system intrusion detection ability or attackers' ability in relation to probabilistically exploit vulnerabilities, when available from for example empirical data or log statistics, can be used to obtain more accurate ranks of computer system states modelled by attack models and attack graphs.

References

1. NuSMV: a new symbolic model checker. http://nusmv.irst.itc.it/.
2. A. Y. NG, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *Proceedings of International Conference on Research and Development in Information Retrieval (SIGIR 2001)*, New York, 2001. ACM.
3. AT&T Research. http://www.graphviz.org/.
4. B. B. Madan, K. G. Popstojanova, K. Vaidyanathan, and K. S. Trivedi. A method for modeling and quantifying the security attributes of intrusion tolerant systems. In *Dependable Systems and Networks-Performance and Dependability Symposium*, number 167-186, 2004.
5. C.A. Phillips and L. P. Swiler. A graph based system for network vulnerability analysis. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2000.
6. G. H. Golub and V. Loan. *Matrix computation.* The Johns Hopkins University Press, 1993.
7. J. Dawkins and J. Hale. A systematic approach to multi-stage network attack analysis. In *Proceedings of the Second IEEE International Information Assurance Workshop*, 2004.
8. M. Bianchini, M. Gori, and F. Scarsell. Inside PageRank. *ACM Transactions on Internet Technology*, 5(1):92–118, Feb 2001.
9. M. Dacier, Y. Deswarte, and M. Kaaniche. Quantitative assessment of operational security: Models and tools. Technical Report 96493, LAAS, May 1996.

10. O. Sheyner, J. Haines S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002.
11. Y. Deswarte R. Ortalo and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. *Software Engineering*, 25(5):633–650, 1999.
12. R. Ortalo, Y. Deshwarte, and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. In *IEEE Transactions on Software Engineering*, pages 71–79, 1999.
13. S. Brin, L. Page, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical Report 1999-66, Standford University, 1999.
14. S. Jha and J. Wing. Survivability analysis of networked systems. In *23rd International Conference on Software Engineering(ICSE01)*, number 03-07, 2001.
15. S. Jha, O. Sheyner, J. Wing. Two Formal Analysis of Attack Graphs. In *15th IEEE Computer Security Foundations Workshop (CSFW'02)*, page 49. IEEE, 2002.
16. V. Mehta, C. Bartzis, H. Zhu, E. Clarke and J. Wing. Ranking Attack Graphs. In *Proceeding of the 9th International Symposium On Recent Advances In Intrusion Detection*, Hamburg, Germany, September 2006. Springer.

A Proof of Theorem 1

Lemma 1. *Each column of the transition matrix $\overline{W}$ constructed by Algorithm 1 sums to 1, i.e.* $\sum_{i=1}^{N} \overline{w}_{i,j} = 1$.

Proof of the above lemma follows directly the way by which $\overline{W}$ is constructed.

Theorem 1. Equation 8 converges at a non-trivial vector x^* where $\sum_i x_i^* = 1$ after multiple iterations.

Proof: Consider a linear transformation of x_p defined in Equation 2. Let

$$x'_p = c_1 \times x_p + c_2 = c_1 \times \sum_{q \in pa[p]} x_q \times \overline{w}_{pq} + c_2 \tag{9}$$

where $c_2 = \frac{1-c_1}{N}$. Stacking all x'_p into one vector x' and using iterative expression, Equation 9 is represented as

$$\mathbf{x(t)'} = c_1 \overline{W} \mathbf{x(t\text{-}1)'} + c_2 \Pi_N \tag{10}$$

A well-known theory states that the condition that $MX(K+1) = NX(K) + b$ converges at $(M-N)^{-1}b$ is $\rho(M^{-1}N) < 1$ [6].

Here we have $M = I$ and $N = c_1\overline{W}$. Therefore $\rho(M^{-1}N) = \rho(c_1\overline{W}) = c_1\rho(\overline{W})$. Assume x is an eigenvector of $\overline{W}$ and λ is the associated eigenvalue, then $\overline{W}x = \lambda x$, i.e. $\forall i$, $\sum_{j=1}^{N} \overline{w}_{i,j} x_i = \lambda x_i$. Extracting the common factor x_i, this can be written as $x_i(\sum_{j=1}^{N} \overline{w}_{i,j} x_i - \lambda) = 0$. As x is an eigenvector, there exist non-zero x_i. Therefore, $\lambda = \sum_{j=1}^{N} \overline{w}_{i,j} x_i$. Following lemma 1, $\sum_{i=1}^{N} \overline{w}_{i,j} = 1$, we know that $\lambda = \sum_{j=1}^{N} \overline{w}_{i,j} x_i = 1$. Therefore, $\rho(\overline{W}) = 1$. On the other hand,

$0 < c_1 < 1$. As a result, $\rho(M^{-1}N) = c_1\rho(\overline{W}) < 1$, and hence Equation 10 converges at a stationary state $\lim_{t\to\infty} \mathbf{x(t)'}$.

We then prove by induction on t that the stationary state $\|\lim_{t\to\infty} \mathbf{x(t)'}\|_1$ of Equation 10 is a unit vector, i.e. $\|\lim_{t\to\infty} \mathbf{x(t)'}\|_1 = 1$.

1. For $t = 0$, Let $\mathbf{x(0)'} = \frac{1}{N}\mathit{\Pi}_N$; hence $\|\mathbf{x(0)'}\|_1 = 1$.
2. Let $t > 0$ and assume by induction that $\|\mathbf{x(t)'}\|_1 = 1$. Then, based on the definition of stochastic matrices,

$$\begin{aligned}\|\mathbf{x(t+1)'}\| = \mathit{\Pi}'_N\mathbf{x(t+1)'} &= c_1\mathit{\Pi}'_N W\mathbf{x(t)'} + c_2\mathit{\Pi}'_N\mathit{\Pi}_N \\ &= c_1\mathit{\Pi}'_N\mathbf{x(t)'} + (1 - c_1) = 1 \end{aligned} \tag{11}$$

   We hence proved that with the initial *unit vector* $\mathbf{x(0)'} = \frac{1}{N}\mathit{\Pi}_N$, $\|\lim_{t\to\infty} \mathbf{x(t)'}\|_1 = 1$. As stationary solution of Equation 10 is independent of the initial value $\mathbf{x(0)'}$ [6], it can be concluded immediately that $\|\lim_{t\to\infty} \mathbf{x(t)'}\|_1 = 1$ with any initial vector $\mathbf{x(0)'}$. Note that it can be seen from Equation 9 that $x'_p > 0$; hence $\|\mathbf{x(t)'}\|_1 = \sum_{p=1}^{N} x'_p = 1$

The stationary state $\mathbf{x(t)}$ of Equation 2 can be retrieved from $\mathbf{x(t)'}$ with linear conversion $\mathbf{x(t)} = \frac{(\mathbf{x(t)'} - c_2)}{c_1}$. $\mathbf{x(t)}$ is not trivial, because

$$\sum_{p=1}^{N} x_p = \sum_{p=1}^{N} \frac{x'_p - c_2}{c_1} = \frac{\sum_{p=1}^{N} x'_p - N \times c_2}{c_1} = \frac{1 - N \times c_2}{c_1} = 1 \tag{12}$$

That is, the stationary state $\mathbf{x(t)}$ is a unit vector. □

Analyzing an Electronic Cash Protocol Using Applied Pi Calculus[⋆]

Zhengqin Luo[1], Xiaojuan Cai[1], Jun Pang[2], and Yuxin Deng[1]

[1] BASICS, Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
{martyluo,cxj,yuxindeng}@sjtu.edu.cn
[2] Carl von Ossietzky Universität Oldenburg
Department für Informatik, 26111 Oldenburg, Germany
jun.pang@informatik.uni-oldenburg.de

Abstract. *Untraceability* and *unreuseability* are essential security properties for electronic cash protocols. Many protocols have been proposed to meet these two properties. However, most of them have not been formally proved to be untraceable and unreuseable. In this paper we propose to use the applied pi calculus as a framework for describing and analyzing electronic cash protocols, and we analyze Ferguson's electronic cash protocol as a case study. We believe that this approach is suitable for many different electronic cash protocols.

1 Introduction

Security protocols for on-line payments play an important role in today's electronic commerce. These protocols can be applied in a myriad of circumstances, from real time money transfer to selling soccer match tickets on the Internet. However, when transactions are monitored, some private information of customers, which should be kept secret, is recorded too. In order to protect users' privacy, researchers have proposed a set of untraceable electronic cash protocols [14,19,11,12,29]. Similar to physical cash, electronic cash is also portable, recognizable, transferable, and untraceable (anonymous).

As exemplified in Figure 1, an electronic cash protocol usually consists of three sub-protocols — the *withdraw* protocol, the *payment* protocol, and the *deposit* protocol. It also has three types of principals — a *bank*, a *payer*, and a *shop*. A typical flow of the protocol is given as follows:

1. By executing a withdraw stage protocol with the bank over an authenticated channel, the payer can obtain electronic coins issued by the bank;
2. The payer spends these coins in the shop, by performing the payment stage protocol, in which the shop can be convinced that these coins are not faked;

⋆ The work is supported by The National Grand Fundamental Research 973 Program of China (2003CB317005), and The National Nature Science Foundation of China (60473006) and (60573002).

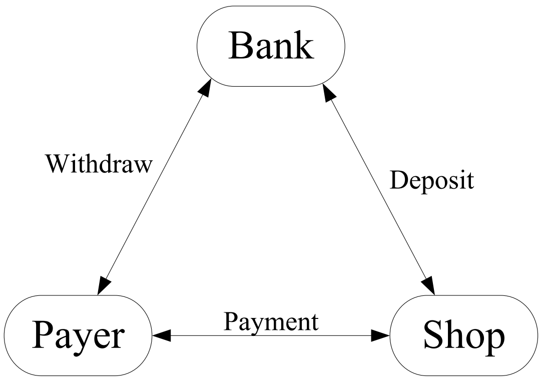


Fig. 1. Typical electronic cash protocol

3. The shop can initiate the deposit stage protocol with the bank, to verify and deposit these coins into its account.

In an on-line electronic cash protocol, the bank is required to stay in a standby condition to verify the coins for the shop in the payment stage protocol, while in an off-line one, its payment stage protocol does not require interactions between the shop and the bank. That is to say, the shop verifies the validity of the electronic coins without the bank's assistance. In this paper, we only consider off-line electronic cash protocols because they reduce the complexity of the bank systems. However this advantage also brings the danger of the payer's using the same coin more than once. So two main security properties are required for any off-line electronic cash protocol:

– *Anonymity (Untraceability).* The bank should not be able to determine if a certain payment is made by a particular payer, even with the shop's cooperation.
– *Detection-of-double-spending (Unreuseability).* If a dishonest payer spends a coin more than once, the bank should be able to detect the payer's identity with an overwhelming probability.

To our knowledge, most electronic cash protocols have not been proved to be untraceable and unreuseable. Recently, the need for applying formal methods to security protocols has been widely recognized and there have been several attempts to develop a formal framework for specifying and reasoning about security properties. For example, CSP [22] and the spi calculus [5] have been used to analyze security protocols [28,6]. The applied pi calculus [4], which is a variant of the pi calculus [26,27] extended with value passing, function symbol, and equational theory over terms and functions, has been successfully applied to verify some security protocols [2]. The protocol verifier ProVerif [8] provides a set of proof techniques which can be used directly in proving the equivalence between processes of the applied pi calculus.

The main contributions of this paper are summarized as follows.

– We model Ferguson's electronic cash protocol in the applied pi calculus by defining an appropriate signature and equational theory.
– We demonstrate that Ferguson's electronic cash protocol fulfills untraceability and unreuseability. The proofs are partly done by ProVerif.

Related work. Anonymity (untraceability) was first proposed by Chaum [13] to solve the Dinning Cryptographer Problem. After that, a great deal of research has been carried out on this topic and various formal definitions and frameworks for analyzing anonymity have been developed in the literature. For example, Schneider and Sidiropoulos analyzed anonymity with CSP [28]. They used substitution and observable equivalence to define anonymity in CSP. In their framework, the automatic tool FDR [24] was used to check the equivalence of two processes. In [23] Kremer and Ryan analyzed the FOO92 voting protocol with the applied pi calculus and proved that it satisfies anonymity. Chothia [15] used bisimulation in the pi calculus to test the anonymity of an anonymous file-sharing system. Chothia *et al.* [16] proposed a general framework based on the process algebraic verification tool μCRL [10] for checking anonymity and applied it to several protocols, including the Dinning Cryptographer Problem and the FOO92 voting protocol. Our framework is similar to [23], but our proofs are partly done by ProVerif when in [23] all the proofs are done manually.

Other works, such as [7,17,18], considered probabilistic anonymity. Bhargava and Palamidessi [7] formulated their notions of probabilistic anonymity in terms of observables for processes in the probabilistic pi calculus [25]. Deng, Palamidessi and Pang [17] extended the work of [7] and defined the notion of weak probabilistic anonymity and used a probabilistic model checker [21] to automatically analyze the Dining Cryptographers Problem. In [18] Deng, Pang and Wu used the notion of relative entropy from information theory to measure the degree of anonymity a protocol can guarantee, and they proposed a probabilistic process calculus to describe protocols. They considered the scenario with nondeterministic and probabilistic users, which is more realistic. However, the expressiveness of the process calculi they used are less powerful than that of the applied pi calculus used here, which can express all the computations by functions. All of these works on probabilistic anonymity have not been applied to electronic cash protocols.

Organization of the paper. In next section, we briefly introduce the applied pi calculus. In Section 3, we model a simplified version of Ferguson's electronic cash protocol. Two crucial properties of the protocol, untraceability and unreuseability, are analyzed in Sections 4 and 5, respectively. Finally, we conclude the paper and discuss some future work in Section 6.

2 The Applied Pi Calculus

In this section, we give a brief overview of this calculus. The reader is referred to [4] for more details.

2.1 Syntax

To describe a process in the applied pi calculus, one should first define a signature Σ which consists of some function symbols. Given a signature Σ, an infinite set of names, and an infinite set of variables, the set of *terms* are defined below:

$L, M, N, T, U, V ::=$		terms
	$a, b, c, \ldots, k, \ldots, m, n$	name
	x, y, z	variable
	$f(M_1, \ldots, M_l)$	function application

Equational theories play an important role in security protocol analysis. An equational theory over a signature usually consists of a set of equations asserting the equality of cryptographic primitives. For example, in order to model the symmetry cryptography, one can use the equation

$$\mathsf{dec}(\mathsf{enc}(x, y), y) = x.$$

Here x represents a plaintext and y is a key. The binary function symbols enc and dec denote encryption and decryption operation, respectively.

We usually use E to denote an equational theory. The notation $\Sigma \vdash M =_E N$ means the equation $M = N$ is in the theory E associated with Σ.

The definition of plain processes is similar to the one in the pi calculus, except that messages can contain terms rather than names.

$P, Q, R ::=$	$\mathbf{0}$	null process
	$P \mid Q$	parallel composition
	$!P$	replication
	$\nu n.P$	name restriction ("new")
	$\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q$	conditional
	$u(x).P$	message input
	$\bar{u}\langle N \rangle.P$	message output

Extended processes introduce active substitutions and variable restrictions.

$A, B, C ::=$	P	plain process
	$A \mid B$	parallel composition
	$\nu n.A$	name restriction
	$\nu x.A$	variable restriction
	$\{M/x\}$	active substitution

Here $\{M/x\}$ is an active substitution which replaces the variable x with the term M, just like "$\mathit{let}\ x = M\ \mathit{in}\ \ldots$". The active substitution $\{M/x\}$ typically appears when the term M has been sent to the environment. The variable restriction νx restricts the scope of active substitutions. We write $fv(A)$, $bv(A)$, $fn(A)$, and $bn(A)$ for free and bound variables and free and bound names of A.

A *closed* extended process A can be rewritten into a form which consists of a substitution and a plain process with some restricted names:

$$A \equiv \nu\,\tilde{n}.\{\widetilde{M}/\tilde{x}\} \mid P$$

where $fv(P) = \emptyset, fv(\widetilde{M}) = \emptyset$, and $\{\widetilde{n}\} \subseteq fn(\widetilde{M})$.

Every extended process can be mapped to a *frame* $\phi(A)$ which contains only restriction and parallel composition of active substitutions, by replacing every plain process in A with $\mathbf{0}$. The frame $\phi(A)$ can be viewed as static knowledge exposed by A to its environment, but not for A's dynamic behavior. We write $dom(\varphi)$ for the domain of φ, a set of variables which appear in active substitutions in φ but not under a variable restriction.

We write $\phi \vdash M$ to mean M can be deduced from ϕ. This relation is called *deduction* which is axiomatized by the following rules.

$$\textsc{Subst} \quad \frac{}{\nu\,\widetilde{n}.\sigma \vdash M}\ if\ \exists\, x \in dom(\sigma)\ s.t.\ x\sigma = M$$

$$\textsc{Nonce} \quad \frac{}{\nu\,\widetilde{n}.\sigma \vdash s}\ if\ s \notin \widetilde{n}$$

$$\textsc{Funct} \quad \frac{\phi \vdash M_1 \ \cdots\ \phi \vdash M_k}{\phi \vdash f(M_1, \cdots, M_k)}\ f \in \Sigma$$

$$\textsc{Equiv} \quad \frac{\phi \vdash M \quad M =_E N}{\phi \vdash N}$$

An *evaluation context*, denote by $C[\_]$, is a context whose hole is not under a replication, a conditional, an input, or an output. An evaluation context $C[\_]$ closes A when $C[A]$ is closed, and $C[\_]$ is called a *closing evaluation context.*

2.2 Semantics

Structural equivalence, written $A \equiv B$, is the smallest equivalence relation on extended processes that is closed under α-conversion on both names and variables, and under evaluation contexts.

Internal reduction $\rightarrow$ is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts:

$$\begin{array}{lrl}
\textsc{Comm} & \bar{a}\langle x\rangle.P \mid a(x).Q \rightarrow & P \mid Q \\
\textsc{Then} & \mathit{if}\ M = M\ \mathit{then}\ P\ \mathit{else}\ Q \rightarrow & P \\
\textsc{Else} & \mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q \rightarrow & Q\ , \quad \text{when } \Sigma \nvdash M =_E N
\end{array}$$

As usual, labeled operational semantics extends reduction semantics and enables us to reason about the interaction between processes and the environment.

We give a labeled operational semantics which defines transitions of the form $A \xrightarrow{\alpha} A'$ with α being a label of input or output action. The following rules are adopted in the labeled operational semantics.

IN $$a(x).P \xrightarrow{a(M)} P\{M/x\}$$

OUT-ATOM $$\bar{a}\langle u\rangle.P \xrightarrow{\bar{a}\langle u\rangle} P$$

OPEN-ATOM $$\frac{A \xrightarrow{\bar{a}\langle u\rangle} A' \quad u \neq a}{\nu u.A \xrightarrow{\nu u.\bar{a}\langle u\rangle} A'}$$

SCOPE $$\frac{A \xrightarrow{\alpha} A' \quad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

PAR $$\frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

STRUCT $$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

2.3 Equivalence Relations

We write $A \Downarrow a$ when A can send a message on channel a, i.e., $A \rightarrow^* C[\bar{a}\langle M\rangle.P]$ for some evaluation context $C[\_]$ that does not bind a.

Definition 1 (Observational equivalence). *Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A \ \mathcal{R} \ B$ implies:*

1. *if $A \Downarrow a$, then $B \Downarrow a$;*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \ \mathcal{R} \ B'$ for some B';*
3. *$C[A] \ \mathcal{R} \ C[B]$ for all closing evaluation contexts $C[\_]$.*

Definition 1 says that two processes which cannot be distinguished in any context are observationally equivalent. The context usually denotes an attacker.

Definition 2. *We say that two terms M and N are equal in the frame ϕ, and write $(M = N)\phi$, if and only if $\phi \equiv \nu\,\widetilde{n}.\sigma$, $M\sigma = N\sigma$, and $\{\widetilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$ for some names $\widetilde{n}$ and substitution σ.*

Definition 3 (Static equivalence). *Two closed frames φ and ψ are statically equivalent, written $\varphi \approx_s \psi$, for $\varphi \equiv \nu\tilde{n}_1\sigma_1$ and $\psi \equiv \nu\tilde{n}_2\sigma_2$, when $dom(\varphi) = dom(\psi)$ and when, for all terms M and N, we have $(M = N)\varphi$ if and only if $(M = N)\psi$.*

Two extended processes are static equivalent if and only if $\phi(A) \approx_s \phi(B)$.

Static equivalence only defines a relation on frames, which are static knowledge exposed to the environment by some processes. More details about static equivalence can be found in [3].

Definition 4 (Labeled bisimilarity). *Labeled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A \mathcal{R} B$ implies:*

1. $A \approx_s B$;
2. *if* $A \rightarrow^* A'$, *then* $B \rightarrow^* B'$ *and* $A' \mathcal{R} B'$ *for some* B'
3. *if* $A \xrightarrow{\alpha} A'$ *and* $fv(\alpha) \subseteq dom(A)$ *and* $bn(\alpha) \cap fn(B) = \emptyset$, *then* $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$

The following theorem was proved in [4].

Theorem 1. $\approx_l = \approx$ *and* $\approx \subseteq \approx_s$.

Theorem 1 shows that observational equivalence coincides with labeled bisimilarity and observational equivalence is finer than static equivalence. Labeled bisimilarity does not consider all the contexts, which leads to easy proofs in many occasions.

3 Modeling an Electronic Cash Protocol

In this section, we first introduce a simplified version of Ferguson's electronic cash protocol [19], then we define an appropriate equational theory with reasonable abstraction for the blind signatures scheme in the protocol. Finally we model the protocol in the applied pi calculus from the viewpoint of three types of principals.

3.1 Ferguson's Electronic Cash Protocol

As mentioned in the introduction, the protocol consists of three types of participants, the *bank*, the *payer*, and the *shop*. The protocol employs a randomized blind signatures scheme based on RSA-signature scheme to ensure that the payer can obtain the bank's signature on the coin, while the bank has no knowledge of the coin. A polynomial secret sharing scheme is used for double-spending detection. The whole protocol can be described by three sub-protocols:

Withdraw Stage

1. The payer chooses three numbers c, k, and g, where c represents a coin, k is a random number, and g is a blinding factor;
2. The payer blinds the coin c with k and g using blinding function $\mathsf{blind}(c, k, g)$, and then sends this blinded coin together with its identity U to the bank over an authenticated channel c_1;
3. The bank signs on the blinded coin using the randomized blind signatures scheme, generates two blinded signatures s_1 and s_2, and sends them back to the payer on c_2;
4. The payer unblinds these two signatures received from the bank using unblinding function $\mathsf{unblind}$ and blinding factor g, and obtains $\mathsf{sign}(C^k A, prvkey)$ and $\mathsf{sign}(C^U B, prvkey)$, which are two RSA signatures with the bank's private key $prvkey$ on $C^k A$ and $C^U B$ separately. Here $C = f_c(c), A = f_a(c), B = f_b(c)$, and f_c, f_a, f_b are public suitable one-way functions.

Payment Stage

1. The payer sends the coin c to the shop over a secret channel c_{pay};
2. The shop sends the payer a randomly chosen (non-zero) challenge x on c_3;
3. The payer computes a response $r = kx + U$ (the share of the payer's identity for double-spending detection) and a signature on $C^r A^x B$ which can be easily deduced from the two RSA-signatures obtained from the bank, and sends them on channel c_4;
4. The shop can verify the signature on $C^r A^x B$ with the bank's public key to ensure the validity of the payer's coin, and then completes the trade with the payer.

Deposit Stage

1. The shop sends the coin c, the challenge x and the response (both r and the signature) to the bank on channel c_5 to deposit the coin;
2. The bank can check whether the coin is valid by verifying the correctness of the signature, and then stores (c, x, r) in the Used-coin Database and informs the shop on channel c_{payOK} if the coin is valid;
3. If the payer spends the coin c twice dishonestly, then there must be two entries (c, x_1, r_1) and (c, x_2, r_2) associated with c in the database. Since (x_1, r_1) and (x_2, r_2) are two different points on the line $r = kx + U$, the bank is able to determine the payer's identity U immediately.

3.2 Equational Theory

To analyze Ferguson's electronic cash protocol, we define signature Σ for cryptography primitives in the following way:

$\mathsf{blind}(c, k, g)$	(* *blind a coin with k, and g* *)
$\mathsf{unblind}(c, g)$	(* *undo blinding* *)
$\mathsf{pk}(sk)$	(* *get public key from private key* *)
$\mathsf{blindsignA}(c, U, sk)$	(* *blind signature algorithm A* *)
$\mathsf{blindsignB}(c, U, sk)$	(* *blind signature algorithm B* *)
$\mathsf{sign}(m, sk)$	(* *RSA signature scheme* *)
$\mathsf{checksign}(m, s, pk)$	(* *verify RSA signature* *)
$\mathsf{hash1}(c, k, U)$	(* *hash1(c,k,U)=*$C^k A$ *)
$\mathsf{hash2}(c, k, U)$	(* *hash2(c,k,U)=*$C^U B$ *)
$\mathsf{hash3}(c, x, r)$	(* *hash3(c,x,r)=*$C^r A^x B$ *)
$\mathsf{resp\text{-}r}(x, k, U)$	(* *polynomial secret sharing scheme* *)
$\mathsf{resp\text{-}s}(x, s_1, s_2)$	(* *compute the coin's signature from the two given by the bank* *)
$\mathsf{reveal}(x, y)$	(* *determine the identity of the double-spender* *)
$(\_,\_), (\_,\_,\_), \ldots$	(* *constructors for combined messages* *)
$\mathsf{F}_1, \ldots, \mathsf{F}_i, \ldots$	(* *projections for combined messages* *)

The equational theory on Σ is given as follows.

– For convenience, we introduce $(\_\,,\_)$, $(\_\,,\_\,,\_)$, ... to denote combinations of messages, and F_i extracts the i-th component of the combined messages. An example of the equation is:

$$\mathsf{F}_i((x_1, \ldots, x_i, \ldots, x_n)) = x_i$$

– For f_a, f_b, and f_c, which are three public suitable one-way functions used in the protocol, we introduce hash1, hash1, and hash3 for some kinds of combinations of f_a, f_b and f_c:

$$\begin{aligned}\mathsf{hash1}(coin, k, U) &= C^k A\\ \mathsf{hash2}(coin, k, U) &= C^U B\\ \mathsf{hash3}(coin, x, r) &= C^r A^x B\end{aligned}$$

There is no equation on hash1, hash2, and hash3, because these one-way functions are collision-free.

– In order to model the RSA signature scheme, sign and checksign are employed to denote corresponding signature and verification procedure, and the function symbol pk is used for deriving public-key from private-key. The equation is:

$$\mathsf{checksign}(M, \mathsf{sign}(M, sk), \mathsf{pk}(sk)) = true$$

– The randomized blind signature scheme in the protocol runs as follows: The payer first blinds the coin c with random number k and blinding factor g, and then sends it together with the payer's identity to the bank. The bank executes two special blind signature algorithms A and B with its private key and the payer's identity to generate blinded signatures, from which the payer can obtain two signatures over $C^k A$ and $C^U B$ (in RSA signature scheme) after unblinding operation. The equations are describe as follows:

$$\begin{aligned}\mathsf{unblind}(\mathsf{blindsignA}(\mathsf{blind}(coin, k, g), U, sk), g) &= \mathsf{sign}(\mathsf{hash1}(coin, k, U), sk)\\ \mathsf{unblind}(\mathsf{blindsignB}(\mathsf{blind}(coin, k, g), U, sk), g) &= \mathsf{sign}(\mathsf{hash2}(coin, k, U), sk)\end{aligned}$$

– Note that we use symbolic abstraction of blind function for the protocol. Since there is no equation on blind functions, a blinded coin can be viewed as a fresh, opaque message, apparently unrelated to the coin c, when the blinding factor g is not revealed.

$$\nu n.\bar{c}\langle n\rangle \approx \nu k.\nu g.\bar{c}\langle \mathsf{blind}(coin, k, g)\rangle$$

– Finally, the polynomial secret sharing scheme in the protocol is modeled with three function symbols. The first one, $\mathsf{resp\text{-}s}(x, s_1, s_2)$, denotes a response over challenge x and two signature s_1 and s_2 which are obtained from the bank by the payer; the second one, $\mathsf{resp\text{-}r}(x, k, U)$, denotes a response over challenge x on the line $r = kx + U$; the last one, reveal, is a special function symbol by which the bank can detect the double-spender's identity. The equations are defined below:

$$\begin{aligned}&\mathsf{resp\text{-}s}(x, \mathsf{sign}(\mathsf{hash1}(coin, k, U), sk), \mathsf{sign}(\mathsf{hash2}(coin, k, U), sk))\\ &\qquad = \mathsf{sign}(\mathsf{hash3}(c, x, \mathsf{resp\text{-}r}(x, k, U)), sk)\end{aligned}$$

$$\mathsf{reveal}(\mathsf{resp\text{-}r}(x_1, k, U), \mathsf{resp\text{-}r}(x_2, k, U)) = U, where\ x_1 \neq x_2$$

3.3 The Payer Process

The payer process models the role of a payer, as the one in the protocol which is mentioned in Section 3.1. First, the payer generates a fresh random number k, a fresh blind factor g, and a coin c. Next, the payer blinds the coin c with k and g, and then sends this blinded coin together with its identity U to the bank, expecting two corresponding signatures. Finally, the payer sends the coin c to the shop and accomplishes a challenge-response procedure with the shop.

It should be noticed that $c_1, \ldots, c_n$ are public channels, used only for synchronizing different protocol steps. For example, the payer process sends its first message on channel c_1, and then the bank process will receive this message on the same channel. Since these channels are public, the adversary will know what has been sent on these channels.

We write $let\ x = M\ in\ P$ instead of $P\{M/x\}$ for ease of understanding.

$$
\begin{aligned}
P_{payer} ::= {} & \nu k.\nu g.\nu c. \\
& \overline{c_1}\langle(\mathsf{blind}(c, k, g), U)\rangle. \\
& c_2(x_1). \\
& let\ signature_1 = \mathsf{unblind}(\mathsf{F}_1(x_1), g)\ in \\
& let\ signature_2 = \mathsf{unblind}(\mathsf{F}_2(x_1), g)\ in \\
& if\ \mathsf{checksign}(\mathsf{hash1}(c, k, U), signature_1, Pub_{bank}) = true\ then \\
& if\ \mathsf{checksign}(\mathsf{hash2}(c, k, U), signature_2, Pub_{bank}) = true\ then \\
& \overline{c_{pay}}\langle c\rangle. \\
& c_3(x_2). \\
& \overline{c_4}\langle(\mathsf{resp\text{-}r}(x_2, k, U), \mathsf{resp\text{-}s}(x_2, signature_1, signature_2))\rangle
\end{aligned}
$$

3.4 The Bank Process

The behavior of the bank is modeled by the process below. After receiving a blinded coin from the payer, the bank sends back two blind signatures, and debits one dollar from the payer's account at the same time. When the shop requests to verify the validity of a coin, the bank first verifies the correctness of signature, and then deposits one dollar to the shop's account.

$$
\begin{aligned}
P'_{Bank} ::= {} & (c_1(x_1). \\
& let\ blindcoin = \mathsf{F}_1(x_1)\ in \\
& let\ U = \mathsf{F}_2(x_1)\ in \\
& \overline{c_2}\langle(\mathsf{blindsignA}(blindcoin, U, Prv_{bank}), \mathsf{blindsignB}(blindcoin, U, Prv_{bank}))\rangle) \mid \\
& (c_5(x_2). \\
& let\ coin = \mathsf{F}_1(x_2)\ in \\
& let\ challenge = \mathsf{F}_2(x_2)\ in \\
& let\ response = \mathsf{F}_3(x_2)\ in \\
& let\ signature = \mathsf{F}_4(x_2)\ in \\
& if\ \mathsf{checksign}(\mathsf{hash3}(coin, challenge, response), signature, Pub_{bank}) = true \\
& then\ \overline{c_{payOK}}\langle\rangle)
\end{aligned}
$$

The bank's public key is defined as a context, in which the private key remains secret and the public is exported.

$$Key_{bank}[\_] ::= \nu Prv_{bank}.(\{\mathsf{pk}(Prv_{bank})/Pub_{bank}\} \mid [\_])$$

The bank process is defined as follows, and the replication operator enables the bank process to deal with multiple requests from payers and shops.

$$P_{Bank} ::= Key_{bank}[!P'_{Bank}]$$

3.5 The Shop Process

The shop is modeled as the process below. In order to determine whether a paid coin is valid, the shop initiates the challenge-response procedure with the payer. Then the shop sends the coin and the result of the challenge-response procedure to the bank to deposit this coin.

$$\begin{aligned}
P_{shop} ::= \nu\ x.& \\
& c_{pay}(x_1). \\
& let\ coin_{pay} = x_1\ in \\
& \overline{c_3}\langle x\rangle. \\
& c_4(x_2). \\
& let\ response = \mathsf{F}_1(x_2)\ in \\
& let\ signature = \mathsf{F}_2(x_2)\ in \\
& if\ \mathsf{checksign}(\mathsf{hash3}(coin_{pay}, x, response), signature, Pub_{bank}) = true \\
& then\ \overline{c_5}\langle(coin_{pay}, x, response, signature)\rangle. \\
& c_{payOK}()
\end{aligned}$$

3.6 The System Process

The whole system is obtained by putting in parallel the three components, the payer process, the bank process, and the shop process. Notice that the public key Pub_{bank} used in P_{payer} and P_{shop} is exported by P_{bank}, while the private key of the bank remains secret.

$$P_{system} ::= P_{bank} \mid P_{payer} \mid P_{shop}$$

4 Analysis of Untraceability

A formal definition of untraceability was first proposed in [20]. We follow this formal definition and analyze Ferguson's protocol in the applied pi calculus. Here, we only consider passive attackers who eavesdrop on channels.

Definition 5 (Untraceability). *Let passive attacker $\mathcal{A}$ has access to all bank's views of withdraw, payment, and deposit protocols. Then for any two coins C_i, C_j and two withdraws W_0, W_1 such that C_i and C_j are originated from W_0 and W_1, $\mathcal{A}$ cannot distinguish whether C_i comes from W_0 or W_1.*

For ease of understanding, we specialize the above general definition to a simple system with two payers P_1 and P_2. Suppose P_1 withdraws $coin_1$, and P_2 withdraws $coin_2$.

$$P_1 ::= P_{payer}\{coin_1/c, payer_1/U\}$$
$$P_2 ::= P_{payer}\{coin_2/c, payer_2/U\}$$

We say that an electronic cash protocol satisfies the requirement of untraceability, when process P_1 with $coin_1$ paralleled by process P_2 with $coin_2$ is observationally equivalent to process P_1 with $coin_2$ paralleled by process P_2 with $coin_1$, i.e., $P_1 \mid P_2 \approx P_1\{coin_2/coin_1\} \mid P_2\{coin_1/coin_2\}$.

Theorem 2 (Untraceability). *Ferguson's electronic cash protocol satisfies the requirement of untraceability.*

Proof. Proving equivalences of two processes which differ only in the choice of some terms is supported by ProVerif [8,9]. We benefit from this feature of ProVerif, and the proof is partly done by this tool.

We use ProVerif to prove the following equivalence

$$P_{payer}\{coin_1/c\} \approx P_{payer}\{coin_2/c\}.$$

The code for proving this equivalence in included in the appendix. Based on the result of ProVerif, the process of withdraw stage is independent from the payment stage. Even if the bank and the shop cooperate, they cannot distinguish whether P_1 withdraws $coin_1$ or $coin_2$, so we have

$$P_1 \approx P_1\{coin_2/coin_1\}.$$

Thus, from the structural equivalence, the following equivalence is obvious.

$$P_1 \mid P_2 \approx P_1\{coin_2/coin_1\} \mid P_2\{coin_1/coin_2\} \qquad \square$$

5 Analysis of Unreuseability

In this section we analyze the unreuseability property of the protocol modeled in Section 3. In an off-line electronic cash protocol, after receiving a coin, the shop does not deposit the coin immediately. Thus, the strategy adopted by the system is to detect the behavior of double-spending instead of preventing it. The formal definition of unreuseability from [20] is given below.

Definition 6 (Unreuseability). *If a coin is successfully deposited twice, then the identity of at least one misbehaving user can be efficiently computed and proved from the bank's view of the deposit.*

In Ferguson's protocol, the payer's identity is embedded in the payment stage by the polynomial secret sharing scheme. Repeated execution of the challenge-response procedure over a same coin will certainly reveal the identity of the payer. Since ProVerif cannot examine the knowledge of a particular participant, the proof of Theorem 3 is done manually.

Theorem 3 (Unreuseability). *Ferguson's electronic cash protocol can detect the identity of a* double-spender, *i.e. if a dishonest payer spends a coin twice*

$$P \xrightarrow{\nu y_1.\overline{c_{pay}}\langle y_1\rangle.c_3(x_2).\nu y_2.\overline{c_4}\langle y_2\rangle} \xrightarrow{\nu y_3.\overline{c_{pay}}\langle y_3\rangle.c_3(x_3).\nu y_4.\overline{c_4}\langle y_4\rangle} A',\ \textit{and}\ (y_1 = y_3)\phi(A')$$

then the payer's identity must be revealed by the bank,

$$\phi(A') \vdash payer$$

Proof. Without loss of generality, we construct a process representing the misbehaving payer.

$$\begin{aligned}
P_{double-spender} ::= {} & \nu k.\nu g.\nu c.\\
& \overline{c_1}\langle(\mathsf{blind}(c,k,g),U)\rangle.\\
& c_2(x_1).\\
& let\ signature_1 = \mathsf{unblind}(\mathsf{F}_1(x_1),g)\ in.\\
& let\ signature_2 = \mathsf{unblind}(\mathsf{F}_2(x_1),g)\ in.\\
& if\ \mathsf{checksign}(\mathsf{hash1}(c,k,U), signature_1, Pub_{bank}){=}true\ then\\
& if\ \mathsf{checksign}(\mathsf{hash2}(c,k,U), signature_2, Pub_{bank}){=}true\ then\\
& \overline{c_{pay}}\langle c\rangle.\\
& c_3(x_2).\\
& \overline{c_4}\langle(\mathsf{resp\text{-}r}(x_2,k,U), \mathsf{resp\text{-}s}(x_2, signature_1, signature_2))\rangle.\\
& \overline{c_{pay}}\langle c\rangle.\\
& c_3(x_3).\\
& \overline{c_4}\langle(\mathsf{resp\text{-}r}(x_3,k,U), \mathsf{resp\text{-}s}(x_3, signature_1, signature_2))\rangle
\end{aligned}$$

$$P ::= P_{double-spender}\{payer/U\}$$

After withdrawing the coin from the bank, and spending the coin twice,

$$P \xrightarrow{\nu y.\overline{c_1}\langle y\rangle.c_2(x_1)} \xrightarrow{\nu y_1.\overline{c_{pay}}\langle y_1\rangle.c_3(x_2).\nu y_2.\overline{c_4}\langle y_2\rangle} \xrightarrow{\nu y_3.\overline{c_{pay}}\langle y_3\rangle.c_3(x_3).\nu y_4.\overline{c_4}\langle y_4\rangle} A'$$

The knowledge exposed by the double-spender to the bank and the shop is

$$\begin{aligned}
\phi(A') = {} & \nu k.\nu g.\nu c.\\
& (\{(\mathsf{blind}(c,k,g), payer)/y\} \mid \{c/y_1\} \mid \{c/y_3\} \mid\\
& \{(\mathsf{resp\text{-}r}(x_2,k,payer), \mathsf{resp\text{-}s}(x_2, signature_1, signature_2)/y_2\} \mid\\
& \{(\mathsf{resp\text{-}r}(x_3,k,payer), \mathsf{resp\text{-}s}(x_3, signature_1, signature_2)/y_4\})
\end{aligned}$$

Notice that x_2 and x_3 are two challenges initiated by the shop in different sessions, so we can infer $x_2 \neq x_3$. From the equations on resp-r and reveal, we have

$$\frac{\phi(A') \vdash \mathsf{resp\text{-}r}(x_2,k,payer),\ \phi(A') \vdash \mathsf{resp\text{-}r}(x_3,k,payer)}{\phi(A') \vdash \mathsf{reveal}(\mathsf{resp\text{-}r}(x_2,k,payer), \mathsf{resp\text{-}r}(x_3,k,payer)) = payer}$$

Clearly, this protocol satisfies *unreuseability.* □

6 Conclusion and Future Work

In this paper we have modeled Ferguson's electronic cash protocol in the applied pi calculus, and we have verified that it satisfies the security properties of untraceability and unreuseability. We believe that the approach used in this paper is suitable for analyzing many other similar electronic cash protocols as well.

As for the future work, it would be interesting to extend the framework of this paper to a probabilistic setting and use it to analyze anonymity of electronic cash protocols. We would also like to analyze other crucial properties of electronic cash protocols, such as divisibility and transferability, and to develop an automatic verification tool to verify security requirements of the protocols. Another future direction is to look at on-line electronic cash protocols, where the bank systems are much more complicated to model.

References

1. Martín Abadi, Cédric Fournet: Private Authentication. Theoretical Computer Science **322** (2004) 427–476.
2. Martín Abadi, Bruno Blanchet, Cédric Fournet: Just Fast Keying in the Pi Calculus. Proceedings of ESOP 2004, Lecture Notes in Computer Science **2986** (2004) 340–354.
3. Martín Abadi, Véronique Cortier: Deciding Knowledge in Security Protocols under Equational Theories. Theoretical Computer Science **367** (2006) 2–32.
4. Martín Abadi, Cédric Fournet: Mobile Values, New Names, and Secure Communication. Proceedings of POPL 2001, 104-115.
5. Martín Abadi, Andrew D. Gordon: A Calculus for Cryptographic Protocols: The Spi Calculus. Information and Computation **148** (1999) 1–70.
6. Martín Abadi, Andrew D. Gordon: Reasoning about Cryptographic Protocols in the Spi Calculus. Proceedings of CONCUR 1997, Lecture Notes in Computer Science **1243** (1997) 59–73.
7. Mohit Bhargava, Catuscia Palamidessi: Probabilistic Anonymity. Proceedings of CONCUR 2005, Lecture Notes in Computer Science **3653** (2005) 65–75.
8. Bruno Blanchet: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. Proceedings of CSFW 2001, 82–96.
9. Bruno Blanchet, Martín Abadi, Cédric Fournet: Automated Verification of Selected Equivalences for Security Protocols. Proceedings of LICS 2005, 331–340.
10. Stefan Blom, Wan Fokkink, Jan Friso Groote, Izak van Langevelde, Bert Lisser, Jaco van de Pol: μCRL: A Toolset for Analysing Algebraic Specifications. Proceedings of CAV 2001, Lecture Notes in Computer Science **2102** (2001) 250–254.
11. Stefan A. Brands: Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). Proceedings of CRYPTO 1993, Lecture Notes in Computer Science **773** (1994) 302–318.
12. Stefan A. Brands: An Efficient Off-line Electronic Cash System Based On The Representation Problem. CWI Technical Report, CS-R9323 (1993).
13. David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Journal of Cryptology **1** (1988) 65–75.

14. David Chaum, Amos Fiat, Moni Naor: Untraceable Electronic Cash. Proceedings of CRYPTO 1988, Lecture Notes in Computer Science **403** (1990) 319–327.
15. Tom Chothia: Analysing the MUTE Anonymous File-Sharing System Using the Pi-Calculus. Proceedings of FORTE 2006, Lecture Notes in Computer Science **4229** (2006) 115–130.
16. Tom Chothia, Simona Orzan, Jun Pang, Mohammad Torabi Dashti: A Framework for Automatically Checking Anonymity with μCRL. Proceedings of TGC 2006, Lecture Notes in Computer Science (to appear).
17. Yuxin Deng, Catuscia Palamidessi, Jun Pang: Weak Probabilistic Anonymity. Proceedings of SECCO 2005, Electronic Notes in Theoretical Computer Science (to appear).
18. Yuxin Deng, Jun Pang, Peng Wu: Measuring Anonymity with Relative Entropy. Proceedings of FAST 2006, Lecture Notes in Computer Science (to appear).
19. Niels Ferguson: Single Term Off-Line Coins. Proceedings of EUROCRYPT 1993, Lecture Notes in Computer Science **765** (1993) 318–328.
20. Matthew K. Franklin, Moti Yung: Secure and Efficient Off-Line Digital Money (Extended Abstract). Proceedings of ICALP 1993, Lecture Notes in Computer Science **700** (1993) 265–276.
21. Andrew Hinton, Marta Kwiatkowska, Gethin Norman, David Parker: PRISM: A Tool for Automatic Verification of Probabilistic Systems. Proceedings of TACAS 2006, Lecture Notes in Computer Science **3920** (2006) 441–444.
22. Tony Hoare: Communicating Sequential Processes. Communications of the ACM **21** (1978) 666–677.
23. Steve Kremer, Mark Ryan: Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. Proceedings of ESOP 2005, Lecture Notes in Computer Science **3444** (2005) 186–200.
24. Gavin Lowe: Breaking and Fixing the Needham-Schroeder Public-Key Proceedings of TACAS 1996, Lecture Notes in Computer Science **1055** (1996) 147–166.
25. Oltea Mihaela Herescu, Catuscia Palamidessi: Probabilistic Asynchronous Pi-Calculus. Proceedings of FoSSaCS 2001, Lecture Notes in Computer Science **1784** (2001) 146–160.
26. Robin Milner, Joachim Parrow, David Walker: A Calculus of Mobile Processes, I. Information and Computation **100** (1992) 1–40.
27. Robin Milner, Joachim Parrow, David Walker: A Calculus of Mobile Processes, II. Information and Computation **100** (1992) 41–77.
28. Steve Schneider, Abraham Sidiropoulos: CSP and Anonymity. Proceedings of ESORICS 1996, Lecture Notes in Computer Science **1146** (1996) 198–218.
29. Yiannis Tsiounis: Efficient Electronic Cash: New Notions and Techniques. PhD Thesis, Northeastern University, 1997.

Appendix - The Code for Proving Theorem 2 in ProVerif

We use this piece of code to prove the following equivalence

$$P_{payer}\{coin_1/c\} \approx P_{payer}\{coin_2/c\}.$$

The P_{payer} is described by `processP` below.

```
(* Ferguson's E-cash protocol *)

(* constant and constructor *)
data true/0.
data U/0.
data con2/2.

(* signature *)
fun blind/3.
(* fun unblind/2. *)
fun pk/1.
fun blindsignA/3.
fun blindsignB/3.
fun sign/2.
fun checksign/3.
fun hash1/3.
fun hash2/3.
fun hash3/3.
fun respr/3.
(* fun resps/3. *)
(* fun reveal/2. *)

(* equational theory *)
equation checksign(m,sign(m,sk),pk(sk))=true.
reduc unblind(blindsignA(blind(c,k,g),U,sk),g)
      =sign(hash1(c,k,U),sk);
      unblind(blindsignB(blind(c,k,g),U,sk),g)
      =sign(hash2(c,k,U),sk).
reduc resps(x,sign(hash1(c,k,U),sk),sign(hash2(c,k,U),sk))
      =sign(hash3(c,x,respr(x,k,U)),sk).
reduc reveal(respr(x1,k,U),respr(x2,k,U))=U.

(* channel *)
free cpk.
free c1.
free c2.
free c3.
free c4.
private free cpay.

let processP = new k; new g; new coin1; new coin2;
           in (cpk, pubBank);
           let c = choice[coin1,coin2] in
           out (c1, con2(blind(c,k,g),U));
           in (c2, con2(bs1,bs2));
           let s1 = unblind(bs1,g) in
           let s2 = unblind(bs2,g) in
           if checksign(hash1(c,k,U),s1,pubBank) = true then
           if checksign(hash2(c,k,U),s2,pubBank) = true then
           out (cpay, coin);
           in (c3, x);
           out (c4, con2(respr(x,k,U),resps(x,s1,s2))).

process processP
```

Cryptanalysis of the TRMC-4 Public Key Cryptosystem

Xuyun Nie[1], Lei Hu[1,⋆], Jintai Ding[2,3,⋆⋆], Jianyu Li[1], and John Wagner[2]

[1] State Key Laboratory of Information Security, Graduate School of Chinese Academy of Sciences, Beijing 100049, China
[2] Department of Mathematical Sciences, University of Cincinnati, Cincinnati, OH, 45220, USA
[3] Fachbereich Informatik, Technische Universität Darmstadt, Germany
{nxy04b,hu}@is.ac.cn, ding@math.uc.edu, ljy@is.ac.cn, wagnerjh@email.uc.edu

Abstract. In 2006, the inventors of TRMC public key cryptosystem proposed a new variant of TRMC, TRMC-4, which can resist the existing attack, in particular, the Joux et al attack. In this paper, we show that the new version is vulnerable to attack via the linearization equations (LE) method. For any given valid ciphertext and its corresponding TRMC-4 public key, we can derive the corresponding plaintext within 2^{24} $\mathbb{F}_{2^8}$-operations, after performing once for the public key a computation of complexity less than 2^{34}. Our results are confirmed by computer experiments.

Keywords: multivariate public key cryptosystem, quadratic polynomial, algebraic cryptanalysis, linearization equation, TRMC.

1 Introduction

For the last three decades, public key cryptosystems (PKC) become an indispensable part of our modern communication system. The security of traditional PKC, such as RSA and ElGamal, depends on hard number theory based problems such as factoring or discrete logarithms. However, due to the quantum computer attack by Shor [Sho97], and demand for more efficient cryptosystems for small devices, there is a need to search for alternatives which are based on other classes of problems.

Multivariate public key cryptosystem (MPKC) is a promising alternative. Different from traditional PKC, the public key of MPKC is usually a set of quadratic polynomials. The security of MPKC relies on the difficulty of solving systems of nonlinear polynomial equations with many variables, and the latter is an NP-hard problem in general. Compared with RSA public key cryptosystems,

⋆ The work of this author is supported by NSFC (60573053) and National 863 Project of China (2006AA01Z416).

⋆⋆ The work of this author is partially supported by the Charles Phelps Taft Research Center and the Alexander von Humboldt Foundation.

the computation in MPKC can be very fast because it is operated on a small finite field.

The first promising construction of MPKC is the Matsumoto-Imai (MI) scheme [MI88] proposed in 1988. Unfortunately, it was defeated by Patarin in 1995 with the linearization method [Pat95].

Tractable rational map cryptosystem (TRMC) is a family of MPKC. It is a type of stepwise triangular system (STS) [Wo05]. There are some STS schemes such as TTM cryptosystems [Moh99] and TTS signature schemes [YC05]. All existing instances of TTM have a common defect: their plaintext and ciphertext variables always satisfy some linearization equations. Hence, they are all insecure [GC00], [DH03], [DS03], [NHLCD06]. Compared to TTM, the construction of TRMC is more systematic. Its central map is a so-called tractable rational map.

A previous version of TRMC is TRMC-2. The decryption of TRMC-2 involves solving a sub-system of equations. Joux et al pointed out that the existence of the sub-system turned out to be a weakness [JKMR05]. Utilizing this weakness, Joux et al introduced a variant of the XL algorithm and built a pseudo-private key equivalent to the original private key for a given valid ciphertext. With this pseudo-private key, they find the corresponding plaintext.

To avoid this attack, the inventors of TRMC proposed TRMC-4 [WC04] recently. But unfortunately, we find there exist some linearization equations satisfied by plaintext variables m_i and ciphertext variables w_j, namely

$$\sum_{i=1,j=1}^{n,m} a_{ij} m_i w_j + \sum_{i=1}^{n} b_i m_i + \sum_{j=1}^{m} c_j w_j + d = 0.$$

Linearization equation attack was proposed first by Patarin in 1995 to defeat the MI scheme [Pat95]. The linearization equation is also called the Patarin relation. The authors claimed that it would be computationally infeasible if one carefully designs the tractable rational maps [WC04]. But for TRMC-4, we find that there are some Paratin relations in TRMC-4 and we can find all linearization equations in 2^{34} operations. Then for a given valid ciphertext, via three eliminations, we can find the corresponding plaintext in 2^{24} operations.

This paper is organized as follows. We introduce tractable rational map and TRMC-4 encryption scheme in Section 2. In Section 3, we describe how to attack TRMC-4, present a practical attack procedure, and calculate the complexity of our attack. Finally, in Section 4, we conclude the paper.

2 TRMC Cryptosystems

2.1 Tractable Rational Map

TRMC is an MPKC. Its central map is a so-called tractable rational map, which is different from other MPKCs such as TTM etc..

Let K be a finite field. A tractable rational map is a map on K of following form:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_j \\ \vdots \\ y_n \end{pmatrix} = \phi \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} r_1(x_1) \\ r_2(x_2) \cdot \frac{p_2(x_1)}{q_2(x_1)} + \frac{f_2(x_1)}{g_2(x_1)} \\ \vdots \\ r_j(x_j) \cdot \frac{p_j(x_1,x_2,\cdots,x_{j-1})}{q_j(x_1,x_2,\cdots,x_{j-1})} + \frac{f_j(x_1,x_2,\cdots,x_{j-1})}{g_j(x_1,x_2,\cdots,x_{j-1})} \\ \vdots \\ r_n(x_n) \cdot \frac{p_n(x_1,x_2,\cdots,x_{n-1})}{q_n(x_1,x_2,\cdots,x_{n-1})} + \frac{f_n(x_1,x_2,\cdots,x_{n-1})}{g_n(x_1,x_2,\cdots,x_{n-1})} \end{pmatrix}$$

where f_j, g_j, p_j, q_j are polynomials on K, r_j is an invertible polynomial over K whose inverse can easily be computed.

The inverse process is very simple. One can derive $x_1 = r_1^{-1}(y_1)$ from $y_1 = r_1(x_1)$, then compute x_2 from x_1 and y_2. By iteration, we can obtain the values of $x_3, \cdots, x_n$ in turn. So TRMC can be regarded as a triangular system.

2.2 TRMC-4

Let $\mathbb{K} = \mathbb{F}_{2^8}$ be a finite field with 2^8 elements. Map $F : \mathbb{K}^{45} \to \mathbb{K}^{50}$ is a composition of 4 maps $\phi_1, \phi_2, \phi_3, \phi_4$. Let

$$(x_1, \cdots, x_{45}) = \phi_1(m_1, \cdots, m_{45}), (y_1, \cdots, y_{50}) = \phi_2(x_1, \cdots, x_{45}),$$

$$(z_1, \cdots, z_{50}) = \phi_3(y_1, \cdots, y_{50}), (w_1, \cdots, w_{50}) = \phi_4(z_1, \cdots, z_{50}),$$

where ϕ_1 and ϕ_4 are invertible affine maps, ϕ_2 and ϕ_3 are tractable rational maps. Note that the central map of TRMC-4 is the composition of two tractable rational maps.

The expressions of ϕ_2 and ϕ_3, except for a few parameters, are public information in the TRMC-4. ϕ_1 and ϕ_4 are taken as the private key, while the expression of the map $(w_0, \cdots, w_{50}) = F(m_0, ..., m_{45})$ is the public key. The public key $F(m_1, \cdots, m_{45})$ is 50 quadratic equations in 45 variables. Denote by F_j the j-th component function of F.

$$\begin{aligned} (w_1, \cdots, w_{50}) &= F(m_1, \cdots, m_{45}) \\ &= \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1(m_1, \cdots, m_{45}) \\ &= (F_1(m_1, \cdots, m_{45}), \cdots, F_{50}(m_1, \cdots, m_{45})) \end{aligned}$$

To list ϕ_2 and ϕ_3, we firstly fix some notation.

Let $\mathbb{E} = \mathbb{F}_{2^{48}}$ be a degree 6 extension field of $\mathbb{K}$. $\pi : \mathbb{E} \to \mathbb{K}^6$ is a natural $\mathbb{K}$-linear isomorphism. Namely we take a basis of $\mathbb{E}$ over $\mathbb{K}$, $\{\theta_1, \cdots, \theta_6\}$, and define π by $\pi(a_1\theta_1 + \cdots + a_6\theta_6) = (a_1, \cdots, a_6)$ for any $a_1, \cdots, a_6 \in \mathbb{K}$. It is natural to regard π as a $\mathbb{K}$-linear isomorphism from $\mathbb{E}^8$ to $\mathbb{K}^{48}$.

In TRMC-4, the intermediate variables $x_1, \cdots, x_{45}, y_1, \cdots, y_{48}$ and $z_1, \cdots, z_{48}$ are grouped into elements in $\mathbb{E}$, shown in Table 1. Here the second and the forth column are the images of entries in the first and the third column, respectively. For example, $\pi(X_1) = c_1\theta_1 + x_1\theta_2 + \cdots + x_5\theta_6$. The $c_1, \cdots, c_6 \in \mathbb{K}$ are constants, such that $c_1, c_4, c_5 \neq 0$ to avoid decryption failure.

Table 1. Intermediate variables and their corresponding entries in $\mathbb{E}$

X_1	$(c_1, x_1, x_2, x_3, x_4, x_5)$	Y_1	$(y_1, y_2, y_3, y_4, y_5, y_6)$
X_2	$(c_2, x_6, x_7, x_8, x_9, x_{10})$	Y_2	$(y_7, y_8, y_9, y_{10}, y_{11}, y_{12})$
X_3	$(c_3, x_{11}, x_{12}, x_{13}, x_{14}, x_{15})$	Y_3	$(y_{13}, y_{14}, y_{15}, y_{16}, y_{17}, y_{18})$
X_4	$(x_{16}, x_{17}, x_{18}, x_{19}, x_{20}, x_{21})$	Y_4	$(y_{19}, y_{20}, y_{21}, y_{22}, y_{23}, y_{24})$
X_5	$(x_{22}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27})$	Y_5	$(y_{25}, y_{26}, y_{27}, y_{28}, y_{29}, y_{30})$
X_6	$(x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{33})$	Y_6	$(y_{31}, y_{32}, y_{33}, y_{34}, y_{35}, y_{36})$
X_7	$(x_{34}, x_{35}, x_{36}, x_{37}, x_{38}, x_{39})$	Y_7	$(y_{37}, y_{38}, y_{39}, y_{40}, y_{41}, y_{42})$
X_8	$(x_{40}, x_{41}, x_{42}, x_{43}, x_{44}, x_{45})$	Y_8	$(y_{43}, y_{44}, y_{45}, y_{46}, y_{47}, y_{48})$
$\tilde{X}_1$	$(c_4, x_1, x_4, x_7, x_{10}, x_{13})$	Z_1	$(z_1, z_2, z_3, z_4, z_5, z_6)$
$\tilde{X}_2$	$(c_5, x_2, x_5, x_8, x_{11}, x_{14})$	Z_2	$(z_7, z_8, z_9, z_{10}, z_{11}, z_{12})$
$\tilde{X}_3$	$(c_6, x_3, x_6, x_9, x_{12}, x_{15})$	Z_3	$(z_{13}, z_{14}, z_{15}, z_{16}, z_{17}, z_{18})$
		Z_4	$(z_{19}, z_{20}, z_{21}, z_{22}, z_{23}, z_{24})$
		Z_5	$(z_{25}, z_{26}, z_{27}, z_{28}, z_{29}, z_{30})$
		Z_6	$(z_{31}, z_{32}, z_{33}, z_{34}, z_{35}, z_{36})$
		Z_7	$(z_{37}, z_{38}, z_{39}, z_{40}, z_{41}, z_{42})$
		Z_8	$(z_{43}, z_{44}, z_{45}, z_{46}, z_{47}, z_{48})$

ϕ_2 is defined as follows:

$$\begin{cases} Y_1 = \tilde{X}_1; \\ Y_2 = \tilde{X}_2\tilde{X}_1; \\ Y_3 = \tilde{X}_3\tilde{X}_2; \\ Y_4 = X_4X_1 + X_3X_2 \\ \begin{pmatrix} Y_5 & Y_6 \\ Y_7 & Y_8 \end{pmatrix} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} \begin{pmatrix} X_5 & X_6 \\ X_7 & X_8 \end{pmatrix} = \begin{pmatrix} X_1X_5 + X_2X_7 & X_1X_6 + X_2X_8 \\ X_3X_5 + X_4X_7 & X_3X_6 + X_4X_8 \end{pmatrix}; \\ y_{49} = L_1L_6 + L_2L_7 + L_3L_8 + L_4L_9 + L_5L_{10}; \\ y_{50} = L_1L_{11} + L_2L_{12} + L_3L_{13} + L_4L_{14} + L_5L_{15}. \end{cases} \tag{2.1}$$

where $L_1, \cdots, L_{15}$ are randomly chosen linear maps in $x_1, \cdots, x_{45}$.

ϕ_3 is defined as follows:

$$\begin{cases} Z_1 = Y_1^2\frac{Y_3}{Y_2} + g_1(\frac{Y_5Y_8+Y_6Y_7}{Y_4}) = \tilde{X}_1\tilde{X}_3 + g_1(X_5X_8 + X_6X_7); \\ Z_2 = Y_2 + g_2(\frac{Y_5Y_8+Y_6Y_7}{Y_4}) = \tilde{X}_2\tilde{X}_1 + g_2(X_5X_8 + X_6X_7); \\ Z_3 = Y_3 + g_3(\frac{Y_5Y_8+Y_6Y_7}{Y_4}) = \tilde{X}_3\tilde{X}_2 + g_3(X_5X_8 + X_6X_7); \\ Z_4 = Y_4 = X_4X_1 + X_3X_2; \\ Z_5 = Y_5 = X_1X_5 + X_2X_7; \\ Z_6 = Y_6 = X_1X_6 + X_2X_8; \\ Z_7 = Y_7 = X_3X_5 + X_4X_7; \\ Z_8 = Y_8 = X_3X_6 + X_4X_8; \\ z_{49} = y_{49}; \\ z_{50} = y_{50}. \end{cases} \tag{2.2}$$

where g_i, $i = 1, 2, 3$, are maps from $\mathbb{E}$ to $\mathbb{E}$, each of them corresponds to a map f_i, where $f_i = \pi \circ g_i \circ \pi^{-1}$, is a $\mathbb{K}$-linear transformation from $\mathbb{K}^6$ to $\mathbb{K}^6$.

The inverting process of TRMC-4 is very simple. Applying ϕ_4^{-1} on $w_1, \cdots, w_{50}$, one can derive the $z_1, \cdots, z_{50}$, then the $Z_1, \cdots, Z_8$ and $Y_4, \cdots, Y_8$. One can compute the value of Y_1, Y_2, Y_3 from the first three formulas of (2.2) and Z_1, Z_2, Z_3. Then from the first three formulas of (2.1), one can obtain $\tilde{X}_1, \tilde{X}_2, \tilde{X}_3$ hence X_1, X_2, X_3. Then one can derive X_4, X_5, X_6, X_7, X_8 from the matrix equation and the fourth equation. So one obtains all the $(x_1, \cdots, x_{45})$. Finally, applying ϕ_1^{-1} on $(x_1, \cdots, x_{45})$, one derives all plaintext $(m_1, \cdots, m_{45})$. Note that if $Z_4 = Y_4 = 0$, the decryption mentioned above will not work.

3 Cryptanalysis on TRMC-4

The inventors of TRMC claimed [WC04] that searching the general Patarin relations would be computationally infeasible by carefully designing the tractable rational maps. But through theoretical analysis, we find that there still exist Patarin relations in TRMC-4 and we can find all Patarin relations in a short times. Given a valid ciphertext, starting from these equations, we can find the corresponding plaintext easily.

3.1 Linearization Equations

Firstly, set

$$M_1 = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}, M_2 = \begin{pmatrix} X_5 & X_6 \\ X_7 & X_8 \end{pmatrix}, M = \begin{pmatrix} Y_5 & Y_6 \\ Y_7 & Y_8 \end{pmatrix} = \begin{pmatrix} Z_5 & Z_6 \\ Z_7 & Z_8 \end{pmatrix}.$$

Denote by A^* the associated matrix of a square matrix; for a second order matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, its associated matrix is $A^* = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

In TRMC-4, we have

$$M = M_1 M_2, \det(M_1) = Y_4 = Z_4,$$

Hence

$$M_2 \det(M_1) = M_1^* M,$$

namely,

$$\begin{pmatrix} X_5 & X_6 \\ X_7 & X_8 \end{pmatrix} Z_4 = \begin{pmatrix} X_4 & X_2 \\ X_3 & X_1 \end{pmatrix} \begin{pmatrix} Z_5 & Z_6 \\ Z_7 & Z_8 \end{pmatrix}. \tag{3.1}$$

Expanding it, that is,

$$\begin{cases} X_4 Z_5 + X_2 Z_7 + X_5 Z_4 = 0; \\ X_2 Z_8 + X_4 Z_6 + X_6 Z_4 = 0; \\ X_1 Z_7 + X_3 Z_5 + X_7 Z_4 = 0; \\ X_1 Z_8 + X_3 Z_6 + X_8 Z_4 = 0. \end{cases} \tag{3.2}$$

Since F is derived from $\phi_3 \circ \phi_2$ by composing from the inner and outer sides by invertible affine maps ϕ_1 and ϕ_4. Hence equation (3.2) imply that for any $(m_1, \cdots, m_{45}) \in K^{45}$ satisfying the equation of the form:

$$\sum_{i=1,j=1}^{45,50} a_{ij} m_i F_j + \sum_{i=1}^{45} b_i m_i + \sum_{j=1}^{50} c_j F_j + d = 0 \tag{3.3}$$

Furthermore, the four equations in (3.2) are all linearly independent, therefore there exist at least 24 linearization equations such that their corresponding coefficient vectors are linearly independent. Actually, given the value of Z_i, the equations in X_i are also linearly independent. Hence, given a valid ciphertext, there still exist at least 24 linearly independent linear equations in $(m_1, \cdots, m_{45})$. Let V denote the $\mathbb{K}$-linear space composed of all linearization equations of the form (3.3), and let $D \geq 24$ be its dimension.

To find all equations in V is equivalent to find a basis of V. The equation (3.3) is equivalent to a system of equations on the coefficients a_{ij}, b_i, c_j, and d. The number of unknowns in these equations is equal to the number of monomials in m_i, F_j. So there are $2346 = 45 \times 50 + 45 + 50 + 1$ unknowns in these equations.

To find a basis of V, we randomly select slightly more than 2346, say 2500, plaintexts $(m_1, \cdots, m_{45})$, substitute them in (3.3) to get a system of 2500 linear equations, and solve the resulting system. Let $\{(a_{ij}^{(\rho)}, b_i^{(\rho)}, c_j^{(\rho)}, d^{(\rho)}), 1 \leq \rho \leq D\}$ be the coefficient vectors corresponding to a basis of V, where i, and j stand for $i = 1, \cdots, 45$, $1 \leq j \leq 50$, respectively. Hence, we derive D linearly independent equations in m_i and F_j. Let $E_\rho (1 \leq \rho \leq D)$ denote the equations:

$$\begin{cases} \sum\limits_{i=1,j=1}^{45,50} a_{ij}^{(\rho)} m_i F_j + \sum\limits_{i=1}^{45} b_i^{(\rho)} m_i + \sum\limits_{j=1}^{50} c_j^{(\rho)} F_j + d^{(\rho)} = 0 \\ (1 \leq \rho \leq D) \end{cases} \tag{3.4}$$

The work above depends only on any given public key, and it can be solved once for all cryptanalysis under that public key.

3.2 First Elimination

Let's assume we have a valid ciphertext $w' = (w'_1, \cdots, w'_{50})$. our goal is to find its corresponding plaintext $m' = (m'_1, \cdots, m'_{45})$.

Substituting $(F_1, \cdots, F_{50}) = (w'_1, \cdots, w'_{50})$ into $E_\rho (1 \leq \rho \leq D)$, we can derive D linear equations in m_i. Reducing these D equations, we can derive a system of linearly independent linear equations. Let l $(l \geq 24)$ denote the number of linearly independent equations in these system. Let $E'_1, \cdots, E'_l$ denote these equations. Doing a simple Gaussian elimination, from these l equations we can represent l variables of $x_1, \cdots, x_{45}$ by linear combinations of other $45 - l$. That is, we can find two disjoint subsets of $\{1, \cdots, 45\}$, $A'_1 = \{u'_1, \cdots, u'_l\}$ and $A_1 = \{u_1, \cdots, u_{45-l}\}$, and linear expressions

$$m_{u'_j} = h_j(m_{u_1}, \cdots, m_{u_{45-l}}), 1 \leq j \leq l \tag{3.5}$$

such that $E'_1, \cdots, E'_l$ holds when (3.5) are substituted into them.

Let S denote a $(45 - l)$-dimensional affine subspace of $\mathbb{K}^{45}$ defined by (3.5); the component $m_{u'_j}$ of any vector $(m_0, \cdots, m_{45})$ in S is $h_j(m_{u_1}, \cdots, m_{u_{45-l}})$.

Now substitute (3.5) into $F_j(m_1, \cdots, m_{45})$ and derive 50 new quadratic functions $\hat{F}_j(m_{u_1}, \cdots, m_{u_{45-l}})$ $(1 \leq j \leq 50)$.

3.3 Second Elimination

Furthermore, through theoretical analysis, we find there still exist linearization equations on S.

Firstly, we denote by Z_i', $i = 1, \cdots, 8$, the value of Z_i corresponding to a given valid ciphertext $w' = (w_1', \ldots, w_{50}')$. Similar notations Y_i', X_i', $\tilde{X}_i'$, x_i' and m_i' are denoted for Y_i, X_i, $\tilde{X}_i$, x_i and m_i, respectively.

Since we have found a basis of all linearization equations and each linearization equation is a linear combination of this basis, this fact holds when the variables F_j in the equations are substituted by w_j'. Applying this fact to (3.1), we know

$$\begin{pmatrix} X_5 & X_6 \\ X_7 & X_8 \end{pmatrix} = \begin{pmatrix} X_4 & X_2 \\ X_3 & X_1 \end{pmatrix} \begin{pmatrix} Z_5' & Z_6' \\ Z_7' & Z_8' \end{pmatrix} Z_4'^{-1} \tag{3.6}$$

namely,

$$\begin{pmatrix} X_5 & X_6 \\ X_7 & X_8 \end{pmatrix} = \begin{pmatrix} (X_4Z_5' + X_2Z_7')Z_4'^{-1} & (X_2Z_8' + X_4Z_6')Z_4'^{-1} \\ (X_1Z_7' + X_3Z_5')Z_4'^{-1} & (X_1Z_8' + X_3Z_6')Z_4'^{-1} \end{pmatrix} \tag{3.7}$$

The linear equations in m_i derived from (3.6), (3.7) are all linear combinations of the equations $E_1', \cdots, E_l'$, in other words, (3.6), (3.7) holds on S.

Calculate the determinants of matrixes in two sides of matrix equation (3.6), then

$$X_5X_8 + X_6X_7 = C'Z_4 \tag{3.8}$$

where $C' = (Z_5'Z_8' + Z_6'Z_7')Z_4'^{-2}$.

Substitute (3.7) (3.8) into (2.2), then

$$\begin{cases} Z_1 = \tilde{X}_1\tilde{X}_3 + g_1(C'Z_4) \\ Z_2 = \tilde{X}_2\tilde{X}_1 + g_2(C'Z_4) \\ Z_3 = \tilde{X}_3\tilde{X}_2 + g_3(C'Z_4) \\ Z_4 = X_1X_4 + X_2X_3 \\ Z_5 = Z_5'Z_4'^{-1}Z_4 \\ Z_6 = Z_6'Z_4'^{-1}Z_4 \\ Z_7 = Z_7'Z_4'^{-1}Z_4 \\ Z_8 = Z_8'Z_4'^{-1}Z_4 \end{cases} \tag{3.9}$$

From the first three equations of (3.9), we can derive:

$$\begin{cases} \tilde{X}_3(Z_2 + g_2(C'Z_4)) = \tilde{X}_1(Z_3 + g_3(C'Z_4)) \\ \tilde{X}_2(Z_1 + g_1(C'Z_4)) = \tilde{X}_1(Z_3 + g_3(C'Z_4)) \end{cases} \tag{3.10}$$

Equation (3.10) implies that there exist at least 10 to 12 linearly independent linearization equations for remaining $45-l$ plaintext variables and the new public key polynomials, that is:

$$\sum_{i=1,j=1}^{45-l,50} \hat{a}_{ij} m_{u_i} \hat{F}_j + \sum_{i=1}^{45-l} \hat{b}_i m_{u_i} + \sum_{j=1}^{50} \hat{c}_j \hat{F}_j + \hat{d} = 0 \tag{3.11}$$

And these equations are still linearly independent when the value of $\hat{F}_i$ is given.

Additionally, from the last five equations, we find that some (at least 24) polynomials of the new public key polynomials can be linearly expressed by other polynomials.

In order to make our attack more efficient, we can do Gauss reduction first on the new public key polynomials. Note that here we must combine the given valid ciphertext with the new public key polynomials. Concretely, we consider the coefficients in each polynomial as a row vector and we concatenate w'_i and the coefficient vector corresponding to F_i. Therefore, we derive a $50 \times (\binom{45-l+2}{2}+1)$ matrix. Doing Gauss reduction on this matrix, we can obtain a matrix whose order less than 26. Hence, we derive a new set of public key polynomials, denoted by $\hat{\hat{F}}_i$. Set there are $t \leq 26$ polynomials in this set. Denote the valid ciphertext corresponding to the new public key polynomials by $\hat{w}'_i$, $i = 1, \cdots, t$.

So the equation (3.11) can be changed into:

$$\sum_{i=1,j=1}^{45-l,t} \hat{a}_{ij} m_{u_i} \hat{\hat{F}}_j + \sum_{i=1}^{45-l} \hat{b}_i m_{u_i} + \sum_{j=1}^{t} \hat{c}_j \hat{\hat{F}}_j + \hat{d} = 0 \tag{3.12}$$

To find all equations of the form (3.12), we can use the same method as the one used for equations (3.3). Firstly, we must derive a system of linear equation in $\hat{a}_{ij}$, $\hat{b}_i$, $\hat{c}_j$ and $\hat{d}$. Since the number of public key polynomials decrease to t, these equation have only

$$(45-l)t + 45 - l + t + 1 \leq 594$$

unknowns. We randomly select 600 $m \in S$, and substitute them in (3.12) to get a system of 600 linear equations and then solve it.

Let $\hat{D}$ and $\{(\hat{a}_{ij}^{(\rho)}, \hat{b}_i^{(\rho)}, \hat{c}_j^{(\rho)}, \hat{d}^{(\rho)}) : 1 \leq \rho \leq \hat{D}\}$ be the dimension and a basis of solution space, respectively. So we derive $\hat{D}$ linearly independent quadratic equations in $m_{u_i}, i = 1, \cdots, 45 - l$ and $\hat{\hat{F}}_j, j = 1, \cdots, t$. Then Substitute $\hat{\hat{F}}_j$ by $\hat{w}_j$ to get $\hat{D}$ linear equations in m_{u_i}. Assuming we can derive k ($k \geq 10$) linearly independent equations, denote these equations by $\hat{E}'_1, \cdots, \hat{E}'_k$. Doing a simple Gaussian elimination, from these k equations we can represent k variables of $m_{u_1}, \cdots, m_{u_{45-l}}$ by linear combinations of other $45 - l - k$. That is, we can find two disjoint subsets of $\{1, \ldots, 45 - l\}$, $B'_1 = \{v'_1, \cdots, v'_k\}$ and $B_1 = \{v_1, \cdots, v_{45-l-k}\}$, and linear expressions

$$m_{v'_j} = \hat{h}_j(m_{v_1}, \cdots, m_{v_{45-l-k}}), 1 \leq j \leq k \tag{3.13}$$

such that $\hat{E}'_1, \cdots, \hat{E}'_k$ holds when (3.13) are substituted into them. Let $\hat{S}$ denote $(45-l-k)$-dimensional affine subspace of S, where for each vector $(m_1, \cdots, m_{45})$ in $\hat{S}$, $m_{v'_j}$ is substituted by (3.13) for any $1 \leq i \leq k$.

Now substitute (3.13) into $F_j(m_1, \cdots, m_{45})$ and derive t new quadratic functions $\tilde{F}_j(m_{v_1}, \cdots, m_{v_{45-l-k}})$, $j = 1, \cdots, t$.

3.4 Third Elimination

Again, through theoretical analysis on $\tilde{F}_j(m_{v_1}, \cdots, m_{v_{45-l-k}})$, $j = 1, \cdots, t$, we find that we can do elimination once more.

Since we have found a basis of all linearization equations on S and each linearization equations is a linear combination of this basis, this fact of course holds when the variables $\hat{\hat{F}}_j$ in the equations are substituted by $\hat{w}'_j$. Applying this fact to (3.10), we know

$$\begin{cases} \tilde{X}_3 = \tilde{X}_1 C'_1 \\ \tilde{X}_2 = \tilde{X}_1 C'_2 \end{cases} \tag{3.14}$$

where $C'_1 = (Z'_3 + g_3(C'Z'_4))(Z'_2 + g_2(C'Z'_4))^{-1}, C'_2 = (Z'_3 + g_3(C'Z_4))(Z'_1 + g_1(C'Z'_4))^{-1}$, in other words, (3.14) holds on $\hat{S}$.

Substitute (3.14) into the first three equations of (3.9), then

$$\begin{cases} Z_1 = \tilde{X}_1^2 C'_1 + g_1(C'Z_4) \\ Z_2 = \tilde{X}_1^2 C'_2 + g_2(C'Z_4) \\ Z_3 = \tilde{X}_1^2 C'_1 C'_2 + g_3(C'Z_4) \end{cases} \tag{3.15}$$

We find $\tilde{X}_1^2$ can be expressed as linear combinations of the Z_i. Utilizing the fact that squaring is a linear operation on a field of characteristic 2, we have, on $\hat{S}$, the 6 expressions corresponding to $\tilde{X}_1^2$ is of the form $\sum a'_i m_i^2 + b'$ and $\mathbb{K}$-linear combinations of $F_j(m_1, \cdots, m_{45})$ and 1 (constant). Thus, of linear combinations of $\tilde{F}_j(m_{v_1}, \cdots, m_{v_{45-l-k}})$, $j = 1, \cdots, t$, there must exist at least 6 expressions which all contain only squaring terms and a constant term and correspond to $\tilde{X}_1^2$.

It is easy to solve the following linear system on the $\tilde{a}_i$ and $\tilde{b}_j$:

$$\begin{cases} \sum\limits_{i=1}^{50} \tilde{a}_i \tilde{F}_i(m_{v_1}, \cdots, m_{v_{45-l-k}}) + \sum\limits_{j=1}^{45-l-k} \tilde{b}_j m_{v_j}^2 + \tilde{c} = 0 \\ \forall m_{v_1}, \cdots, m_{v_{45-l-k}} \in \hat{W} \end{cases} \tag{3.16}$$

Set $(\tilde{a}_1^{(\rho)}, \cdots, \tilde{a}_{50}^{(\rho)}, \tilde{b}_1^{(\rho)}, \cdots, \tilde{b}_{45-l-k}^{(\rho)}, \tilde{c}^{(\rho)})$, $1 \le \rho \le p$ (where p such that $p + k = 15$, because the vectors in $\mathbb{K}^6$ corresponding to $\tilde{X}_i$ have 15 variables), is a basis of solution space of system (3.16). Set

$$\begin{cases} \sum\limits_{j=1}^{45-l-k} (\tilde{b}_j^{(\rho)})^{1/2} m_{v_j} + (\sum\limits_{i=1}^{50} \tilde{a}_i w'_i)^{1/2} + \tilde{c}^{(\rho)} = 0 \\ 1 \le \rho \le p \end{cases} \tag{3.17}$$

For any $(m_1, \cdots, m_{45}) \in \hat{S}$, its corresponding $(m_{v_1}, \cdots, m_{v_{45-l-k}})$ satisfied (3.17). Therefore we can represent p variables of $m_{v_1}, \cdots, m_{v_{45-l-k}}$ as linear expressions of the remaining variables.

So far, we represent totally $l + k + p$ variables of $(m_1, \cdots, m_{45})$ as linear expressions of the remaining $45-l-k-p$ variables. In other words, we eliminated $l + k + p$ variables in public key polynomials.

3.5 Finding The Plaintext

Substitute the linear expressions derived from (3.17) into $\tilde{F}_j(m_{v_1}, \cdots, m_{v_{45-l-k}})$, $j = 1, \cdots, t$ to get t new public key polynomials. There are $45 - l - k - p(\leq 6)$ in these new polynomials. Denote them by $\tilde{\tilde{F}}_j(m_{v_1}, \cdots, m_{v_{45-l-k}})$, $j = 1, \cdots, t$. Since $45 - l - k - p(\leq 6)$ is very small, in principle, we can use the Gröbner bases method or XL method to solve the system

$$\tilde{\tilde{F}}_j = \hat{w}'_j \tag{3.18}$$

very easily and to find the plaintext.

3.6 A Practical Attack Procedure, Its Complexity and Experimental Verification

Our attack can be further divide into the following five steps.

Step 1: *Find a basis of the linear space of the coefficient vectors* (a_{ij}, b_i, c_j, d) *of the linearization equations.*

As mentioned in subsection (3.1), we randomly select 2500 plaintexts $(m_1, \cdots, m_{45})$ and substitute them into equation (3.3) to get a linear system of 2500 equations on 2346 unknowns. The computational complexity to solve it is

$$2346^2 \times 2500 < 2500^3 < 2^{34}.$$

operations on the finite field $K = \mathbb{F}_{2^8}$.

This step is independent of the value of the ciphertext w' and can be done once for a given public key.

Our computer experiments show that indeed D is equal to 24.

Step 2: *For a given valid ciphertext* $(w'_1, \cdots, w'_{50})$, *we substitute it into (3.4) and solve the system of linear equations to get linear expression (3.5). Substitute (3.5) into the public key polynomials to derive a set of new public key polynomials* $\hat{F}_1, \cdots, \hat{F}_{50}$. *Then we combine the given valid cipheretext and the new public key polynomials and do Gauss reduction as subsection (3.3) described. At last, we derive* t *linearly independent public key polynomials and* t *new valid ciphertext components.*

The first part of this step is of computational complexity about

$$45^2 \cdot D < 45^3 < 2^{15},$$

and the second part is

$$\left(\binom{45 - l + 2}{2} + 1\right)^2 \times 50 < 2^{22}.$$

Our computer experiments show that the number of linear expression derived in this step is $l = 24$, and the number of the linearly independent public key polynomials is $t = 26$.

Step 3: *Solve (3.12) to get a basis of solution space of (3.12),* $\{(\hat{a}_{ij}^{(\rho)}, \hat{b}_i^{(\rho)}, \hat{c}_j^{(\rho)}, \hat{d}^{(\rho)}) : 1 \le \rho \le \hat{D}\}$ *then substitute the given ciphertext into result system of equations to derive linear expression (3.13).*

The first part of this step is of computational complexity about

$$(594)^2 \times 600 < 2^{24},$$

and the second part is

$$(45 - l)^2 \cdot \hat{D} < 2^{11}.$$

Our computer experiments show $\hat{D} = k = 12$.

Substituting (3.13) into $\hat{\hat{F}}_j, j = 1, \cdots, t$, we can derive a set of new public key polynomials $\tilde{F}_j(m_{v_1}, \cdots, m_{v_{45-l-k}}), j = 1, \cdots, t$.

Step 4: *Solve (3.16) to get a basis of solution space of it and then solve the system of equations (3.17) to derive p linear expressions in remainder* $45-l-k-p$ *components.*

The first part of this step is of computational complexity about

$$(96 - l - k)^3 < 2^{18},$$

and the second part is

$$p(45 - l - k)^2 < 2^{10}.$$

Our computer experiments show $p = 3$,

Step 5: *Use the Gröbner basis method to solve the system of equations (3.18) to get* $45 - l - k - p$ *values of plaintext components and then collect all linear expressions between the variables derived in previous steps to get the values of remainder plaintext components.*

Our computer experiments show that there is 6 variables and 8 polynomials in the last new public key polynomials $\tilde{\tilde{F}}_j(m_{v_1}, \cdots, m_{v_{45-l-k}})$, $j = 1, \cdots, t$. The computational complexity in this step is

$$\left(\frac{6^3}{3!}\right)^3 < 2^{18}.$$

Hence, the total computational complexity of our attack is less then 2^{34} $\mathbb{F}_{2^8}$-operations.

We implement our attack on a Pentium IV 2.4Ghz PC with 256M memory, and we code the attack using VC++. For any given valid ciphertext, our experiments successfully find the corresponding plaintext less than 7 minutes, where 6 minutes were spent on the execution of the step 1 in subsection (3.6), and less than 1 minute was spent to execute the remaining steps.

4 Conclusion

In this paper, we present a very efficient attack on TRMC-4. We need to do precomputation first, which takes 6 minutes on a PC with a 2.4Ghz Pentium

IV processor. Our attack then recovers the corresponding plaintext of any valid ciphertext in less than 1 minute. The total computational complexity is less than 2^{34} $\mathbb{F}_{2^8}$-operations. The key point of the attack is finding all linearization equations in polynomial time. Therefore, TRMC-4 is totally insecure.

Although we break the TRMC-4, we still think the design of TRMC is a interesting idea; one can carefully design the tractable rational map to improve the security of TRMC.

References

[DH03] J.Ding and T.Hodges. Cryptanalysis of an Implementation Scheme of TTM. *J. Algebra Appl.*, pages 273-282, 2004. http://eprint.iacr.org/2003/084.

[DS03] J.Ding and D.Schmidt. The new TTM implementation is not secure. In H.Niederreiter K.Q.Feng and C.P. Xing, editors, *Proceedings of International Workshop on Coding, Cryptography and Combinatorics (CCC 2003)*, pages 106–121, 2003.

[GC00] L.Goubin and N.Courtois. Cryptanalysis of the TTM cryptosystem. *LNCS, Springer Verlag*, 1976:44–57, 2000.

[JKMR05] A. Joux, S. Kunz-Jacques, F. Muller, P.-M. Ricordel, Cryptanalysis of the Tractable Rational Map Cryptosystem, *PKC 2005, p258-274, Lecture Notes in Computer Sciences* 3386.

[MI88] T.Matsumoto and H.Imai. Public quadratic polynomial-tuples for efficient signature verification and message encryption. In C.G. Guenther, editor, *Advances in cryptology –EUROCRYPT'88, LNCS*, volume 330, pages 419–453. Springer, 1988.

[Moh99] T.Moh. A fast public key system with signature and master key functions. *Lecture Notes at EE department of Stanford University*, May 1999. http://www.usdsi.com/ttm.html.

[NHLCD06] X.Nie, L.Hu, J.Li, C.Updegrove and J.Ding. Breaking A New Instance of TTM Cryptosystem. *Advances in ACNS2006, LNCS*, volume 3989, Springer, 2006.

[Pat95] J.Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In D. Coppersmith, editor, *Advances in Cryptology – Crypto'95, LNCS*, volume 963, pages 248–261, 1995.

[Sho97] P.Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing,* 26(5):1484-1509, October 1997.

[WC04] L.Wang and F.Chang. Tractable Rational Map Cryptosystem, available at http://eprint.iacr.org/2004/046, revised on February 3,2006.

[Wo05] C.Wolf, Multivariate Quadratic Polynomials in Public Key Cryptography, *Cryptology ePrint Archive, Report 2005/393*, 2005, avalable at http://eprint.iacr.org/.

[YC05] B.Yang and J.Chen. Building Secure Tame-like Multivariate Public key Cryptosystems–The New TTS. Information Security and Privacy: 10th Australasian Conference–ACISP 2005, LNCS 3574, 2005, Springer, P. 518-531.

Estimating the Prime-Factors of an RSA Modulus and an Extension of the Wiener Attack

Hung-Min Sun, Mu-En Wu, and Yao-Hsin Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 30013
hmsun@cs.nthu.edu.tw, {mn,saint_chen}@is.cs.nthu.edu.tw

Abstract. In the RSA system, balanced modulus N denotes a product of two large prime numbers p and q, where $q < p < 2q$. Since Integer-Factorization is difficult, p and q are simply estimated as $\sqrt{N}$. In the Wiener attack, $2\sqrt{N}$ is adopted to be the estimation of $p+q$ in order to raise the security boundary of private-exponent d. This work proposes a novel approach, called EPF, to determine the appropriate prime-factors of N. The estimated values are called "EPFs of N", and are denoted as p_E and q_E. Thus p_E and q_E can be adopted to estimate $p+q$ more accurately than by simply adopting $2\sqrt{N}$. In addition, we show that the Verheul and Tilborg's extension of the Wiener attack can be considered to be brute-guessing for the MSBs of $p+q$. Comparing with their work, EPF can extend the Wiener attack to reduce the cost of exhaustive-searching for $2r+8$ bits down to $2r-10$ bits, where r depends on N and the private key d. The security boundary of private-exponent d can be raised 9 bits again over Verheul and Tilborg's result.

Keywords: RSA, continued fraction, the Wiener attack, exhaustive-searching, most significant bit.

1 Introduction

RSA [7] has been conventionally adopted cryptosystem since 1978. The advantage of using RSA is that its security is based on the difficulty of Integer-Factorization. A 1024-bit RSA modulus N, which is a product of two 512-bit prime numbers, (*i.e.*, $N = pq$), is adopted to make the factoring infeasible. However, this system is inefficient for digital signature signing and verifying. Many practical issues have been considered in implementing issues, such as reducing the verifying and signing time [3], [8], [9]. In the real world, powerful computers such as servers are frequently employed to execute the verifying task. Lightweight devices with weak computational power, *e.g.*, smart card, wireless sensors or IC card, are employed to execute the signing task. Therefore, most research is focused on reducing the signing time rather than verifying time.

Since the complexity of signing depends on the bit-length of private-exponent, the most popular method to reduce the signing time is to apply a small private-exponent d. To achieve this purpose, a small private-exponent d is first chosen in the RSA key generation algorithm, and the corresponding public-exponent e satisfying $ed \equiv 1 \ (mod\phi(N))$ is then calculated. This RSA variant is called RSA-Small-d. However, RSA-Small-d also causes security problems [1], [4], [5], [11], [12]. Indeed, instances of RSA with $d < N^{1/4}$, can be efficiently broken by Wiener's continued fraction attack, which is called the Wiener attack [11]. Boneh and Durfee's lattice-based attack [2], which was proposed in 1998, indicates that the instance of RSA with $d < N^{0.292}$ should be considered as unsafe system. Although their attack is heuristic, it can work very well.

Verheul and Tilborg [10] proposed a technique to extend the Wiener attack in 1997. Their technique costs an exhaustive-searching for $2r + 8$ bits, where $r = \log_2 d - \log_2 N^{1/4}$ to raise r bits over the security boundary of the Wiener attack. Assume that brute-searching for 56 bits is feasible in terms of current computational ability. Solving r for the equation: $2r + 8 = 56$ yields $r = 24$. Therefore, the boundary of the Wiener attack can be raised 24 bits by Verheul and Tilborg's extension.

This work indicates that Verheul and Tilborg's extension can be considered as brute-guessing for the most significant bits (MSBs) of $p + q$, thus providing a motivation to study how to find out the MSBs of $p + q$ as many as possible. Consequently, this work develops an approach to estimate the appropriate prime-factors of N. Assume that the estimated prime-factors are termed p_E and q_E respectively. These terms p_E and q_E are called the "EPFs of N**"**, where EPF is short for "Estimated Prime-Factor". Using EPF, $p + q$ can be estimated more accurately than simply adopting $2\sqrt{N}$ as the estimated value.

Given a 1024-bit RSA modulus N, which is a product of two 512-bit prime numbers p and q, the values $p_E + q_E$ and $p + q$ generally match 10 to 12 MSBs. Therefore, if EPF is adopted to extend the Wiener attack, then the cost of exhaustive-searching for $2r + 8$ bits is reduced to that of exhaustive-searching for $2r - 10$ bits. Consequently, the security boundary of private-exponent d can be raised 9 bits again over that of Verheul and Tilborg's extension.

The remainder of this paper is organized as follows: Section 2 briefly reviews some basic results used in the paper, including continued fraction, the Wiener attack, and Verheul & Tilborg's extension. Section 3 then proposes the approach of EPF and shows the experiment results. Next, Section 4 gives another look on Verheul and Tilborg's extension and applies EPF to improve its performance. Conclusions are finally drawn in Section 5, along with recommendations for future work.

1.1 Our Contribution

The contributions of this work are listed in the following:

(1) A novel approach, called EPF, is provided to evaluate the appropriate prime-factors of N.
(2) Verheul and Tilborg's extension of the Wiener attack is considered as brute-guessing for the MSBs of $p + q$.
(3) Combine the results of (1) and (2), the exhaustive-searching for the extension of the Wiener attack can be reduced from $2r + 8$ bits to $2r - 10$ bits.

2 Preliminary

Some background knowledge is reviewed in this section, including continued fractions, the Wiener attack, and the Verheul and Tilborg's extension.

2.1 Continued Fractions

First we give the definition of continued fractions and some related theorems. The details can be referenced in [6].

Definition 1. *For any poistive real number ξ_0, define $a_i = \lfloor \xi_i \rfloor$, $\xi_{i+1} = 1/(\xi_i - a_i)$ for $i = 0, 1, 2, ..., n$, until ξ_n is an integer. Then ξ_0 can be expanded into the following form:*

$$\xi_0 = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + ... + \cfrac{1}{a_n}}} \tag{1}$$

(1) is called the continued fraction expression of ξ_0. For simplicity, we write (1) to be $\xi_0 = (a_0, a_1, a_2, ..., a_n)$. Besides, $(a_0, a_1, ..., a_i)$ is denoted as the i'th convergent of the continued fraction expansion of ξ_0.

Theorem 2. *If ξ_0 is a rational number, then the process of calculating continued fraction expression would be finished in some finite index n. Otherwise, if ξ_0 is an irrational number, the process would not stop and n is approaching to infinite.*

Theorem 3. *For any positive real number ξ_0, suppose $\frac{h_n}{k_n}$ is the i'th convergent of the continued fraction expression of ξ_0. Define $h_{-2} = 0$, $h_{-1} = 1$; $k_{-2} = 1$, $k_{-1} = 0$, then $h_i = a_i h_{i-1} + h_{i-2}$ and $k_i = a_i k_{i-1} + k_{i-2}$ for $i \geqq 0$.*

Theorem 4. *The convergents $\frac{h_n}{k_n}$ are successively close to ξ_0, that is*

$$\left|\xi_0 - \frac{h_n}{k_n}\right| < \left|\xi_0 - \frac{h_{n+1}}{k_{n+1}}\right| .$$

Furthermore, if ξ_0 is an irrational number, then $\lim\limits_{n\to\infty} \frac{h_i}{k_i} = \xi_0$.

Theorem 5. *Let ξ_0 denote any real number. If there is a rational number $\frac{a}{b}$ with $1 \le b$ satisfying*

$$\left|\xi_0 - \frac{a}{b}\right| < \frac{1}{2b^2} ,$$

then $\frac{a}{b}$ is one of the convergents of the continued fraction expression of ξ_0.

2.2 The Wiener Attack

Wiener [11] first applied the technique of continued fraction to attack RSA-Small-d. He observed that RSA equation $ed = k\varphi(N) + 1$ can be rewritten as the following form:

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \left|\frac{1}{d\varphi(N)}\right|. \tag{2}$$

Replacing $\frac{e}{\varphi(N)}$ in (2) by $\frac{e}{N}$ yields

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{2d^2}. \tag{3}$$

According to Theorem 5, if (3) is hold, then $\frac{k}{d}$ equals one of the convergents of the continued fraction expression of $\frac{e}{N}$. Since $\gcd(k, d) = 1$, the values of d and k can be extract out actually. Since $N^{1/2} \approx p \approx q$ and $d \approx k$, the left side of (3) reduces to

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \frac{Nk - ed}{Nd} = \frac{k\,(p + q - 1) - 1}{Nd} \approx \frac{2}{N^{1/2}}. \tag{4}$$

In order to apply Theorem 5 again, we have to set

$$\frac{2}{N^{1/2}} < \frac{1}{2d^2}\ ,$$

which leads to:

$$d < \tfrac{1}{2}N^{1/4}. \tag{5}$$

After ignoring the small constatnt $\frac{1}{2}$ in (5), the Wiener attack shows that RSA is insecure when the private-exponent d is smaller than $N^{1/4}$. For instance of 1024-bit RSA modulus, d should be chosen larger than 256 bits.

2.3 Verheul and Tilborg's Extension of the Wiener Attack

While considering the private-exponent d which is slightly larger than $N^{1/4}$, the Wiener attack would be failed. Hence, in order to avoid this situation, Verheul and Tilborg [10] propose a technique to raise the security boundary of $N^{1/4}$ with exhaustive-searching for $2r+8$ bits, where $r = \log_2 d - \log_2 N^{1/4}$. They consider the following identity:

$$\frac{k}{d} = \frac{p_{j+1}U + (U\varDelta + V)\,p_j}{q_{j+1}U + (U\varDelta + V)\,q_j}, \tag{6}$$

where $\frac{p_i}{q_i}$ is the i'th convergent of the continued fraction of $\frac{e}{N}$. "U" and "V" are unknown numbers with upper bound: $\log_2 U \le r + 4$ and $\log_2 V \le r + 4$ respectively. The item "$\varDelta$" is a small number, *e.g.*, 1 or 2., thus we omit its uncertainty. Consequently, the uncertainty of $\frac{k}{d}$ in (6) is about $2r+8$ bits, which means we need to do an exhaustive-searching for about $2r + 8$ bits to extract out the correct value of $\frac{k}{d}$.

Assume that brute-guessing a number with quantity 2^{56} is feasible in terms of current computational ability. Solving r for $2r + 8 = 56$ yields the boundary of Verheul and Tilborg's result. That is, Verheul and Tilborg's extension can further extend the security boundary of d up to 24 bits over Wiener's result. Thus, the instance of RSA with $d < N^{1/4}2^{24}$ can be totally broken by the technique of continued fraction. In this paper, we show Verheul and Tilborg's extension can be regarded as brute-guessing the MSBs of $p+q$. Furthermore, we reduce the cost of original exhaustive-searching for $2r + 8$ bits to $2r - 10$ bits, where $r = \log_2 d - \log_2 N^{1/4}$.

3 The Proposed Approach (EPF) to Estimate the Prime-Factors of $N = pq$

In this section, a novel approach, called EPF, to estimate the prime-factors of N $(= pq)$ is proposed. The point is to find out two numbers, p_E and q_E, by imitating the properties of p and q as similar as possible. The properties includes the bit-length, the most significant bits, and the product of p_E and q_E. Also, we name p_E and q_E "EPFs of N", where EPFs is short for "Estimated Prime-Factors".

First, we focus on how to estimate $p + q$. Note that the hardness of finding $p + q$ is the same as the hardness of finding p and q due to the formula:

$$(p - q)^2 = (p + q)^2 - 4N. \tag{7}$$

Thus solving p and q is obvious by computing $p - q$ with the formula (7).

3.1 How to Estimate $p + q$?

Suppose $N = pq$, where p and q are two large prime-numbers with the same bit-length. Without loss of generality, we assume that $q < p < 2q$. Define D_p to be the difference of $\sqrt{N}$ and p. Similarly, define D_q to be the difference of q and $\sqrt{N}$. That is,

$$p = \sqrt{N} + D_p \text{ and } q = \sqrt{N} - D_q \tag{8}$$

Applying (8) to $N = pq$ we have

$$N = pq = (\sqrt{N} + D_p)(\sqrt{N} - D_q) = N + \sqrt{N}(D_p - D_q) - D_pD_q \tag{9}$$

After simplifying (9) yields

$$D_pD_q = \sqrt{N}(D_p - D_q) \tag{10}$$

Dividing by $\sqrt{N}D_pD_q$ in both sides of (10) leads to

$$\frac{1}{\sqrt{N}} = \frac{D_p - D_q}{D_pD_q}. \tag{11}$$

To estimate the appropriate quantities of D_p and D_q, we compute the i'th convergent of $1/\sqrt{N}$, denoted h_i/k_i, in the continued fraction expression. Hence, according to Theorem 4, $\{h_i/k_i\}_i$ is a rational sequence such that

$$\frac{h_i}{k_i} \to \frac{1}{\sqrt{N}} = \frac{D_p - D_q}{D_p D_q}, \text{ as } i \to \infty. \tag{12}$$

Since $\sqrt{N}$ must be an irrational number, or we can factor N immediately, the three values $1/\sqrt{N}$, $D_p - D_q$ and $D_p D_q$ in (12) are irrational numbers as well. Due to the reason of inconvenience for operations on irrational numbers, we consider the rational number $\frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor}$, which is close to $\frac{D_p - D_q}{D_p D_q}$. The integer parts of $D_p - D_q$ and $D_p D_q$ are almost the same as the integer parts of $\lceil D_p \rceil - \lfloor D_q \rfloor$ and $\lfloor D_p D_q \rfloor$ respectively. Setting $p = \left\lfloor \sqrt{N} \right\rfloor + \lceil D_p \rceil$ and $q = \left\lfloor \sqrt{N} \right\rfloor - \lfloor D_q \rfloor$, we have

$$p + q = 2 \left\lfloor \sqrt{N} \right\rfloor + \lceil D_p \rceil - \lfloor D_q \rfloor . \tag{13}$$

According to (13), we know that the information of $\lceil D_p \rceil - \lfloor D_q \rfloor$ is still uesful for us to estimate $p + q$.

Next, Theorem 6 shows that $\frac{h_n}{k_n}$ and $\frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor}$ are quite near. This implies that adopting $\frac{h_n}{k_n}$ to be the estimation of $\frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor}$ is reasonable. Moreover, we give the upper bound of the difference between $\frac{h_n}{k_n}$ and $\frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor}$.

Theorem 6. *If $k_n < D_p D_q$, we have*

$$\left| \frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor} - \frac{h_n}{k_n} \right| < \frac{3}{\lfloor D_p D_q \rfloor}. \tag{14}$$

Proof. Since $k_n < D_p D_q$ and $k_n < k_n k_{n+1}$, we have

$$\frac{1}{k_n k_{n+1}} < \frac{1}{k_n} < \frac{1}{D_p D_q} < \frac{1}{\lfloor D_p D_q \rfloor}.$$

Now we prove (14) by triangle inequality:

$$\left| \frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor} - \frac{h_n}{k_n} \right| = \left| \frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor} - \frac{D_p - D_q}{D_p D_q} + \frac{D_p - D_q}{D_p D_q} - \frac{h_n}{k_n} \right|$$

$$\leqq \left| \frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor} - \frac{D_p - D_q}{D_p D_q} \right| + \left| \frac{D_p - D_q}{D_p D_q} - \frac{h_n}{k_n} \right| < \frac{2}{\lfloor D_p D_q \rfloor} + \frac{1}{k_n k_{n+1}} < \frac{3}{\lfloor D_p D_q \rfloor}.$$

Done. ■

Since $\lfloor D_p D_q \rfloor$ is much larger than 3, $\frac{3}{\lfloor D_p D_q \rfloor}$ is close to 0. Consequently, the value of $\frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor}$ is almost the same as the vaule of $\frac{h_n}{k_n}$. Also, if the bit-length of h_n is equal to or slightly smaller than the bit-length of $\lceil D_p \rceil - \lfloor D_q \rfloor$, h_n may be considered as the estimation of $\lceil D_p \rceil - \lfloor D_q \rfloor$ reasonably. Hence, to select the suitable index n, we apply the following rule:

$$h_n < \lceil D_p \rceil - \lfloor D_q \rfloor < h_{n+1}$$

Notice that h_n is smaller than h_{n+1} due to Theorem 3. An apparent question is that how to choose the right value of n without the information of $\lceil D_p \rceil - \lfloor D_q \rfloor$. To solve this problem, in fact, we choose several index as candidates according to the statistical result. The experiment shows the average of n is 299, with the standard deviation 12. Therefore, while searching the right value of n for each modulus N, it may increase a little complexity. Here we simply estimate the appropriate quantity which is slightly smaller than the vaule of $\lceil D_p \rceil - \lfloor D_q \rfloor$. Thus we choose h_n as estimated value rather than h_{n+1}. However, it has no theory to justify the difference of bit-lengths of h_n and $\lceil D_p \rceil - \lfloor D_q \rfloor$. Thus we show that the bit-length of h_n is actually slightly smaller than the bit-length of $\lceil D_p \rceil - \lfloor D_q \rfloor$ by implementing experiments. Table 1 gives the results for 1024-bit and 2048-bit RSA modulus respectively. We take 100 instances for each case and compute the average bit-length and its standard deviation. According to our experiments, for 1024-bit RSA modulus, h_n is about 502 bits with standard deviation 2.01. As for 2048-bit RSA modulus, h_n is about 1011 bits with standard deviation 4.42.

Table 1. The Bit-lengths of Estimated and Real Values

Modulus N	h_n	$\lceil D_p \rceil - \lfloor D_q \rfloor$	h_{n+1}
1024 bits	502 bits	503 bits	505 bits
Standard deviation	2.01	1.43	2.10
2048 bits	1011 bits	1012 bits	1013 bits
Standard deviation	4.42	4.19	4.40

3.2 Estimated Prime-Factors of N (EPFs of N)

Here we show how to estimate the prime-factors of N, where $N = pq$. The estimated prime-factors are denoted as p_E and q_E, which are called "EPFs of N". Since $\lfloor D_p D_q \rfloor \approx \lceil D_p \rceil \cdot \lfloor D_q \rfloor$, k_n can be regarded as the estimation of $\lceil D_p \rceil \cdot \lfloor D_q \rfloor$. In other words, we have three fractions,

$$\frac{h_n}{k_n} \approx \frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lfloor D_p D_q \rfloor} \approx \frac{\lceil D_p \rceil - \lfloor D_q \rfloor}{\lceil D_p \rceil \cdot \lfloor D_q \rfloor} ,$$

which are all close to each other. Besides, the bit-lengths of their numerator and denominator are almost the same. Hence, h_n and k_n can be regarded as the estimations of $\lceil D_p \rceil - \lfloor D_q \rfloor$ and $\lceil D_p \rceil \cdot \lfloor D_q \rfloor$, that is,

$$h_n \approx \lceil D_p \rceil - \lfloor D_q \rfloor \text{ and } k_n \approx \lceil D_p \rceil \cdot \lfloor D_q \rfloor$$

Computing $\lceil D_p \rceil + \lfloor D_q \rfloor$ by the formula:

$$\begin{aligned} (\lceil D_p \rceil + \lfloor D_q \rfloor)^2 &= (\lceil D_p \rceil - \lfloor D_q \rfloor)^2 + 4 \lceil D_p \rceil \cdot \lfloor D_q \rfloor \\ &= h_n^2 + 4k_n \end{aligned} \tag{15}$$

Solving $\lceil D_p \rceil$ and $\lfloor D_q \rfloor$ from (15) we get

$$\lceil D_p \rceil = \left\lceil \frac{\sqrt{h_n^2+4k_n}+h_n}{2} \right\rceil \text{ and } \lfloor D_q \rfloor = \left\lfloor \frac{\sqrt{h_n^2+4k_n}-h_n}{2} \right\rfloor$$

Therefore, the EPFs of N are set to

$$p_E = \left\lfloor \sqrt{N} \right\rfloor + \left\lceil \frac{\sqrt{h_n^2+4k_n}+h_n}{2} \right\rceil \text{ and } q_E = \left\lfloor \sqrt{N} \right\rfloor - \left\lfloor \frac{\sqrt{h_n^2+4k_n}-h_n}{2} \right\rfloor .$$

3.3 The Accuracy of EPFs of N

Here we show the accuracy of EPFs of N by experiments. The statistical data in Table 2 shows the difference of EPFs and practical prime-factors, *i,e,*, $p - p_E$, $q_E - q$. We also compute the average bit-length of $N - N_E$, where $N_E = p_E q_E$. The data comes from the average of 100 samples for each case. Note that the p_E and q_E are the same 7 MSBs with p and q respectively for 1024-bit RSA modulus. Also, for the case of 2048-bit RSA modulus, p_E and q_E are the same 9 MSBs with p and q respectively.

Table 2. The Accuracy of EPFs

the bit-length of N	1024 bits	2048
the average bit-length of $p - p_E$	505 bits	1015 bits
standard deviation of $p - p_E$	1.52	2.57
the average bit-length of $q_E - q$	505 bits	1015 bits
standard deviation of $q_E - q$	1.49	2.57
the average bit-length of $N - N_E$	510 bits	1022 bits
standard deviation	1.56	2.06

4 Another Look on Verheul and Tilborg's Extension and Its Improvement

We show that Verheul and Tilborg's extension can be regarded as brute-guessing for the MSBs of $p + q$ in this section. By applying EPF to improve the Wiener attack, the new result raises the security boundary of d again. In the remainder of the paper, we suppose the estimation of $p + q$ is $2A$, *i.e.*, $A \approx \frac{p+q}{2}$. Under such assumption $\phi(N) = (N + 1) - (p + q)$ is estimated as $(N + 1) - 2A$.

4.1 Improvement of the Wiener Attack

Consider the following question:

Question:
When considering the RSA equation: $ed = k(p-1)(q-1)+1$, where $N = pq$, what range of d would satisfy the following inequalities?

$$|\frac{e}{N+1-2A} - \frac{k}{d}| < \frac{1}{2d^2} < |\frac{e}{N} - \frac{k}{d}| \tag{16}$$

The meaning of (16) is shown as follows: In the right side of (16), the inequality means the range of d that the Wiener attack would fail. Instead, in the left side of (16), the inequality means the the Wiener attack can work successfully. The difference between left inequality and right inequality of (16) is whether applying new estimation of $\phi(N)$, *i.e.*, $N+1-2A$, to replace N. Now we simplify (16) in the following:

Consider the right side of (16), since

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \frac{ed-Nk}{Nd} = \frac{k(p+q-1)-1}{Nd} \ ,$$

the right side inequality of (16) is equivalent to $\frac{1}{2d^2} < \frac{k(p+q-1)-1}{Nd}$, that is

$$N < 2dk[(p+q)-1] - 2d. \tag{17}$$

Similarly, in the left side of (16), since

$$\left|\frac{e}{N+1-2A} - \frac{k}{d}\right| = \left|\frac{ed-k(N+1-2A)}{(N+1-2A)d}\right| = \frac{k[(p+q)-2A]-1}{(N+1-2A)d} \ ,$$

the left side inequality of (16) is equivalent to $\frac{k[(p+q)-2A]-1}{(N+1-2A)d} < \frac{1}{2d^2}$, that is

$$2dk[(p+q)-2A] - 2d < N+1-2A. \tag{18}$$

In order to combine (17)and (18), we rearrange (18) in the following form:

$$2dk[(p+q)-1] - 2d < N + (2dk-1)(2A-1) \ , \tag{19}$$

which is the same format of (17). Consequently, after combining (17) and (19), (16) is equivalent to

$$N < 2dk[(p+q)-1] - 2d < N + (2dk-1)(2A-1). \tag{20}$$

Note that if $A = \frac{p+q}{2}$, the right side of (20) changes to

$$2dk\,(p+q-1) - 2d < N + (2dk-1)\,((p+q)-1) = 2dk\,(p+q-1) + \varphi(N) \ ,$$

which is always hold for any size of private-exponent d.
Solving d in the right inequality of (20) we get its upper bound:

$$d < \frac{N+1-2A}{2k\,(p+q-2A)-2}. \tag{21}$$

According to (21), the private-exponent d should not be chosen smaller than $\frac{N+1-2A}{2k(p+q-2A)-2}$ or RSA system can be totally broken immediately. In addition, the closer the distance between $2A$ and $p+q$ is, the larger upper bound the insecure private-exponent is. Therefore, to raise the security boundary of d, we should try to find the estimated value of $p+q$ as appropriate as possible. This conclusion also implies that the complexity of extending the Wiener attack can be considered as the complexity of finding $p+q$.

4.2 Applying EPF to the Proposed Extension of the Wiener Attack

Now we analyze how much complexity could be reduced when we apply EPF to the proposed extension of the Wiener attack. Define a variable "Λ" to denote the difference of $\frac{p+q}{2}$ and A, *i.e.*, $\Lambda = \frac{p+q}{2} - A$. Note that A is the esitmated value of $\frac{p+q}{2}$, thus Λ is represented the uncertainty part of $\frac{p+q}{2}$. Replacing A by $\frac{p+q}{2} - \Lambda$ into the right inequality of (20) yields

$$\begin{aligned} &2dk(p+q-1) - 2d \\ &< N + (2dk-1)\left(2\left(\tfrac{p+q}{2} - \Lambda\right) - 1\right) \\ &= 2dk(p+q-1) + \varphi(N) - 2\Lambda(2dk-1) \end{aligned} \tag{22}$$

Eliminating $2dk(p+q-1)$ in both sides of (22), we have

$$2\Lambda(2dk-1) - 2d < \varphi(N). \tag{23}$$

According to (22), we have the following conclusion: The parameters Λ, k, and d in (23) play the main role to determine whether the Wiener attack can work or not. Since d and k are pre-determined parameters in the key-generation of RSA, the only variable we could control is the parameter Λ which represents the uncertainty part of $\frac{p+q}{2}$. This implies the more accuracy A is estimated, the smaller quantity of Λ will be. Therefore, to raise the security boundary of private-exponent d, we should focus our effort on finding out the MSBs of $p+q$ as many as possible. In the following, Table 3 gives the experiment results about how many MSBs of $p+q$ that EPF can be found out.

The statistics data in Table 3 comes from the averages of computing 100 instances for $\frac{p+q}{2}$, A and Λ. Note that $2A$ is the estimation of $p+q$. Thus $2A$ is set to be $2\left\lfloor\sqrt{N}\right\rfloor + h_n$ according to EPF.

Table 3. The Difference between Estimated and Real Values

Modulus N	$\frac{p+q}{2}$	$A = \frac{2\lfloor\sqrt{N}\rfloor + h_n}{2}$	$\Lambda = \frac{p+q}{2} - A$
1024 bits	512 bits	512 bits	500 bits
Standard deviation	0	0	1.89
2048 bits	1024 bits	1024 bits	1009 bits
Standard deviation	0	0	4.03

In Table 3, for the case of 1024-bit N, the average bit-length of Λ is 500 bits with the standard deviation 1.89. This implies that $\frac{p+q}{2}$ and A usually match in 10 MSBs at least, where 10 is computed from $512 - \lceil 500 + 1.89 \rceil$. For the case of 2048-bit RSA modulus, the average bit-length of Λ is 1009 bits with the standard deviation 4.03. Thus, $\frac{p+q}{2}$ and A usually also match in 10 MSBs at least, where 10 is computed from $1024 - \lceil 1009 + 4.03 \rceil$.

4.3 Better Result Compared with Verheul and Tilborg's Extension

We compare our improvement with Verheul and Tilborg's result. Consider the case of 1024-bit RSA modulus N, in order to further reduce the quantity of Λ, we do an exhaustive-searching for finding out the s MSBs of Λ and write $\Lambda = (2^{500-s})\Lambda_1 + \Lambda_2$, where $\Lambda_1 \in \left[2^{s-1}, 2^s\right]$, and $\Lambda_2 \in \left[2^{500-s}, 2^{501-s}\right]$. Suppose that Λ_1 can be totally gotten by exhaustive-searching and Λ_2 is still an unknown part. Under such assumption, $\frac{p+q}{2}$ can be estimated as $A + (2^{500-s})\Lambda_1$ more accurately instead of just estimating as A. In addition, the values of $\frac{p+q}{2}$ and $A + (2^{500-s})\Lambda_1$ usually match $12 + s$ MSBs from tha above result. Thus the uncertainty part of $\frac{p+q}{2}$, Λ_2, remains $500 - s$ bits.

Now we analyze how much improvement after adopting brute-guessing for Λ_1. Applying uncertainty part of $\frac{p+q}{2}$, Λ_2, to (23) yields

$$2\Lambda_2(2dk-1) - 2d < \varphi(N). \tag{24}$$

In order to satisfy (24), we should compute the bit-length of each side. Denote $|d|$ and $|k|$ to be represented the bit-length of d and k respectively, thus the bit-length of $2\Lambda_2(2dk-1) - 2d$ is

$$1 + (500 - s) + 1 + |d| + |k|$$

which is mainly determined by $2\Lambda_2(2dk-1)$ in (24). Furthermore, to satisfy (24) we have to set

$$1 + (500 - s) + 1 + |d| + |k| < 1024 \tag{25}$$

where 1024 is the bit-length of $\varphi(N)$ in (24). Since the bit-length of d and k are almost the same with high probability in the key-generation algorithm of RSA-Small-d, we can assume $|d| = |k|$. Suppose that $|d| = |k| = 256 + r$. *i.e.*, the private-exponent d exceeds 256 bits ($N^{1/4}$) more r bits. Applying $256 + r$.to (25) we get

$$1 + (500 - s) + (1 + 2(256 + r) < 1024 \ ,$$

which is equivalent to

$$2r - 10 < s. \tag{26}$$

By (26), we have a conclusion which is simlilar to Verheul and Tilborg's result: To extend the Wiener's boundary r bits, we only have to do an exhaustive-searching for about $2r - 10$ bits, where $r = \log_2 d - \log_2 N^{1/4}$. Compared with

Verheul and Tilborg's result [10], which costs an exhaustive-searching for $2r+8$ bits, our result is 18 bits fewer than Verheul and Tilborg's. Thus it is more efficient to applying our method on the extension of the Wiener attack.

Suppose that the complexity that the current computer can work with is under $O(2^{56})$. This means brute-searching for any number whose bit-length less than 56 is feasible. Verheul and Tilborg's extension can attack successfully on $d < N^{1/4}2^{24}$, where 24 comes from by solving r for $2r+8 = 56$. With our result in (26): Solving r for $2r-10 = 56$ yields $r = 33$. Hence, the proposed method can attack successfully on $d < N^{1/4}2^{33}$, which is more 9 bits than Verheul and Tilborg's result.

Table 4 shows the comparisons between the original Wiener attack, Verheul and Tilborg's extension (V-T Extension), and our improvement.

Table 4. The comparison between each attack

	Upper Bound of d	Complexity
The Wiener Attack	$d < N^{1/4}$	Polynomial time
V-T Extension	$d < N^{1/4}2^{24}$	exhaustive-searching for $2r+8$ bits
Our Improvement	$d < N^{1/4}2^{33}$	exhaustive-searching for $2r-10$ bits

5 Conclusion

This work presents a novel approach, called EPF, to determine the estimated prime-factors of N through the continued fractions. Experiment results shows that the 12 MSBs of $p+q$ can be estimated correctly for the 1024-bit N. This technique reduces the error between the real and estimated $\varphi(N)$, and raises the security boundary of private-exponent d. Besides, We show a result that Verheul and Tilborg's extension of the Wiener attack can be consider as brute-guessing for MSBs of $p+q$. By applying EPF to the proposed result, the security boundary of d can be raised again. Assuming that exhaustive-searching for 56 bits is feasible, Verheul and Tilborg's extension raises 24 bits over the Wiener's boundary. The proposed method raises 9 bits over the Verheul and Tilborg's boundary. Therefore, the instance of RSA with $d < N^{1/4}2^{33}$ can be totally broken by the technique of the continued fractions.

An open problem has been mentioned many times in the past research. Whether exists a better method to evaluate the estimated value of $\varphi(N)$? The boundary of the Wiener attack can be raised again as the accuracy of the estimate of $\varphi(N)$. In the futrue, we will try to design an efficient and accurate method based on the continued fractions and other mathematic materials.

Acknowledgments. This research was supported in part by the National Science Council, Taiwan, under contract NSC95-2221-E-007-030. We are grateful to anonymous reviewers for their valuable comments.

References

[1] D. Boneh, “Twenty Years Attacks on the RSA Cryptosystem”, Notices of the American Mathematical Society, Vol. 46, No. 2, pp. 203-213, 1999.

[2] D. Boneh and G. Durfee, “Cryptanalysis of RSA with private key d less than $N^{0.292}$”, *IEEE Trans. on Information Theory*, Vol. 46, No. 4, pp. 1339-1349, 2000.

[3] D. Boneh and H. Shacham, “Fast Variants of RSA”, CryptoBytes, Vol. 5, No. 1, Springer, 2002.

[4] G. Durfee, P. Q. Nguyen, “Cryptanalysis of the RSA Schemes with Short Secret Exponent form Asiacrypt ’99”, Proceedings of Cryptology - ASIACRYPT’00, LNCS 1976, Springer-Verlag, pp.1-11, 2000.

[5] H.-S. Hong, H.-K. Lee, H.-S. Lee and H.-J. Lee, “The better bound of private key in RSA with unbalanced primes”, *Applied Mathematics and Computation*, Vol. 139, pp. 351-362, 2003.

[6] I. Niven, H. S. Zuckerman, *An Introduction to the Theory of Number*, John Wiley and Sons Inc,1991.

[7] R. Rivest, A. Shamir and L. Aldeman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, *Communications of the ACM* , Vol. 21, No.2, pp.120-126, 1978.

[8] H.-M. Sun, W.-C. Yang and C.-S. Laih, “On the Design of RSA with Short secret-exponent”, Proceedings of Cryptology - ASIACRYPT’99, LNCS 1716, Springer-Verlag, pp.150-164, 1999.

[9] H.-M. Sun and C.-T. Yang, ”RSA with Balanced Short Exponents and Its Application to Entity Authentication”, Proceeding of Public Key Cryptography 05 - PKC’05, LNCS 3386, Springer-Verlag, pp.199-215, 2005.

[10] E. Verheul and H. van Tilborg, “Cryptanalysis of less short RSA secret-exponents”, *Applicable Algebra in Engineering, Communication and Computing*, Vol. 8, Springer-Verlag, pp. 425-435, 1997.

[11] M. J. Wiener, “Cryptanalysis of RSA with short secret-exponents”, *IEEE Trans. on Information Theory*, Vol. 36, pp.553-558, 1990.

[12] B. de Weger, “Cryptanalysis of RSA with small prime difference”, *Applicable Algebra in Engineering, Communication and Computing*, Vol. 13, pp. 17-28, 2002.

A Timing Attack on Blakley's Modular Multiplication Algorithm, and Applications to DSA

Bahador Bakhshi and Babak Sadeghiyan

Computer Engineering and Information Technology Department
Amirkabir University of Technology, Tehran, Iran
bbakhshi@aut.ac.ir, basadegh@ce.aut.ac.ir

Abstract. In this paper, we introduce a timing attack scheme against a 160-bit modular multiplication with Blakley's algorithm. It is assumed that a set of public inputs are multiplied by a secret parameter and running time of each multiplication is given, but the multiplication result is not known and a machine similar to victim machine isn't available. The proposed attack extracts all 160 bits of the secret parameter. Running time of Blakley's algorithm is analyzed and it is shown that running time of each step is dependent on the running time of other steps. The dependencies make the parameters of the attack be dependent on the secret key, while it makes the attack rather complicated. A heuristic algorithm is used to find the parameters of the attack. As a real scenario, the attack is applied against on-line implementation of Digital Signature Algorithm, which employs Blakley's modular multiplication. Practical results show that secret key of DSA will be found using 1,000,000 timing samples.

Keywords: timing attack, modular multiplication, Blakley's algorithm, DSA.

1 Introduction

Any cryptographic primitive, such as a digital signature, can be considered in two different aspects. It can be viewed as an abstract mathematical function that takes some inputs and produces outputs. Alternatively, it can be viewed as an implementation of a mathematical function in the real-world software/hardware system. In the latter view, the cryptographic system interacts with environment through side channels, such as power consumption channel and execution time channel. Side channel attacks use the leaked data from the side channels to attack on a certain cryptographic system, while make some assumptions about the implementation. Among different side channel attacks, the timing attack has special feature, i.e. very limited equipment is required to gather timing data.

Idea of timing attack was first introduced publicly by Kocher in [10]. He showed that difference between the required execution times for various inputs, can be exploited in order to find secret parameters of the underlying system.

Kocher mentioned some systems, which maybe vulnerable to timing attack, including RSA and DSA. He also showed how RSA implementation, which employs square-and-multiply algorithm to implement modular exponentiation is vulnerable to timing attack. Dhem et al. described a practical timing attack on an RSA implementation [4]. These attack schemes are not applicable to CRT based implementations of RSA. In [16], Schindler proposed another timing attack scheme on RSA implementation based on CRT method and Montgomery multiplication. Contrary to Kocher's and Dhem's attacks, Schindler's attack does not directly find the secret key. His attack factorizes RSA-modulus, instead. Schindler also introduced advance statistical and stochastic method to model and optimize timing attack in [17,18]. Brumley and Boneh employed and improved Schindler's idea to attack remotely on the RSA implementation which is used in OpenSSL library [2]. They showed that not only smart cards but also general purpose applications such as cryptographic operations in network communication and operating systems are vulnerable to timing attack. All these attacks are based on timing vulnerability of modular exponentiation, in which the exponent and modulus are constant, but a different base is used in each exponentiation.

Another timing attack, based on the vulnerability of modular exponentiation, was introduced against GPS identification system in [3]. Differing from previous attacks, the exponent isn't constant in this attack. GPS uses two secret keys, short-term key which is generated in each execution and a long-term key that is permanent for each user. The short-term key is used in a modular exponentiation and in a modular addition with the long-term key. In the attack, timing samples are used to find out only the hamming weight of the short-term key. Using inputs and outputs of GPS algorithm and the hamming weight, bits of the long-term key are guessed.

In addition to RSA and GPS, several block ciphers have been also examined under timing attack such as DES [8], RC5 [7] , Rijndael [11] . Timing analysis has been also applied on web privacy [5], in which a malicious web site can determine, using response of browser to the request, whether or not the user has recently visited some other, unrelated web page. The attack on web privacy was formally modeled by Focardi et al. [6]. Timing attack may also be combined with other side channel attacks in order to improve its performance and efficiency, such hybrid attacks were introduced in [14,15].

Modular multiplication is used in some cryptographic algorithms, such as public key encryption and digital signature. In such cases, a secret parameter of system is maybe multiplied by a public value. By default, modular multiplication is not an NP problem, if attacker knows the public input and the multiplication result, he simply finds out the secret parameter. But in some cryptographic algorithms, such as ElGamal Signature and Digital Signature Algorithm, the multiplication result isn't known value; it is kept secret [13].

To our best knowledge, until now almost all known timing attacks, except timing attacks against block ciphers, are based on time measurement of a modular exponentiation. In this paper we present new timing attack on modular multiplication. We use a technique like Dhem's technique, which is used to attack on

RSA, and propose practical timing attack on Blakley's modular multiplication algorithm. In the proposed attack, it is assumed that Blakley's modular multiplication is used to compute $a.b$ mod q, attacker knows b, q, and running time of algorithm but he is unaware of the multiplication result. The timing attack finds the parameter a. Running time of each step of Blakley's algorithm is not independent of other steps, so correlation between running time of steps will be considered in the attack. The attack is applied on DSA's on-line signature generation phase. Experimental results show that 1,000,000 time measurements are sufficient to find 160 bits of the secret key.

This paper is organized as follows. Blakley's modular multiplication algorithm and its application in DSA's signature generation phase are described in section 2. The proposed timing attack is explained in section 3. Running time of Blakley's algorithm and inter-steps dependencies are also discussed in section 3. Section 4 introduces a heuristic algorithm to solve some involved problems with attack. Practical results of applying the attack on an on-line implementation of DSA are presented in section 5 and section 6 concludes this paper.

2 Blakley's Modular Multiplication Algorithm

Blakley or interleaved algorithm is one of the algorithms are deployed in implementing modular multiplication [1]. Blakley's algorithm interleaves multiplication and modular reduction. At each step of multiplication, intermediate results are reduced to desired modulus q. If all numbers are t bits, the algorithm is as following:

Step 3 is like left shift in multiplication. If current bit of operand a, a_j, is 1, operand b is added to partial product, p. Steps 4 and 5 as well as steps 8 and 9 reduce partial product to modulo q. After step 3 or 7, there is always $p < 2q$, so one subtraction, in steps 4 or 8, is sufficient to reduce partial product to modulo q.

In the remainder of this paper following definitions are used:

Addition-1: The modular addition is done in steps 3, 4, and 5 to compute $p = p + p \bmod q$.

Addition-2: The modular addition is done in steps 7, 8, and 9 to compute $p = p + b \bmod q$.

$Er_{1,i}$: When the condition of step 5 in round $j = i$ is true, an extra assignment is done, this is called as occurrence of $Er_{1,i}$.

$Er_{2,i}$: When the condition of step 9 in round $j = i$ is true, an extra assignment is done, this is called as occurrence of $Er_{2,i}$.

Er : Either $Er_{1,i}$ or $Er_{2,i}$.

$p_{1,i}$: Value of p, which is used in the right hand side of Addition-1 in round $j = i$.

$p_{2,i}$: Value of p, which is used in the right hand side of Addition-2 in round $j = i$.

Blakley's algorithm has a few conditional statements that cause running time of the algorithm be dependent on the input values. Hence, it is vulnerable to timing attack. In our attack scheme, it is assumed that attacker knows b, q and can measure running time of the algorithm, but he does not know the multiplication result. The input parameter a is the secret parameter. Such situation exist in signature generation phase of DSA. DSA signs a hash of message M, $h(M)$, and produce two values r and s as following:

- $r = (g^k \bmod p) \bmod q$.
- $s = k^{-1}.(h(M) + x.r) \bmod q$.

Where g, p and q are public parameters, k is a random number which is generated for each message, and x is the long-term key, i.e. signer's permanent secret key. In order to improve performance and reduce bit size of intermediate results, it is preferred that compute the output s in the following steps:

1. $z = x.r \bmod q$.
2. $z' = h(M) + z \bmod q$.
3. $s = k^{-1}.z' \bmod q$.

It is assumed that modular multiplications is implemented using Blakley's algorithm instead of Montgomery multiplication. As, Montgomery algorithm is not suitable for a single modular multiplication. Suppose a, b, and q are t-digit integers with $0 < a, b < q$. Montgomery algorithm requires a precomputation on inputs, a, b. Neglecting the cost of the precomputation on the input, Montgomery multiplication algorithm computes $a.b.R^{-1} \bmod q$, where R is a constant parameter of the algorithm, while the result is obtained through $2t(t+1)$ single-precision multiplications. The computation of $a.b \bmod q$ is done in $4t(t+1)$ single-precision operations through the application of Montgomery multiplication to $a.b.R^{-1} \bmod q$ and $R^2 \bmod q$. Using classical modular multiplication (Multiplication $a.b$ then division by q) would require $2t(t+1)$ single-precision operations and no precomputation. Hence, the classical algorithm is superior for doing a single modular multiplication [12]. But Blakley's algorithm is more computational effective that classic modular multiplication, because it requires less memory to store intermediate results and it does not use the complicated division operation. In addition to using Blakley's algorithm, it is assumed that the secret key x is passed as parameter a to Blakley's algorithm. With these assumptions, the conditions of the timing attack are met, i.e. the attacker knows r and q, while z is kept as a private intermediate result. But attacker can not measure the running time of modular multiplication in step 1 directly.

There are two general implementations of DSA: on-line and off-line. In on-line implementation, both r and s are computed when there is signing request. But in off-line implementation, r is computed regardless of a signing request and the computed (r, k^{-1}) is stored. When there is a signing request, only s will be computed using $h(M)$ and a stored (r, k^{-1}). In this paper, the on-line implementation of DSA are attacked. In this case an attacker measure the running time for computing r, z, z', and s altogether. The execution time of r, s, and z' will be considered as noise included in a measured timing data. Attacker can compensate for the noise by increasing the number of measurements.

3 Timing Attack

Our approaches for guessing bits likes the approach which Dhem et al. used to attack on RSA, modular exponentiation [4]. In which, to guess a bit of secret key; it is assumed that the bit is one and algorithm is simulated on attacker's machine using gathered inputs from the under attack machine. According to a feature of implementation, extra reduction in Montgomery algorithm, the input set is divided into two subsets. Statistics of the running time of these two subsets are obtained. According to an oracle function, which is defined on the statistics, the bit of the secret key is guessed. In our attack, occurrence of the $Er_{2,i}$ is used as implementation feature. The statistic is the average running time and the oracle function uses difference between the averages. With the following assumptions, attacker uses "Timing Attack" algorithm to guess bit a_i.

- Attacker collects a set of inputs, which are used for a parameter b of Blakley's algorithm, $B = \{b_{(1)}, b_{(2)}, \ldots, b_{(n)}\}$.
- Attacker measures running timing of the algorithm for each input, $T = \{T_1, T_2, \ldots, T_n\}$, where T_i is the running time of the algorithm for input $b_{(i)}$.
- Attacker knows a few most significant bits of the parameter a, i.e. $a' =< a_{t-1}a_{t-2}a_{t-3} \ldots a_{i+1} >$, where $< a_i a_j >$ denotes concatenation of the a_i and a_j bits.

```
**ALGORITHM** Timing Attack
**INPUT:** $B$, $T$, $q$, $\overline{d}$, and $a'$
**OUTPUT:** $a_i$
1. Generate a temporary key, $a'' =< a'1 >$
2. Run Blakley's algorithm from $j = t - 1$ to $j = i$ using $a''$ for each $b_{(k)} \in B$.
3. According to the occurrence of $Er_{2,i}$, $T$ is divided into two sets, $T'_0$ and $T'_1$:
        $T'_0 = \{T_j \in T | Er_{2,i} \text{ does not occur}\}$
        $T'_1 = \{T_j \in T | Er_{2,i} \text{ occurs}\}$
4. Compute average of $T'_0$ and $T'_1$, which are named $\overline{T_0}$ and $\overline{T_1}$ respectively.
5. Find difference between the average of times: $d = \overline{T_1} - \overline{T_0}$.
6. If $d > \overline{d}$, $a_i$ is guessed as zero,
      else $a_i$ is guessed as one.
```

Note that step 2 of the "Timing Attack" algorithm, simulating Blakley's Algorithm using gathered data from victim's machine, is done on attacker's machine. The variable d is the statistic of attack, the variable can be either of two random variables d_0 or d_1. When the bit a_i is actually one, d is equal to d_1, and when the bit is zero, d is equal to d_0. The "Timing Attack" algorithm will guess bit a_i correctly, if distributions of the random variables d_0 and d_1 are not overlapped, in this case the input $\overline{d}$ is the border between distribution of d_0 and d_1. In step 6, it is decided whether d equals d_0 through comparing it with $\overline{d}$, and a guess on a_i is given according to the result. Attacker repeats this algorithm to find out the subsequent bits.

Correctness of “Timing Attack” depends upon following claims:

Claim 1: Distributions of the random variables d_0 and d_1 are not overlapped, therefore we can find the border, $\overline{d}$.

Claim 2: $\forall d' \in d_0$ and $\forall d'' \in d_1$ we have $d' > d''$, so the step 6 guesses the a_i correctly.

In following sections, we investigate the claim 1 and 2 and propose an algorithm to find the $\overline{d}$.

3.1 Running Time of Blakley's Algorithm

This subsection elaborates the claim 1 and 2. In order to investigate the claims, distributions of d_0 and d_1 should be obtained. It requires that running time of Blakley's algorithm be inspected in more details. The running time of Blakley's algorithm can be formulated as following:

$$T(r) = \sum_{i=t-1}^{0} (t_1 + \alpha_i t_2 + \beta_i (t_1 + \gamma_i t_2)) \tag{1}$$

where:

- t_1: is the running time of either steps 3 and 4 or steps 7 and 8.
- t_2: is running time of either step 5 or step 9.
- α_i: is 1 if $Er_{1,i}$ occurs, otherwise it is 0.
- β_i: is one if bit a_i is one.
- γ_i: is 1 if $Er_{2,i}$ occurs, otherwise it is 0.

Running time of each modular addition in Blakley's Algorithm is not independent of other modular additions. The dependency between running time of steps of Blakley's algorithm is a major difficulty in obtaining the distributions. To consider effect of modular multiplications on each other, *Extra Reductions Neighborhood Window*, ERNW, is used. $ERNW\{x, y\}$ of an Er is a set, containing x numbers of Ers may be occurred before the Er and y numbers of Ers can be occurred after the Er. It is supposed that occurrence of the Er is independent of other Ers that do not belong to the window. For example, if $a_{i+1} = 1$, $a_i = 1$ and $a_{i-1} = 0$, the $ERNW\{1, 1\}$ of $Er_{2,i}$ is $\{Er_{1,i}, Er_{1,i-1}\}$, and $ERNW\{2, 2\}$ of $Er_{2,i}$ is $\{Er_{2,i+1}, Er_{1,i}, Er_{1,i-1}, Er_{1,i-2}\}$. The $Er_{2,i-1}$ does not belong to this set, because $a_{i-1} = 0$ and $Er_{2,i-1}$ cannot occur.

To compute the distributions, we need expected number of messages in T'_0 and T'_1 and their running time. Equation 1 states that running time of a message is a summation of running time of modular additions. Two random variables, p and b, are involved in running time of modular additions. Expected number of messages and running time of each message are approximated in $(p, b)-$plan. The $(p, b)-$plan can be divided into some regions according to a Er and its $ERNW\{x, y\}$. For example if $Er_{2,i}$ and its $ERNW\{1, 0\} = \{Er_{1,i}\}$ are considered, the $(p, b)-$plan is first partitioned into two regions, a region where $Er_{1,i}$

is occurred and a region where it does not occurred. Each of these regions are then divided into two sub-regions according to $Er_{2,i}$. This partitioning is shown in Fig. 1. An unique code is assigned to each region that shows a situation of $Er_{1,i}$ and $Er_{2,i}$, for example "region 01" contain all $p_{1,i}$ that cause $Er_{1,i}$ occurs but $Er_{2,i}$ does not occur.

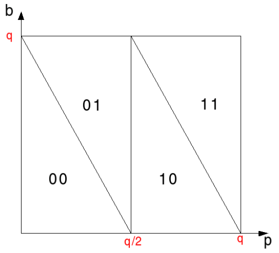


Fig. 1. $(p, b)-$plane for $ERNW = \{1, 0\}$ of $Er_{2,i}$

The "Approximate the d_0 or d_1" algorithm uses such partitioning, obtains expected messages numbers in T'_0 and T'_1 and their running time, and finds an approximation for d_0 or d_1 for given $ERNW\{x, y\}$.

```
ALGORITHM Approximate the $d_0$ or $d_1$
INPUT: $B$, $T$, $ERNW\{x, y\}$ of $Er_{2,i}$, and $a'$
OUTPUT: Approximation for $d_0$ or $d_1$.
1. To obtain an approximation for $d_0$ , create $ERS = ERNW\{x, y\}$.
To obtain an approximation for $d_1$, create
$ERS = ERNW\{x, y\} \bigcup Er_{2,i} = \{\underbrace{Er_1, Er_2, Er_3, \ldots, Er_x}_{x}, Er_{2,i}, \underbrace{Er_1, Er_2, \ldots, Er_y}_{y}\}$.
2. According to $Er$s belong to $ERS$, the $(p, b)-$plane is divided into $2^{|ERS|}$ regions.
Each region specifies a situation of $Er$ occurrence and a code is assigned to this region
as $< Er_1 Er_2 \ldots Er_y >$. For example, the region $< 00 \ldots 0 >$ covers all $(p, b)$ that none
of modular additions has extra reduction.
3. According to $Er_{2,i}$, $B$ is divided into two subsets, i.e. $B_0$ and $B_1$. $B_0$
contains all inputs that $Er_{2,i}$ does not occur for them, if $a'' =< a'1 >$ is used to
simulate the Blakley's algorithm. $B_1$ contains remaining members of $B$. $B_0$ and $B_1$ also
define two regions in $(p, b)$-plane.
4. For $B_0$ and $B_1$, find their regions overlap size with the created regions in step 2.
5. Find approximation for average running time of $B_0$ and $B_1$:
$\overline{T_0} = \sum_{\text{for all regions}}(\text{Overlap Sized with } B_0)(\text{Ham(the code of region)})$
$\overline{T_1} = \sum_{\text{for all regions}}(\text{Overlap Sized with } B_1)(\text{Ham(the code of region)})$
Where Ham($< \ldots >$) is a hamming weight of a code.
6. Return $\overline{T_1} - \overline{T_0}$
```

Table 1. d_0 and d_1 for some $ERNW$

$ERNW$	$\{\}$	$\{Er_{1,i}, Er_{1,i-1}\}$	$\{Er_{1,i+1}, Er_{1,i}, Er_{1,i-1}, Er_{2,i-1}\}$	$\{Er_{1,i+1}, Er_{2,i+1}, Er_{1,i}, Er_{1,i-1}\}$
d_0	0	$\frac{1}{4}t_2$	$\frac{1}{2}t_2$	$\frac{120}{192}t_2$
d_1	t_2	$\frac{7}{16}t_2$	$\frac{1}{3}t_2$	$\frac{1}{3}t_2$

Dividing $(p, b)-$plane in step 2 and finding overlap size in step 4 of the algorithm, will be very complicated, if there is no constraint on inputs. We consider the following assumptions:

1. p and b have uniform distributions in interval $[0, q]$.
2. p and b, which are used in the right hand side of the Addition-2, are independent.

The validity of these assumptions will be discussed in subsection 3.2. Using these assumptions, d_0 and d_1 are approximated for some $ERNW$, which are shown in Table 1. It can be seen from the table that:

– When bigger $ERNW$ are considered, i.e. dependency between more bits are considered, d_0 increases and d_1 decreases.
– For adequate large $ERNW$, we have $d_0 > d_1$.

It is easy to experimentally apply the approximation algorithm to larger $ERNW$ and verify that $d_0 > d_1$. Table 1 also indicates that d_0 and d_1 are dependent on $ERNW$ members. Hence they are dependent on the bits of the input a, which is assumed as secret parameter of a cryptographic function. Attacker can not find distribution of d_0 and d_1 directly, because he does not know the secret parameter. Even if he had a machine similar to the victim's machine, he could not directly find the distribution of the random variables, because the distributions are dependent on the secret parameter value. In the section 4 a heuristic algorithm is described to have a solution for the problem.

3.2 Assumptions

Two assumptions yield $d_0 > d_1$. First, it is supposed that p and b are distributed uniformly on interval $[0, q]$. If an uniform random number generator is used, operand b, which is passed as input to Blakley's algorithm, has uniform distribution. p is obtained from b. It is easy to show that if has b uniform distribution, p will distribute uniformly, so the first assumption is valid.

Second, it is supposed that the used p and b in the right hand side of the Addition-2 are independent. Blakley's algorithm scans bits of the a from the most significant to the least significant bit. Before the first most significant "1" of the a, the Addition-2 hasn't been executed. In a few rounds after the most significant one, if Addition-2 executes, p and b are not independent; however the

Table 2. Correlation coefficient for a few rounds

i	$t-1$	$t-2$	$t-3$	$t-4$	$t-5$	$t-6$
a_i	0	0	1	1	1	1
Correlation Coefficient	-	-	-	0.500	0.167	0.071

correlation coefficient of p and b decreases in each round. Correlation coefficient in each round is dependent on previous bits of a. Table 2 shows the value of bit a_i and the correlation coefficient of $p_{2,i}$ and b in each round, in an exemplary experiment.

Therefore the second assumption is also valid, except a few rounds in start of Blakley's algorithm. Due to these correlations in the rounds, the random variable d (step 5 of "Timing Attack") does not show expected behavior, so $d_0 \leq d_1$ in a few rounds in start of Blakley's algorithm. Thus, the attacker can not guess the most significant bits correctly by simply comparing random variable d, against distributions of d_0 and d_1. The following heuristic algorithm solves this problem, too.

4 Threshold Finding Algorithm

Here, we propose a simple heuristic algorithm to solve the above mentioned problems. First, obtaining the exact distribution of random variables d_0 and d_1 is not necessary to run the attack, but separating the distributions of d_0 and d_1 is sufficient. "Find Threshold" algorithm separates the distributions and finds the border, $\overline{d}$, between the distribution of d_0 and the distribution of d_1. The step 6 in the "Timing Attack" uses the $\overline{d}$ to guess a bit of the secret key. It was already shown that $d_0 > d_1$, hence in this step if $d > \overline{d}$ it means that $d \in d_0$ and the bit a_i is guessed as 0, but if $d < \overline{d}$ then $d \in d_1$. Second, when attacker uses the "Find Threshold" algorithm, he won't be worry about misbehavior of random variable d in start of attack.

"Find Threshold" assumes that attacker knows position of the most significant "1" of the secret parameter a, which is shown by o. He tests all cases of a few subsequent bits and find $\overline{d}$. The method is described formally in "Find Threshold" algorithm.

After $\overline{d}$ is obtained, the "Timing Attack" algorithm is applied on B, T and $a^{(i)}$, and the remaining bits, $a_{t-o-w-1}$ to a_0, are found. Attacker gets 2^w guesses for $< a_{t-o-w-1} \ldots a_0 >$ which only one of them is valid. Other constraints on parameters might be used to find the correct guess. For example when we are attacking on DSA. Obtained secret key, x, must satisfy $y = g^x \bmod q$.

In addition to finding an estimation of $\overline{d}$, the attacker ignores misbehavior of d with this algorithm. As if w is large enough, the correlation between p and b is degraded and the assumptions which were discussed in subsection 4.1 are valid, hence it is expected that $d_0 > d_1$.

```
ALGORITHM Find Threshold
INPUT: Position of the most significant one, $o$, size of guessing window, $w$.
A set of inputs of Blakley's Algorithm, $B = \{b_{(1)}, b_{(2)}, \ldots, b_{(n)}\}$.
Running time of algorithm for each input. $T_i$ is running time of $b_{(i)}$, $T = \{T_1, T_2, \ldots, T_n\}$
OUTPUT: $\overline{d}$
1. Construct $2^w$ guesses for $w$ successive bits after the most significant one:
$a^{(0)} = \underbrace{00\ldots0}_{o-1}1\underbrace{00\ldots0}_{w}$, $a^{(1)} = \underbrace{00\ldots0}_{o-1}1\underbrace{00\ldots1}_{w}$, $a^{(2)} = \underbrace{00\ldots0}_{o-1}1\underbrace{00\ldots10}_{w}, \ldots,$
$a^{(2^w-1)} = \underbrace{00\ldots0}_{o-1}1\underbrace{11\ldots1}_{w}.$
2. For each $a^{(i)}$, apply the "Timing Attack" algorithm using $B$, $T$ and $a^{(i)}$.
In this case no bit is guessed, only obtained $d$ will be used. The obtained $d$ is added
to set $D'$.
3. Threshold $\overline{d}$ is average of $D'$.
```

Table 3. Attack result, when $w = 2$

$\lvert T\rvert \times (10^5)$	5	5	5	10	10	10	10
$\overline{d}$	11370.3	12911.6	11178.7	13096.8	10529.2	11765.0	10239.8
$Err.Num.$	5	0	1	0	1	0	1

5 Practical Results

The proposed timing attack was practically applied on pure modular multiplication and on-line implementation of DSA. Here, only the results of the attack against DSA are presented. Victim machine is an on-line implementation of DSA running in MS-DOS operating system on Athlon-XP 800 MHz. Two internal 32 bit counters of CPU are used as timing measurement facilities [9]. Running time is measured with respect to the number of required CPU cycles to run the DSA algorithm.

There are two parameters in our attacks, i.e. size of T, the number of timing measurements, and size of the guessing window, w, which is used to in "Find Threshold" algorithm. Table 3. and Table 4. show $\overline{d}$ and the number of incorrect guesses in a run of attack for a constant secret key x, for different numbers of timing measurements and window sizes of 2 and 3, respectively.

In these tables, $\overline{d}$ and the number of incorrect guesses are not reported, when the number of measurements is less than 5×10^5. As in such a case, the distributions of d_0 and d_1 are overlapped and the border between them, $\overline{d}$, is meaningless. Hence, applying the attack is impossible. These tables show that as the number of measurements increases, the number of errors degrades. The tables also show that, using bigger window is less erroneous. When very small window are used, d is prone to error. So, obtained $\overline{d}$ is unreliable to properly separate the distribution of d_0 and d_1. Our practical attacks found 160 bit of DSA secret key using window size 3 and 10^6 timing measurements.

Table 4. Attack result, when $w = 3$

$\|T\| \times (10^5)$	5	5	5	10	10	10	10
$\overline{d}$	11288.3	11731.3	11064.1	13004.4	10410.6	11691.1	10124.6
$Err.Num.$	1	0	5	0	0	0	0

The running time of the attack and the amount of required time to gather the timing samples are directly related to the size of the parameters w and $|T|$. Although increasing the size of parameters enhances the results of attack, it causes more running time. The Running time of the "Timing Attack" algorithm is $O(|T|)$, running time of "Find Threshold" algorithm is exponentially related to w. An increasing in the window size from w to $w + 1$, doubles the attack time to find valid $a^{(i)}$, because the number of $a^{(i)}$ is 2^w.

In our experiments, given the T and the a^i, it takes about 70 min. to run the "Timing Attack". Empirical running time of "Threshold Finding" is dependent on the w. When $w = 3$ it takes about 12 min.

6 Conclusion

In this paper, we proposed a new timing attack on an implementation of modular multiplication, which is called as Blakley's modular multiplication algorithm. To our best knowledge this is first time that timing attack is applied on modular multiplication. To attack, it is assumed that the secret key is passed as parameter a to Blakley's algorithm, attacker knows the parameter b, and can measure the running time of Blakley's algorithm. Blakley's algorithm may be employed for multiplication in public key cryptography or digital signature. If the assumptions are valid, their implementations are vulnerable.

In this paper we focused on DSA and applied our timing attack on on-line implementation of DSA. Our experimental results shows that the proposed timing attack finds out the secret key used in DSA, practically. The attack only gathers timing sample from victim machine and does not require extra information about implementation details or a machine similar to victim's machine. The set of measured timing data are divided into two subsets based on the extra reduction in the Addition-2 of Blakley's algorithm. It is shown that the secret key can be guessed according to the difference between averages of these subsets.

Our timing attack is against a specific implementation of modular multiplication. The modular multiplication $a.b \bmod q$ is a symmetric operation in a and b, this is not anymore the case in the internal operations performed by the Blakley's Algorithm. So, if x (DSA secret key) is passed as input b of Blakley's Algorithm then not much information leaks and our attack is not applicable. Using $Er_{1,i}$ instead of $Er_{2,i}$ in attack, timing attack on more general implementation of Blakley's algorithm, in which secret parameter is passed as parameter b, use mathematical model to prove $d_0 > d_1$, and timing attack on other implementation of modular multiplication are open problems.

References

1. G. R. Blakley. "A computer algorithm for calculating the product AB modulo M". *IEEE Transactions on Computers*, 32(5):497-500, May 1983.
2. D. Brumley, D. Boneh, "Remote Timing Attacks are Practical", *Proceedings of the 12th Usenix Security Symposium*, 2003.
3. J. Cathalo, F. Koeune, and J.-J. Quisquater, "A New Type of Timing Attack: Application to GPS", *Cryptographic Hardware and Embedded Systems - CHES 2003*, LNCS 2779, 2003, pp. 291–303.
4. J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestre, J. J. Quisquater and J. L. Willems, "A Practical Implementation of the Timing Attack", *3rd Working Conference on Smart Card Research and Advanced Applications - CARDIS 1998*, Springer-Verlag, LNCS No. 1820, 1998.
5. E. W. Felten, and M. A. Schneider, " Timing Attacks on Web Privacy", *CCS 2000, Athens, Greece*, 2000.
6. R. Focardi, R. Gorrieri, R. Lanotte, A. Maggiolo-Schettini, F. Martinelli, S. Tini, E. Tronci, "Formal Models of Timing Attacks on Web Privacy", *Electronic Notes in Theoretical Computer Science*, vol. 62, 2001.
7. H. Handschuh, and H. M. Heys, "A Timing Attack on RC5", *SAC'98*, LNCS 1556, 1999, pp. 306–318.
8. A. Hevia, M. Kiwi, "Strength of Two Data Encryption Standard Implementations Under Timing Attacks", *LATIN'98*, LNCS 1380, 1998, pp. 192–205.
9. Intel, "Using the RDTSC instruction for performance monitoring", *Technical report*, 1997.
10. P. C. Kocher, "Timing Attacks on Implementations of Diffie–Hellman, RSA, DSA, and Other Systems", *Advances in Cryptology - CRYPTO'96*, Springer-Verlag, LNCS No. 1109, 1996, pp. 104–113.
11. F. Koeune, J. J. Quisquater, "A Timing Attack against Rijndael", *Technical Report CG-1999/1*, June 1999.
12. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, "Handbook of Applied Cryptography", *CRC Press*, 1996.
13. National Institute of Standard and Technology (NIST), "Digital Signature Standard", FIPS PUB 186-2, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf
14. W. Schindler, "A Combined Timing and Power Attack", *PKC 2002*, LNCS 2274, 2002, pp. 263–279.
15. W. Schindler, C. D. Walter, "More Detail for a Combined Timing and Power Attack against Implementations of RSA", *9th IMA International Conference on Cryptography and Coding*, LNCS No. 2898, 2003, pp. 245–263.
16. W. Schindler, "A Timing Attack against RSA with the Chinese Remainder Theorem", *Cryptographic Hardware and Embedded Systems - CHES 2000*, Springer-Verlag, LNCS No. 1965, 2000, pp. 109–124.
17. W. Schindler, "Optimized timing attacks against public key cryptosystems", *Statistics and Decisions*, volume 20, 2002, pp. 191–210.
18. W. Schindler, "On the Optimization of Side-Channel Attacks by Advanced Stochastic Methods", *8th International Workshop on Practice and Theory in Public Key Cryptography PKC 2005*, Springer-Verlag, LNCS No. 3386, 2005, pp. 85–103.

Protecting AES Software Implementations on 32-Bit Processors Against Power Analysis

Stefan Tillich, Christoph Herbst, and Stefan Mangard

Graz University of Technology,
Institute for Applied Information Processing and Communications,
Inffeldgasse 16a, A–8010 Graz, Austria
{Stefan.Tillich,Christoph.Herbst,Stefan.Mangard}@iaik.tugraz.at

Abstract. The Advanced Encryption Standard is used in many embedded devices to provide security. In the last years, several researchers have proposed to enhance general-purpose processors with custom instructions to increase the efficiency of cryptographic algorithms. In this work we have evaluated the impact of such instruction set extensions on the implementation security of AES. We have compared several AES implementation options which incorporate state-of-the-art software countermeasures against power-analysis attacks—with and without the use of instruction set extensions. For both scenarios we provide a thorough analysis for different countermeasures with regard to security, performance, and memory. We have found that even a moderate level of protection requires a considerable overhead both in terms of speed and memory. The instruction set extensions, which have been solely designed to increase performance, help to reduce this overhead, but it still remains high. An implementation with proper protection through software countermeasures is only feasible in a setting where the need for resistance against power analysis outweighs the need for performance.

Keywords: Advanced Encryption Standard, side-channel attacks, software countermeasures, instruction set extensions, implementation security, DPA, power analysis.

1 Introduction

Today, more and more computational tasks are performed on small embedded systems. Most of these systems feature an embedded processor with a wordsize of 8, 16, or 32 bit. 32-bit processors are common in mid-range to high-end embedded systems like PDAs and cellphones but can also be found in wireless sensor networks and even in some high-end smartcards. Many applications require the execution of cryptographic algorithms in order to achieve some security assurances, e.g. data confidentiality or authentication. But while the algorithms themselves are secure, a straightforward implementation on a device is very likely to be vulnerable to side-channel attacks. Such attacks measure and analyze some physical property of the device while it performs cryptographic operations, with the goal of extracting the key used by the device. Power-analysis attacks, which exploit the power consumption, have been studied very thoroughly, and many proposals have been made on how to make them more effective as well as on how to defend against them.

While it is unlikely that side-channel attacks can be fully prevented, appropriate countermeasures can hamper an attack to the point where it becomes practically infeasible. Some hardware countermeasures have proven to be rather effective in doing this. On the other hand, counteracting power-analysis in software is very hard, as the programmer normally has only a very limited influence on the power consumption of the processor. To make things even worse, in the last years new attack variants have emerged, which are very effective against software implementations of cryptographic algorithms.

We have investigated the current situation regarding software countermeasures against state-of-the-art power-analysis attacks. We have focused on 32-bit embedded processors and on the AES algorithm, but most of the discussed methods also work for processors of different wordsize and other cryptographic algorithms. This paper is organized as follows: In Section 2 we discuss software countermeasures for power analysis in principal. We elaborate on the effectiveness of these countermeasures in Section 3. Section 4 focuses on state-of-the-art attacks on protected software implementations. In Section 5 we estimate the effect of different countermeasures on performance, memory, and implementation security. We draw conclusions in Section 6.

2 Power-Analysis Countermeasures for Software Implementations

In order to secure software implementations of cryptographic algorithms against power-analysis attacks there are two suitable approaches, namely masking and hiding. Masking conceals all intermediate values during the calculation with a random mask. Hiding techniques try to break or at least weaken the link between processed intermediate values and the side-channel leakage at a certain moment of time. In this section we give an overview of these two types of countermeasures.

2.1 Masking

Masking means to conceal each intermediate value a with a random value m, which is called mask. These masks are generated by the device for each execution of the algorithm and they are not known by an attacker. Generally, we can distinguish between Boolean and arithmetic masking. In arithmetic masking, intermediate values and masks are combined with an arithmetic operation like addition or multiplication. For AES, Akkar et al. suggested a multiplicative masking scheme [1] where the intermediate values are concealed with a multiplicative mask $a_m = a \cdot m \pmod{n}$. Boolean masking uses the exclusive-or operation to combine intermediate values and masks $a_m = a \oplus m$. Masking schemes for software implementations of AES based on Boolean masking have been proposed in [5] and [6].

Power-analysis attacks are prevented by masking because each intermediate value is masked with a random value and thus the power consumption caused by this value can not be predicted by an attacker. This holds under the condition, that each masked value a_m is independent of a. Usually the masks are applied to the plaintext values at the beginning of the algorithm. During the execution of the algorithm one has to keep track of the modification of the masks by the operations of the algorithm. For the AES operations ShiftRows and AddRoundKey, this can be done with virtually no effort, because they are linear and do not change the applied masks. The MixColumns operation combines different values of one column of the AES State, and therefore, one has to calculate the modified masks after this operation. To monitor the change of the masks through the nonlinear SubBytes transformation, a more elaborated approach is needed. A very common way to implement the SubBytes transformation in software is to use lookup tables: $a_{out} = S(a_{in})$, where S denotes the AES S-box, which is used on every byte of the State for SubBytes. In order to mask the SubBytes transformation, we have to calculate a masked table S' such that $S'(a_m) = S'(a \oplus m) = S(a) \oplus m'$. When implementing such a masking scheme, care has to be taken that all intermediate values stay masked during the critical computations of the algorithm. At the end of the calculation of the algorithm all masks have to be removed. Especially unintended unmasking has to be considered. This can happen in a device which leaks the Hamming distance of subsequently processed values, i.e. the Hamming weight of the exclusive-or of these two values [12]. Two subsequent values with the same Boolean mask would therefore be unmasked.

Provably secure masking schemes for AES have been published in [2] and [15]. These schemes focus on hardware implementations. In [16] a proposal for a software implementation of the scheme presented in [15] has been made. This scheme has higher performance rates than a conventional lookup scheme, as long as a set of masks is only used for a single encryption. In schemes where masks are used for more than one encryption, the lookup table approach is still faster. This is one of the reasons why we have chosen a lookup based scheme for our implementation.

Masking schemes are an appropriate choice to defeat first-order power-analysis attacks. Nevertheless, masked implementations are still vulnerable to higher-order and template attacks. Higher-order attacks are discussed in [14], [19], [17], and [8]. A template based attack on a protected AES software implementation has been published by Oswald et al. in [13]. Due to the presence of these powerful attacks it is mandatory to combine masking schemes with a second type of countermeasures to raise the level of security.

2.2 Hiding

In general, hiding can take place in two domains, namely in the time domain and in the amplitude domain. Hiding in the time domain tries to randomize the time of occurrence of a specific operation, whereas hiding in the amplitude domain tries to reduce the effect of the performed operation on the overall power consumption.

For software implementations, hiding in the time domain is normally easier to achieve. The goal is to distribute the occurrence of critical operations and intermediate values over a given period during each execution of an algorithm. This leads to a reduced correlation of targeted values at specific points of time. Two appropriate methods to achieve this randomization are the insertion of dummy operations and shuffling of operations. Both insertion and shuffling are controlled by random values generated by the device. Inserted dummy operations should not be distinguishable from normal operations. Otherwise an attacker could be able to remove their effect from the power trace. Shuffling of operations means that for each execution of the algorithm, the order of the occurring intermediate values is changed. How these two methods can be applied to a software implementation of AES is described in Section 3.3.

Hiding in the amplitude domain is rather hard for software implementations. Nevertheless, a designer has the opportunity to choose only such instructions which leak a minimum amount of information. This technique highly depends on the used device and its leakage properties. The statistic effects of hiding have been investigated in [11], [4], and [3].

3 Effectiveness of Software Countermeasures

This section gives a thorough evaluation of software countermeasures that can be applied to secure an AES implementation on a 32-bit platform. In this context we have considered two classes of processing platforms. The first class consists of typical 32-bit embedded processors with a standard RISC instruction set architecture. The second class includes processors which have explicit support for cryptographic operations in their instruction set (cryptographic instruction set extensions).

For the instruction set extensions we have used the "advanced word-oriented AES extensions with implicit ShiftRows" described in [18]. These instructions work on 32-bit words performing either four parallel AES S-box lookups (`sbox4s/isbox4s/sbox4r`) or a MixColumns transformation for a single State column (`mixcol4s/imixcol4s`). In the following, we will give the definition of the functionality of the `sbox4s` and `mixcol4s` instructions. Note that *rs1* and *rs2* denote the two 32-bit input operands and *rd* the 32-bit result of the instruction. Brackets with indices are used to select a part of the respective 32-bit value, while | concatenates four 8-bit or two 16-bit values to a 32-bit value. *S-box* substitutes an 8-bit value according to the AES S-box, while *MixColumns* transforms a 32-bit value following the AES MixColumns operation.

sbox4s rs1, rs2, rd: rd[31..0] := S-box(rs1[31..24]) | S-box(rs2[23..16])
| S-box(rs1[15..8]) | S-box(rs2[7..0]);
mixcol4s rs1, rs2, rd: rd[31..0] := MixColumns(rs1[31..16] | rs2[15..0]);

The definition of `isbox4s` and `imixcol4s` is similar, with the difference that the inverse AES S-box and the InvMixColumns transformation are used, respectively. Finally, the `sbox4r` instruction has only one input operand, whose bytes are transformed with the AES S-box and where the result is rotated 8 bits to the left:

sbox4r rs1, rd: rd[31..0] := S-box(rs1[23..16]) | S-box(rs1[15..8])
| S-box(rs1[7..0]) | S-box(rs1[31..24]);

The `sbox4r` instruction is designed for use in the AES key schedule, while the other instructions are intended to speed up the AES round transformations.

In the following sections we analyze different options for power-analysis countermeasures. The most powerful attacks are listed and implementation-specific details for use of the instruction set extensions are given. The maximum correlation coefficient ρ is stated for each attack. The security gain can be approximated by the quotient of the correlation coefficient for an attack on an unprotected and on a protected implementation: An attack on the protected implementation requires at least $(\frac{\rho_{unprotected}}{\rho_{protected}})^2$ more power traces [12]. For our estimations we have $\rho_{unprotected} = 1$ and can therefore state the security gain as $(\frac{1}{\rho})^2$, where ρ always denotes the correlation coefficient for an attack on the protected implementation. Note that we state ρ for noise-free environments, which is sufficient to make a relative comparison of unprotected and protected implementations. The correlation coefficients observed in practical attacks will be lower due to noise. The correlation coefficient has been determined under the assumption that the Hamming weight of processed values leaks through the power consumption. Many devices leak the Hamming distance of subsequently processed values, but it is very hard to determine the correlation coefficient for such a setting without taking many details of the processor architecture and software implementation into account. We therefore take the Hamming-weight leakage model as a lower bound for devices that leak the Hamming distance. This assumption holds as long as the software implementation avoids potential vulnerabilities due to the Hamming-distance leakage, e.g. unintended unmasking as explained in Section 2.1.

3.1 Unprotected Implementation

An unprotected 32-bit AES software implementation is vulnerable to a multitude of attacks. One of the most powerful attacks is a first-order DPA on an 8-bit intermediate result after the S-box lookup ($\rho = 1$). The key expansion can also be targeted directly with a template-like attack as described in [10]. This attack extracts the Hamming weights of 8-bit intermediate values of the key expansion and uses the dependency of these values to narrow down the number of potential keys. The use of the instructions set extensions from [18] allows to calculate the key schedule with 32-bit values only, which makes this kind of attack infeasible.

3.2 Masking

A masked implementation protects critical intermediate values with a random mask. An intermediate value of the AES operation can be considered critical when it depends on a small portion of the (round) key and on the plaintext or ciphertext. In this case the attacker can guess the part of the key and verify her guess through analysis of the measured power traces. The choice of masks and the processing order of masked values must always be done carefully with regard to the leakage of the device to prevent problems like unintended unmasking.

If the masking countermeasure is implemented properly, it can prevent first-order DPA attacks. However, a masked implementation is still vulnerable to higher-order DPA attacks. In such an attack, several points of each power trace are combined to a single value with a preprocessing function *pre*. The resulting value again depends on some predictable value. The preprocessed values can then be subjected to a first-order DPA attack. Normally, a second-order DPA attack is sufficient to break a masked implementation. The targeted values for preprocessing are either a masked intermediate value and the corresponding mask, or two intermediate values with the same mask.

The best vantage point to break a masked AES implementation is the masked S-box lookup, which is used for SubBytes. This lookup requires masked 8-bit input and output values, which are easier to target than the masked 32-bit values resulting from other transformations (e.g. MixColumns). The cost for precomputing a single masked S-box is very high, and it is therefore necessary to reuse masked S-box tables. This results in the processing of 8-bit values with the same mask, which can be targeted in a second-order attack ($\rho = 0.24$, see [12]). But even if no 8-bit value carries the same mask, the preprocessing function could use the power consumption of the mask itself (which must occur at some time in the computation) as second value. In the worst case for the attacker, this 8-bit mask will only occur in form of a 32-bit word, where the other 24 bits are random (this can only be achieved with the help of the instruction set extensions). Even in this case, the level of protection against a second-order DPA attack is rather low ($\rho \approx 0.1$).

A possibility to prevent the S-box lookup in software is to perform most of the AES round as table lookup (T-box lookup). However, the precomputation of masked T-boxes would be much more costly than the precomputation of masked S-boxes. Moreover, a T-box lookup still requires an 8-bit masked input value, which can be targeted in an attack.

A masked implementation could use more than one mask for every intermediate value. However, it seems very difficult to generate a masked S-box table without processing the definite input and output masks at some point of time. All in all, the vulnerability to second-order DPA attacks is very hard to remove in a masked AES implementation.

3.3 Randomization

In the following, we will denote countermeasures of hiding in the time domain (cf. Section 2.2) as *randomization*. In a randomized AES implementation, the occurrence of a specific intermediate value at a specific point in time is reduced to a certain probability. This can be done by shuffling of operations and by random insertion of dummy operations. In this case an attacker needs to capture more power traces, in order to compensate for this uncertainty.

Simple solutions, like the random insertion of `nop` instructions, are likely to be detected and removed by an attacker. Therefore, if dummy operations are added, it is important that they can not be distinguished from the genuine operations. This can be achieved by performing the AES transformations on some dummy data.

The best degree of randomization can be achieved by using both the shuffling of operations and the insertion of dummy operations. In AES, the smallest unit of data, whose processing can be randomized, is the 8-bit input and output value used in the S-box lookup. The 16 S-box lookups per AES round can therefore be shuffled, resulting in a probability of $p = \frac{1}{16}$ for a specific value at a specific point in time. Dummy operations can be inserted by processing a certain number of dummy values. Processing of complete dummy States (i.e. 4×4-byte matrices) seems to be a good granularity for that purpose. If N dummy States are processed in addition to the genuine State, then the probability for the occurrence of a specific value goes down to $p = \frac{1}{(N+1) \cdot 16}$.

It would be very inefficient to perform a selection for each of the $(N+1) \cdot 16$ byte values separately. Moreover, the AES algorithm does not allow to perform all critical round transformations with just a single byte. The smallest value which is sufficient for all those transformations is a single State column. For practical implementation it is sufficient to determine the processing order of the bytes in an orthogonal way: The States are processed one after the other, i.e. the processing of the genuine State is randomly embedded within the processing of the dummy States. For each State, the columns are processed in a fixed order beginning with a randomly chosen column. For each column, the bytes are processed separately and also in a fixed order starting with a randomly chosen byte.

The randomization degree p determines the resistance against DPA attacks. The power traces obtained from an implementation with randomization are often referred to misaligned power traces. A direct DPA attack on the misaligned traces would require $(\frac{1}{p})^2$ more traces to compensate for the randomization. However, Clavier et al. [4] have proposed to sum up all points in the power trace, where the targeted value can occur. This approach is often referred to as *"windowing"*. With this approach, an attacker only requires $\frac{1}{p} = (N+1) \cdot 16$ more traces to defeat the randomization.

Therefore we can assume that the number of power traces to attack a randomized AES implementation scales up with a factor of only $(N+1) \cdot 16$, as $\rho = \frac{1}{p} = \frac{1}{(N+1) \cdot 16}$. Most of the overhead of a randomized implementation comes from the preparation of the randomization and the byte-wise processing of the AES State. Doubling the security (which corresponds to doubling of $(N+1)$) roughly doubles the total running time. This results in a very large overhead.

3.4 Masking and Randomization

For better protection, an AES implementation needs to combine masking and randomization countermeasures. However, there are still several possible attacks which can break such an implementation rather efficiently.

An attacker will try to defeat the masking with a second-order attack. At least one of the attacked intermediate values (i.e. a masked 8-bit value) is protected by randomization. As we have already outlined in Section 3.3, a very effective way to defeat randomization is to sum up the power consumption at all moments in time where the attacked value can occur (recall that for our considered randomization there are $(N+1) \cdot 16$ points in time, where N is the number of dummy States). The second attacked value can either be the mask of the first value, or another randomized intermediate value carrying the same mask as the first value.

There are two main strategies on how to use the second value in an attack. On the one hand, this value can be employed to introduce a bias in the occurring masks. On the other hand, the value can be combined with the first one to yield a result that depends on the unmasked value.

4 Attacks on Masked and Randomized AES Implementations

We have shown in the previous section, that a protection by masking or randomization alone can not withstand power-analysis attacks. In this section we analyze the possible attacks on software implementations which use a combination of both countermeasures. The attacks presented in this section have either been published and evaluated or are natural extensions or combinations of existing attacks. The method of windowing (cf. Section 3.3) published by Clavier et al. [4] is fundamental for all of the examined attacks, as it is a very good way to compensate the effects of the randomization countermeasure. The possibility of second-order DPA attacks has already been mentioned in the original publication of Kocher et al. [9]. Second-order attacks on software implementations of block ciphers have been analyzed in [14].

In [13], Oswald et al. have evaluated the effectiveness of template-based attacks against masked software implementations and have shown that such methods can be very effective. However, as long as the targeted operation used for template-building remains randomized in time, we assume that it is very hard to create well-matching templates, which lead to better results than techniques based on counteracting randomization, e.g. windowing.

4.1 Biasing Masks

A very powerful attack is to introduce a bias into the masks used by the device, which leads to a dramatic decrease of security. This idea has been introduced by Jaffe [7], and practically evaluated by Oswald et al. [13]. In practice, an attacker can bias a mask by examining a point of the power trace where the mask is processed and by discarding all traces which have a value above (or below) a certain threshold. Figure 1 shows the timeline of a power trace, where the time of occurrence of targeted values is marked at the top. Below the timeline, it is shown how the power consumption values at these times would be used in a biased-mask attack. Windowing is used to sum up the power consumption at all points in time in the selected traces where the attacked value can occur (due to randomization). A classical first-order DPA attack is performed on the resulting preprocessed power values.

Without instruction set extensions, the 8-bit masks of the S-box can be targeted directly during the generation of the masked S-box. With instruction set extensions, the masked S-box can be generated using only 32-bit masks (provided that four masked S-boxes are used). A bias of either the 8-bit or 32-bit mask has a devastating effect on the security. For example, biasing the 8-bit masks to a Hamming weight (HW) ≥ 6 yields $\rho = -0.1$ (for $N = 1$). For 32-bit masks, a bias of HW ≥ 20 results in $\rho = -0.05$ (again for $N = 1$).

Increasing the degree of randomization does not lower the correlation coefficient very effectively (see Table 2). Note that a possible defence against this attack could consist of randomizing the time of occurrence of each mask. However, the mask and values directly dependent on the mask occur at several points in the computation, e.g. generation of the mask, appliance of the mask to the S-box, calculation of the mask after MixColumns, (re)masking of the key schedule. Proper randomization of all these operations would be quite challenging and also incur a considerable overhead in terms of performance.

4.2 Combining Second-Order DPA and Windowing

A second-order DPA can be combined with windowing to break the masking and randomization. This approach can be seen as performing multiple second-order DPA attacks in parallel. The attack can be done by combining the power consumption for the

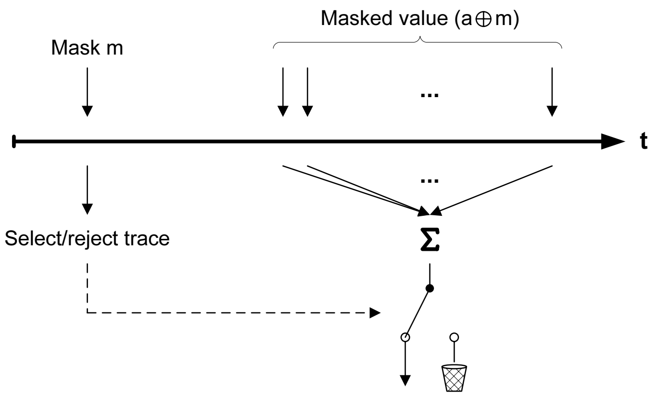


Fig. 1. Information extraction from power traces in a biased-mask attack

mask processing with each of the $(N+1)\cdot 16$ points in time where the targeted masked value can occur (due to randomization) using a second-order preprocessing function.

However, due to the randomization, the attacker does not know which of the resulting values corresponds to her targeted value. This is the same problem as in an implementation which has only randomization countermeasures. Consequently, an efficient solution is to sum up all $(N+1)\cdot 16$ preprocessed values and to perform a first-order DPA attack on the result. Figure 2 depicts this approach.

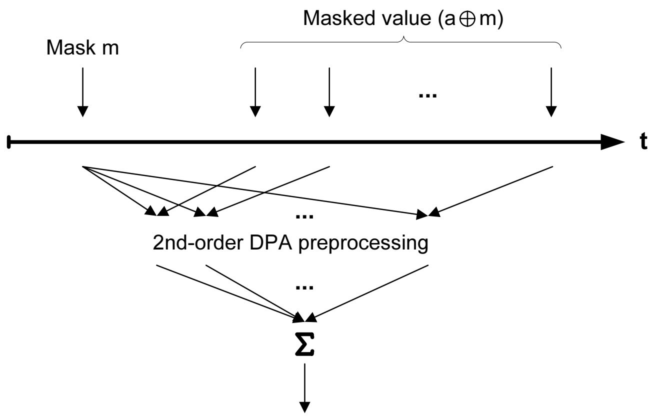


Fig. 2. Information extraction from power traces for second-order DPA combined with windowing

The effectiveness of this attack can be evaluated for both attack stages separately. In the first stage, the second-order DPA preprocessing function is applied to each pair of values (mask and masked value). For our randomization scheme we have an 8-bit masked value. As already stated in Section 3.2 we have $\rho = 0.24$ for 8-bit masks and $\rho \approx 0.1$ for 32-bit masks (using instruction set extensions). The summation of the second attack stage corresponds to windowing, which scales down the correlation coefficient with a factor of $\frac{1}{\sqrt{(N+1)\cdot 16}}$. The overall correlation coefficient is therefore very high: For the 8-bit masks we get $\rho = \frac{0.24}{\sqrt{(N+1)\cdot 16}}$, and for 32-bit masks we get $\rho \approx \frac{0.1}{\sqrt{(N+1)\cdot 16}}$. So in order to achieve $\rho = 0.01$, we would need at least $N = 5$.

Principally, it would be desirable to randomize the occurrence of the mask to the same degree as the masked value. This measure would require to sum up all possible combinations where mask and masked value can appear. The number of combinations is $((N+1)\cdot 16)^2$, which would lead to a reduction of the correlation by a factor of $(N+1)\cdot 16$ after windowing. At $N = 1$, the correlation would already be about as low as $\rho = 0.003$ for 32-bit masks. However, as already mentioned in Section 4.1, randomization of the mask would be very costly in terms of performance.

4.3 Targeting Weak Randomization

Targeting two randomized intermediate values which carry the same mask is normally less efficient than to target one fixed (e.g. the mask) and one randomized value. However, a weak randomization can be broken more easily with this strategy.

In this context, a weak randomization is one where two intermediate values with the same mask always occur with a fixed distance in time. An example for this are the S-box inputs of the first and second AES round, when the used S-boxes have the same input masks and the two lookups are not randomized separately. The attacker can therefore apply the second-order DPA preprocessing function to each such pair of values, which is depicted in Figure 3. The rest of the attack is exactly the same as the previously described one (summation followed by first-order DPA).

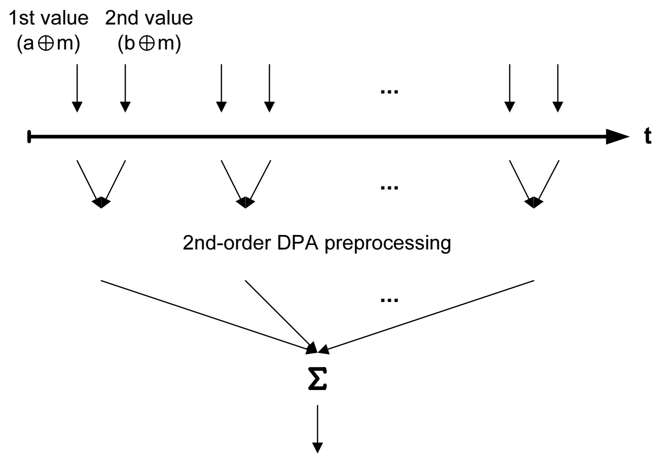


Fig. 3. Information extraction from power traces when attacking a weak randomization

Targeting two 8-bit intermediate values with the same mask is equivalent to targeting one intermediate value and the according 8-bit mask. The correlation coefficient is therefore $\rho = \frac{0.24}{\sqrt{(N+1)\cdot 16}}$. However, it might be necessary to hold some parts of the plaintexts constant and guess more than a single key byte to be able to set up a good hypothesis.

The effectiveness of this attack can again be evaluated for both attack stages independently. In the first stage, the second-order DPA preprocessing function is applied to each pair of values with the same mask. For our randomization scheme we have two masked 8-bit values, which yields $\rho = 0.24$ [12]. The summation of the second attack stage again corresponds to windowing, which reduces the correlation coefficient by a factor of $\frac{1}{\sqrt{(N+1)\cdot 16}}$. The resulting correlation coefficient remains rather high with $\frac{0.24}{\sqrt{(N+1)\cdot 16}}$ (e.g. $\rho = 0.01$ would require $N = 35$).

To counteract this attack it would be necessary to randomize the S-box lookup in the first and second AES round separately (and similarly in the two last rounds). This countermeasure would render the described attack less efficient than the other described attacks.

4.4 "Classical" Second-Order DPA on Windowed Traces

Another way to combine second-order DPA and windowing is to perform windowing first to counteract the effects of randomization, and to do a "classical" second-order DPA attack on the result. Figure 4 depicts the processing steps performed on every power trace. The resulting value can then be subjected to a first-order DPA attack. A preprocessing function *pre*, which is generally very effective, is the absolute difference of the inputs: $pre(a,b) = |a-b|$ [12]. For this function, it is important that both a and b are of the same magnitude, e.g. if a is a single point from the trace and b is a sum of n points, then the preprocessing function should scale a up to b: $pre(a,b) = |n \cdot a - b|$.

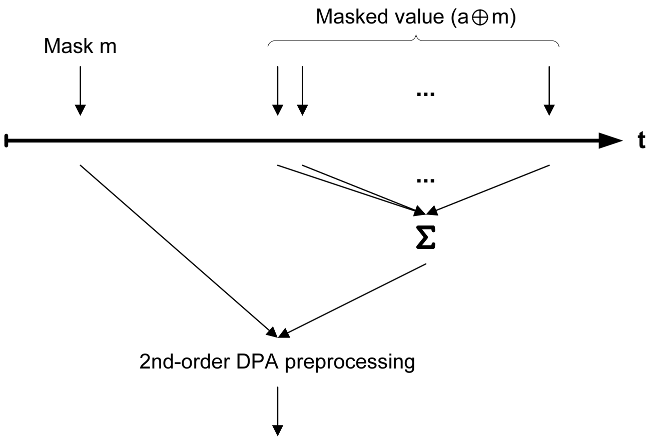


Fig. 4. Information extraction from power traces in a "classical" second-order DPA

For a randomization degree of $N = 1$, the correlation is about $\rho = 0.013$ for 8-bit masks and $\rho = 0.012$ for 32-bit masks. Doubling the randomization degree approximately halves the correlation coefficient.

5 Performance Estimation

The security evaluation of the last section has shown that there are powerful attacks which can break implementations even when they employ very sophisticated countermeasures. Under the assumptions of our analysis, one might be inclined to regard the use of software countermeasures as futile. Nevertheless there are scenarios, where a protected implementation might be desired, even if the provided protection is rather moderate:

- In a device with a fixed processor, the use of software countermeasures is likely to be the only available option. In some applications, a certain degree of implementation security could still be much better than none at all.
- The most powerful attacks used in our security evaluation might not be applicable due to other security measures of the device (e.g. limited number of AES encryption/decryptions, plaintext/ciphertext not selectable by the attacker, etc.).
- The device has some hardware countermeasures (e.g. noise generators) and the resistance against power-analysis should be amplified by the software countermeasures.

In order to provide performance estimations for different countermeasures, we have implemented AES-128 encryption with both masking and a scalable randomization. With the help of this implementation we have estimated the performance for several design options and degrees of randomization. First, we present the most important design decisions and implementation characteristics of our solution. Then we give the performance figures for interesting implementation variants regarding expected security level, speed, and memory requirements.

5.1 Features of Our Protected AES Implementation

Some basic design decisions for our 32-bit implementation are similar to the secure AES implementation for 8-bit microcontrollers presented in [5]. This mainly concerns the basic types of countermeasures (masking and randomization), the concept of randomized zones, etc. We assume the availability of a random number generator to provide mask values and randomization parameters.

The masking scheme requires six distinct byte masks as input. Two mask bytes are used to derive a masked S-box lookup table with input mask M and output mask M'. The four other bytes (denoted $M1$, $M2$, $M3$, and $M4$) mask each input column to the MixColumns transformation. The corresponding output masks can be derived by performing MixColumns on the mask values alone. More precisely, $M1$ to $M4$ are used as an input column for the MixColumns transformation, resulting in the output masks $M1'$, $M2'$, $M3'$, and $M4'$.

All operations which yield intermediate results depending on a relatively small portion of the key are executed in a randomized fashion. Randomization is achieved both by shuffling of operations as well as the addition of dummy operations. The processing of the AES State is shuffled so that each byte is processed at one of 16 moments in time with equal probability. Dummy operations are inserted as normal AES round transformations, but work on a random State (*dummy State*). The processing of the *genuine State* is randomly embedded in between the processing of several dummy States. The parts of the encryption where execution is randomized are denoted as *randomized zones*. The randomized zone at the beginning of AES encryption reaches up to and including the SubBytes operation of round 2, while the randomized zone at the end starts with SubBytes in round 9. Figure 5 gives a general overview of the program flow for the AES implementation and shows the masks on the State as well as the randomized transformations.

Randomization of operations is costly in terms of performance. Therefore it is desirable to keep the randomized zones as short as possible. In our implementation we

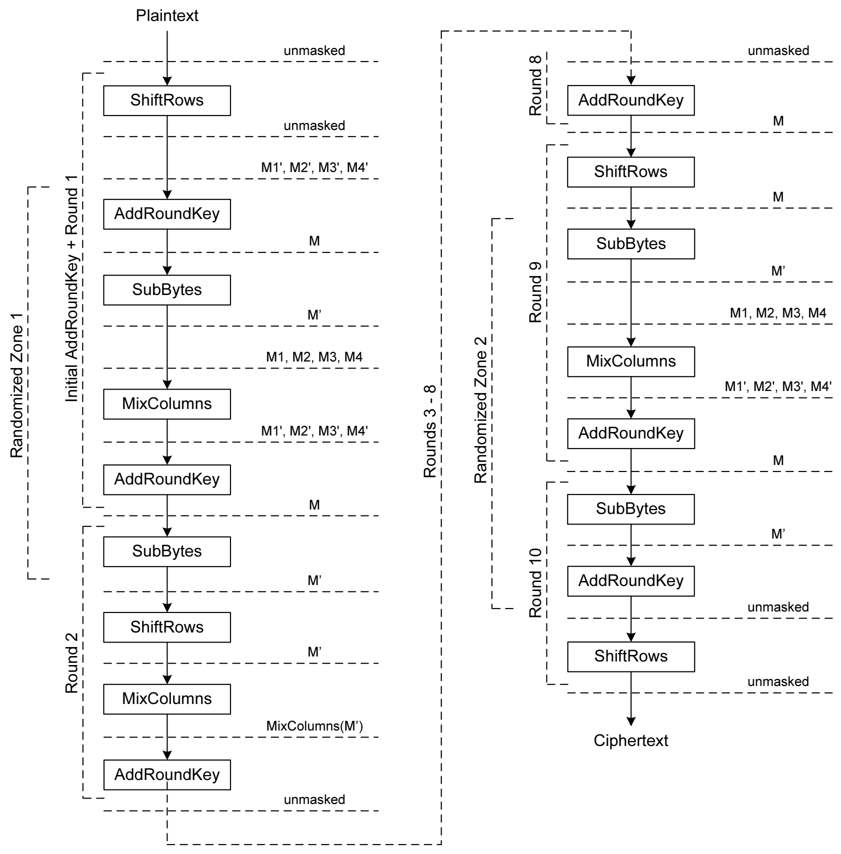


Fig. 5. Program flow of masked and randomized AES encryption

Table 1. Performance and RAM requirements of AES-128 encryption implementations

Countermeasures	**Pure Software**	**ISE**	**Memory (RAM)**
	cycles	cycles	bytes
None	1,637	196	176
1SB_WR	$6,465+1,888N$	$2,023+1,028N$	476
4SB_WR	$14,958+1,888N$	$3,631+1,028N$	1248
4SB_SR	$15,332+2,208N$	$3,978+1,348N$	1388

have reordered the round transformations, so that ShiftRows is not included in the randomization. This reordering requires the first and last round key to be transformed with ShiftRows or InvShiftRows.

In order to reduce the overhead for masking, the AddRoundKey operation is used for remasking whenever possible. This requires masks to be applied on some of the round keys. The masks on these round keys must be renewed whenever the masks change. When the masks are changed for each AES encryption—which is the ideal case—then it would be equally efficient to change the mask explicitly during the AES encryption.

In our implementation, the rounds 3 to 8 are not masked. AddRoundKey of round 2 removes the masks from the State, and AddRoundKey of round 8 masks the State again. All unmasked intermediate values have therefore been subjected to three AddRoundKey transformations and depend on sufficiently many key bytes, to prevent an efficient DPA attack. The advantage of the unmasked inner rounds is that the AES instruction set extensions can be fully used.

The randomization follows the concepts described in Section 3.3. In the randomized zones, only values which depend on a single State byte are processed. This allows for a randomization degree of $(N+1)\cdot 16$, where N is the number of dummy States.

5.2 Performance Figures

Table 1 contains the execution times and RAM requirements for several implementations of AES-128 encryption with masking and randomization countermeasures. The performance figures are given for the case without instruction set extensions (pure software) as well as with instruction set extensions (ISE). The RAM requirements for a specific implementation is always the same for both cases. The cycle counts are given in dependence on the number of dummy States (N).

We have given performance figures for three protected implementations, which employ both masking and randomization countermeasures. The cycle counts include all overhead when the masks are refreshed for each new encryption. The first implementation (1SB_WR) uses 1 masked S-box and a weak randomization (weak in the sense defined in Section 3.4). The second implementation (4SB_WR) is similar, but uses 4 masked S-boxes. The last implementation (4SB_SR) has a strong randomization.

For comparison, the performance figures of an unprotected implementation, as stated in [18], are provided.

Table 2 gives a complete analysis of the security/performance trade-off for the three protected implementations. Note that SW denotes the software implementation, while ISE denotes the respective implementation with instruction set extensions. The table

Table 2. Analysis of the security/performance trade-off

Implementation	Performance	BM	2W	WR	W2	max(ρ)
1SB_WR (SW), $N = 0$	6,465	-0.14	0.06	0.06	0.03	-0.14
1SB_WR (SW), $N = 3$	12,129	-0.07	0.03	0.03	< 0.01	-0.07
1SB_WR (SW), $N = 5$	15,905	-0.06	0.02	0.02	< 0.01	-0.06
1SB_WR (SW), $N = 11$	27,233	-0.04	0.02	0.02	< 0.01	-0.04
1SB_WR (ISE), $N = 0$	2,023	-0.14	0.06	0.06	0.03	-0.14
4SB_WR (ISE), $N = 0$	3,631	-0.05	0.03	0.06	0.02	0.06
4SB_SR (ISE), $N = 0$	3,978	-0.05	0.03	N/A	0.02	-0.05
4SB_SR (ISE), $N = 1$	5,326	-0.04	0.02	N/A	0.01	-0.04
4SB_SR (ISE), $N = 3$	8,022	-0.03	0.01	N/A	< 0.01	-0.03
4SB_SR (ISE), $N = 5$	10,718	-0.02	0.01	N/A	< 0.01	-0.02
4SB_SR (ISE), $N = 11$	18,806	-0.01	< 0.01	N/A	< 0.01	-0.01

lists the estimated correlation coefficients for the four attacks presented in Section 3.4: Biased mask attack (BM), combined second-order DPA and windowing attack (2W), weak randomization attack (WR), and "classical" second-order DPA attack on windowed traces (W2). The maximum correlation coefficient is listed in the last column.

For the pure software implementation, the biased-mask attack (BM) is the most powerful one. In software, the only option is to increase the randomization degree N. But the correlation coefficient only decreases very slowly with rising N. When instruction set extensions are available, we can work exclusively with 32-bit masks if we use four masked S-boxes instead of one (4SB_WR). In that case, the attack exploiting the weak randomization becomes the most efficient one. To counteract, we use the implementation with strong randomization (4SB_SR), which makes this attack inapplicable. Then the biased-mask attack becomes again the most effective one. With heavy randomization ($N = 11$), the correlation coefficient can be pushed down to $\rho = -0.01$. This corresponds to an increase of the security level by four orders of magnitude in comparison to an unprotected implementation. This comes at the price of an execution time, which is increased by two orders of magnitude (cf. Table 1). Compared to the unprotected pure software implementation, the execution time is increased by one order of magnitude.

6 Conclusions

In this paper we have provided a thorough analysis of power analysis countermeasures in software in the face of state-of-the-art attacks. We have concentrated on 32-bit embedded processors, but most of the results could also be applied to 8-bit and 16-bit processors. By means of an AES implementation we have shown the impact of power analysis countermeasures on the performance and RAM requirements. When restricted to the original instruction set architecture, the attainable degree of protection of our protected implementation is increased by three orders of magnitude. If the processor is equipped with custom instructions for AES, then a protection level of four orders of magnitude is achievable. But the performance penalty is rather high, so that it is probably not acceptable for all applications. As of now, no set of software countermeasures seems suited to offer a reasonable degree of protection at a negligible overhead.

Future Work. The use of existing instruction set extensions for AES is not sufficient to support power analysis countermeasures. A promising approach which we will investigate in the future is to enhance the extensions with hardware countermeasures.

Acknowledgements. The research described in this paper has been supported by the Austrian Science Fund (FWF) under grant number P18321-N15 ("Investigation of Side-Channel Attacks") and P16952-N04 ("Instruction Set Extensions for Public-Key Cryptography"), by the European Commission under grant number FP6-IST-033563 (Project SMEPP) and, in part, by the European Commission through the IST Programme under contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Çetin Kaya Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.
2. J. Blömer, J. Guajardo, and V. Krummel. Provably Secure Masking of AES. In H. Handschuh and M. A. Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2005.
3. S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
4. C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Çetin Kaya Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.
5. C. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2006*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2006.
6. K. Itoh, M. Takenaka, and N. Torii. DPA Countermeasure Based on the Masking Method. In K. Kim, editor, *Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings*, volume 2288 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2002.
7. J. Jaffe. More Differential Power Analysis: Selected DPA Attacks, June 2006. Presented at ECRYPT Summerschool on Cryptographic Hardware, Side Channel and Fault Analysis.
8. M. Joye, P. Paillier, and B. Schoenmakers. On Second-Order Differential Power Analysis. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

9. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
10. S. Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In P. J. Lee and C. H. Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2003.
11. S. Mangard. Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In T. Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.
12. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. ADIS Book Series. Springer, 2007. ISBN 0-387-30857-1.
13. E. Oswald and S. Mangard. Template Attacks on Masking—Resistance is Futile. In M. Abe, editor, *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*, volume 4377 of *Lecture Notes in Computer Science*, pages 243–256. Springer, February 2007.
14. E. Oswald, S. Mangard, C. Herbst, and S. Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.
15. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption, 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.
16. E. Oswald and K. Schramm. An Efficient Masking Scheme for AES Software Implementations. In J. Song, T. Kwon, and M. Yung, editors, *Information Security Applications, 6th International Workshop, WISA 2005, Jeju Island, Korea, August 22-24, 2005, Revised Selected Papers*, volume 3786 of *Lecture Notes in Computer Science*, pages 292–305. Springer, 2006.
17. F.-X. Standaert, E. Peeters, and J.-J. Quisquater. On the Masking Countermeasure and Higher-Order Power Analysis Attacks. In *International Symposium on Information Technology: Coding and Computing (ITCC 2005), 4-6 April 2005, Las Vegas, Nevada, USA, Proceedings*, volume 1, pages 562–567. IEEE Computer Society, April 2005.
18. S. Tillich and J. Großschädl. Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 270–284. Springer, 2006.
19. J. Waddle and D. Wagner. Towards Efficient Second-Order Power Analysis. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.

Constant-Round Authenticated Group Key Exchange with Logarithmic Computation Complexity*

Junghyun Nam[1], Juryon Paik[2], Ung Mo Kim[2], and Dongho Won[2,**]

[1] Department of Computer Science, Konkuk University, Korea
jhnam@kku.ac.kr
[2] Department of Computer Engineering, Sungkyunkwan University, Korea
juryon.paik@gmail.com, umkim@ece.skku.ac.kr, dhwon@security.re.kr

Abstract. Protocols for group key exchange (GKE) are cryptographic algorithms that describe how a group of parties communicating over a public network can come up with a common secret key. Due to their critical role in building secure multicast channels, a number of GKE protocols have been proposed over the years in a variety of settings. However despite many impressive achievements, it still remains a challenging problem to design a secure GKE protocol which scales very well for large groups. Our observation is that all constant-round authenticated GKE protocols providing forward secrecy thus far are not fully scalable, but have a computation complexity that scales only linearly in group size. Motivated by this observation, we propose a new and the first forward-secure authenticated GKE protocol that achieves both constant round complexity and logarithmic computation complexity. In particular, our GKE protocol is fully scalable in all key metrics when considered in the context of a broadcast network. The scalability of the protocol is achieved by using a complete binary tree structure combined with a so-called "nonce-chained authentication technique". Besides its scalability, our protocol features provable security against active adversaries under the decisional Diffie-Hellman assumption. We provide a rigorous proof of security for the protocol in a well-defined formal model of communication and adversary capabilities. The result of the current work means that forward-secure generation of session keys even for very large groups can be now done both securely and efficiently.

Keywords: Cryptography, group key exchange, scalability, binary tree, nonce-chained authentication, provable security.

1 Introduction

The primary goal of cryptography is to provide a means for communicating confidentially and with integrity over a public channel. Roughly speaking,

* This work was supported by the Korean Ministry of Information and Communication under the Information Technology Research Center (ITRC) support program supervised by the Institute of Information Technology Assessment (IITA).
** Corresponding author.

confidentiality ensures that communications and messages are kept secret between authorized parties, and integrity guarantees that any unauthorized modifications to the transferred data will be detected. In practice, these two main security properties are best achieved with key exchange protocols which allow the parties communicating over an insecure network to establish a common secret key called a *session key*. Typically, the communicating parties, who want confidentiality and integrity, first generate a session key by running an appropriate key exchange protocol and then use this key together with standard cryptographic algorithms for message encryption and authentication. Thus, the problem of establishing confidential and integrity-preserving communication is commonly reduced to the problem of getting a right protocol for session key generation. Needless to say, a tremendous amount of research effort has been devoted to the design and analysis of key exchange protocols in a variety of different settings (e.g., [19,24,37,8,25] and their follow-ups).

The first priority in designing a key exchange protocol is placed on ensuring the security of session keys to be established by the protocol. Even if it is computationally infeasible to break the cryptographic algorithms used, the whole system becomes vulnerable to all manner of attacks if the keys are not securely established. But unfortunately, the experience has shown that the design of secure key exchange protocols is notoriously difficult; there is a long history of protocols for this domain being proposed and later found to be flawed (see [16] for a comprehensive list of examples). Thus, key exchange protocols must be subjected to a thorough and systematic scrutiny before they are deployed into a public network, which might be controlled by an adversary. This concern has prompted active research on formal models [6,7,40,5,11,15,2,28] for security analysis of key exchange protocols, and highlighted the importance of proofs of protocol security in a well-defined model. Although rigorously proving a protocol secure can often be a lengthy and complicated task, proofs are advocated as invaluable tools for obtaining a high level of assurance in the security of key exchange protocols [27,11,29,2,33,17].

Efficiency is another important consideration in designing key exchange protocols. In particular, it may become a critical practical issue in the group setting where quite a large number of parties are likely to get involved in session key generation. The efficiency of a group key exchange (GKE) protocol is typically measured with respect to communication cost as well as computation cost incurred by the protocol. Three common measures for gauging the communication cost of a protocol are (1) the *round complexity*, the number of rounds until the protocol terminates, (2) the *message complexity*, the maximum number of messages both sent and received per user in the protocol, and (3) the *bit complexity*, the maximum number of bits (i.e., the maximum combined length of messages) both sent and received per user in the protocol. In order for a GKE protocol to be scalable, it is desirable in many real-life applications that the protocol be able to complete in a constant number of rounds. The computation cost of a protocol is directly related to the *computation complexity* which we define as the maximum amount of computation done by a single user in the protocol.

By computation, we do not mean simple traverses of the identities of the protocol participants, but mean *any kinds of cryptographic operations* such as public-key and symmetric-key operations, modular arithmetic operations, hash function evaluations, etc. Although the above definitions of various complexities are largely based on those given in the full version of [29][1], there is a noteworthy difference in defining message and bit complexities. Our definitions for these complexities counts both the sent and received traffics whereas those in the full version of [29] considers only the sent one. We believe this modification provides a more accurate way to measure the communication efficiency of any distributed protocols.

Motivation. Efficient and secure generation of session keys for large groups is a difficult problem that needs more work to solve it. The difficulty of the problem is well indicated by the fact that it took nearly two decades before we got the first provably-authenticated GKE protocol [11] even with round complexity $O(n)$ in a group of size n. Still up to now, there are only a very limited number of constant-round protocols [9,29,30,21] carrying a claimed proof of security against active adversaries in a formal model. However, all these constant-round protocols suffer from the number of public-key operations that scales linearly in group size, and thus exhibit $O(n)$ computation complexity under the definition above. These best-known protocols are categorized as key agreement protocols, but the situation is not much different for authenticated key transport protocols [23,35,26]. Indeed, we are unaware of any, provably secure or not, authenticated GKE protocols achieving both constant round complexity and logarithmic computation complexity. The protocols of [23,35,9,26] requires one distinct user to perform $O(n)$ modular exponentiations or public-key encryptions. The other protocols from [29,30,21] is all a novel extension of the protocol (i.e., protocol 3) by Burmester and Desmedt [12], but commonly require each user to perform $O(n)$ signature verifications. For moderate size groups, these previous solutions are clearly appealing. But for large groups, many applications will likely demand a protocol whose computation complexity scales logarithmically with group size. It is this observation that prompted the present work aimed at designing an authenticated GKE protocol which scales very well for large groups.

Contribution. The result of this work is the first forward-secure authenticated GKE protocol that achieves $O(1)$ round complexity and $O(\log n)$ computation complexity. In Tables 1 and 2, we summarize the computation and communication requirements of our protocol and other authenticated GKE protocols [23,35,9,29].[2] (By the tables, we are not arguing that one is overall superior to another, but meant to provide an asymptotic analysis for comparing scalability of different protocols.) Like the protocols of [23,35], our GKE protocol is categorized as a key transport protocol. The protocol of [9][3] features optimal

[1] The full version of [29] is available at `http://www.cs.umd.edu/∼jkatz`

[2] Although the protocols from [30,21] may perform better in practice than the protocol of [29], they fall into the same category from the computation complexity perspective.

[3] We refer to [17] for a security enhancement to this protocol.

Table 1. Computation requirements of authenticated GKE protocols

	Exp	Sig/Dec	Ver/Enc	Div	Mul
Boyd-Nieto [9]		$O(1)/$	$/O(n)$		
Katz-Yung [29]	$O(1)$	$O(1)/$	$O(n)/$	$O(1)$	$O(n \log n)$
Hirose-Yoshida [23]	$O(n)$	$O(n)/$	$O(n)/$		$O(n)$
Mayer-Yung [35]		$O(1)/$	$O(n)/O(n)$		
Here	$O(\log n)$	$O(1)/$	$O(\log n)/$	$O(1)$	$O(\log n)$

Note. "Mayer-Yung [35]" refers to *a-MKT with Consistency 1* of [35].
Exp: the maximum number of modular exponentiations performed per user.
Sig/Dec: the maximum numbers of signature generations and public-key decryptions performed per user.
Ver/Enc: the maximum numbers of signature verifications and public-key encryptions performed per user.
Div: the maximum number of modular divisions performed per user.
Mul: the maximum number of modular multiplications performed per user.

Table 2. Communication requirements of authenticated GKE protocols

	Rounds	Messages		Bits	
		PtP	Broadcast	PtP	Broadcast
Boyd-Nieto [9]	1	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$
Katz-Yung [29]	3	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Hirose-Yoshida [23]	3	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Mayer-Yung [35]	4	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$
Here	3	$O(n)$	$O(\log n)$	$O(n)$	$O(\log n)$

Note. "Mayer-Yung [35]" refers to *a-MKT with Consistency 1* of [35].
Rounds: the number of communication rounds required to complete the protocol.
Messages: the maximum number of messages both sent and received per user.
Bits: the maximum number of bits both sent and received per user.
PtP: the point-to-point network model.
Broadcast: the broadcast network model.

round complexity [4], but lacks perfect forward secrecy [20]. As Table 1 shows, the maximum computation rate per user is bounded by $O(\log n)$ in our protocol, whereas this rate per user rises up to $O(n)$ in the other protocols. Thus from a theoretical point of view, our main contribution is to show the possibility of achieving logarithmic computation complexity in constructing forward-secure constant-round protocols for authenticated group key exchange. However, it is also important from a practical viewpoint to notice that for reasonable values of n, the actual computation in our protocol can be heavier than that in the other protocols.

Our result can be even stronger in a broadcast network model, where each message sent is assumed to be received by all parties in the network. In the broadcast model, our protocol distinguishes itself from the other protocols in that it achieves $O(\log n)$ message and bit complexities as shown in Table 2. (Recall that both the sent and received traffics are considered for estimating message and bit complexities.) Thus if we assume a broadcast network, our protocol can be regarded as the first forward-secure authenticated GKE protocol that not only achieves $O(1)$ round complexity but also bounds all other complexities (i.e., bit, message, and computation complexities) by $O(\log n)$.

Furthermore, our protocol is provably secure against a powerful active adversary under the decisional Diffie-Hellman assumption. We provide a rigorous proof of security for the protocol in a refinement of the standard security model [11,9,29,30,21]. From the standpoint of the adversary's capabilities, our security model is a unique combination of previous results from [11,10,2,36], which are in turn based on earlier work of Bellare, Pointcheval, and Rogaway [5]. In particular, our model maximizes the overall attacking ability of the adversary in two ways. Firstly, we allow the adversary to query the Test oracle as many times as it wants [2]. Secondly, we incorporate *strong corruption* [5] into the model by allowing the adversary to ask users to release any short-term and long-term secret information. A detailed discussion on this is deferred to Section 2. Our security proof of course captures important security notions of perfect forward secrecy and known key security [18]. In addition since security is proved in the strong corruption model, our protocol also guarantees that the release of short-term secrets used in some sessions does not jeopardize the security of other sessions.

Tree-Based Protocols. A number of GKE protocols, including ours, have leveraged a tree structure in order to provide better scalability. As is widely known, the protocols of Wallner et al. [41] and Wong et al. [42] are based on a logical tree of key encryption keys. These protocols make substantial progress towards scalable key management in very large groups, by reducing the cost of rekeying operations associated with group updates from $O(n)$ to $O(\log n)$. But, these group rekeying methods (and their many optimizations and extensions, e.g., [39]) fail to provide (perfect) forward secrecy, requiring long-term pairwise secure channels between a key server and all users.

The approach using logical key trees has been extended by Kim et al. [31,32] to the forward-secure case. Their protocols require no secure channels of any kind and offer distributed functionality. Later, Lee et al. [34] present a pairing-based variant of the TGDH protocol of [31]. All these works [31,32,34], however, provide no explicit treatment of key exchange for initial group formation, focusing only on key updates upon group membership changes.

Ren et al. [38] make use of a binary key tree in their generic construction where an authenticated GKE protocol is built upon any authenticated protocol for two-party key exchange. Barua et al. [3] and Dutta et al. [22] construct their protocols by combining a ternary tree structure with the one-round tripartite protocol of Joux [25]. Back in 1994, Burmester and Desmedt [12] also proposed a tree-based GKE protocol. This protocol (i.e., protocol 2 of [12]) seems to be the first GKE protocol utilizing a binary tree structure, and differs from all other protocols mentioned above in that there exists a bijective mapping between protocol participants and tree nodes. But, this protocol, in common with other protocols from [3,38,22], has round complexity $O(\log n)$, in contrast to $O(1)$ in our protocol.

After the first version of this paper was written, we became aware that in 1996, Burmester and Desmedt [13] presented a graph-based protocol called CKDS. The CKDS protocol (more precisely, the multicast version of CKDS) has a potential to achieve the same level of complexities as our protocol, in the sense that the minimum spanning tree of the graph it used could have a height of $O(\log n)$. But unlike our provably-authenticated protocol, this protocol assumes a passive adversary and justifies its security on purely heuristic grounds without providing no formal analysis of security.

2 Formal Setting

Any form of security analysis of a cryptographic construction should be preceded by clear definitions of its security goals and tools. In this section we provide such a preliminary formalism for group key exchange.

2.1 Communication and Adversary Model

Participants. Let $\mathcal{U}$ be a set of all users who are potentially interested in participating in a group key exchange protocol. The users in any subset of $\mathcal{U}$ may run the group key exchange protocol at any point in time to establish a session key. Each user may run the protocol multiple times either serially or concurrently, with possibly different groups of participants. Thus, at a given time, there could be many instances of a single user. We use Π_i^π to denote the π-th instance of user U_i. Before the protocol is executed for the first time, each user $U_i \in \mathcal{U}$ creates a long-term public/private key pair (PK_i, SK_i) by running a key generation algorithm $\mathcal{K}(1^\kappa)$. All instances of a user share the public/private keys of the user even if they participate in their respective sessions independently. Each private key is kept secret by its owner while the public keys of all users are publicized.

Partners. Intuitively, the *partners* of an instance is the set of all instances that should compute the same session key as the instance in an execution of the protocol. Like most of previous works, we use the notion of *session IDs* to define partnership between instances. Literally, a session ID (denoted as sid) is a unique identifier of a communication session. Following [14,15,28], we assume that session IDs are assigned and provided by some higher-level protocol. While this assumption is unnecessary in some protocols [9,29] which use only broadcast messages (in these protocols, a session ID can readily be defined as the concatenation of all message flows), it seems very useful in other protocols where some protocol messages are not broadcast and thus not all participants have the same view of a protocol run. Supporting this assumption, Katz and Shin [28] have recently made an interesting observation: *since a user may be running many instances of a key exchange protocol concurrently, users in practice need a means to identify the sessions to which incoming messages belong. Therefore, in some sense, pre-defined session IDs are implicit even in the models [11,9,29] that use a customized* *definition of session IDs.* We let $\mathcal{SID}$ be the algorithm used by the higher-level protocol to generate session IDs, and assume that $\mathcal{SID}$ is publicly available.

We also need the notion of *group IDs* to define partnership properly. A group ID (denoted as gid) is a set consisting of the identities of the users who intend to establish a session key among themselves. This notion is clearly natural because it is impossible (not even defined) to ever execute a group key exchange protocol without participants. Indeed, a group ID is a both necessary and important input to any protocol execution.

In order for an instance to start to run the protocol, we require that both sid and gid should be given as input to the instance. We use sid_i^π and gid_i^π to denote respectively sid and gid provided to instance Π_i^π. Note that gid_i^π should always include U itself. Session IDs and group IDs are public and hence available to the adversary. Indeed, the adversary in our model generates these IDs on its own; it generates a session ID by running $\mathcal{SID}$ and a group ID by choosing a subset of $\mathcal{U}$. However, there is an important point regarding the generation of session IDs. Our model does not require the adversary to be honest in generating session IDs. This means that the adversary may try to replay a session ID as many times as necessary for its attack, but only at its own risk. In other words, the uniqueness of a session ID is not guaranteed by the model but should be checked by users themselves.

An instance is said to *accept* when it successfully computes a session key in a protocol execution. Let acc_i^π be a boolean variable that evaluates to TRUE if Π_i^π has accepted, and FALSE otherwise. We say that any two instances Π_i^π and Π_j^ω are *partners* of each other, or equivalently, *partnered* iff all the following three conditions are satisfied: (1) $\mathsf{sid}_i^\pi = \mathsf{sid}_j^\omega$, (2) $\mathsf{gid}_i^\pi = \mathsf{gid}_j^\omega$, and (3) $\mathsf{acc}_i^\pi = \mathsf{acc}_j^\omega = \text{TRUE}$. We also say that two instances Π_i^π and Π_j^ω are *potential partners* of each other, or equivalently, *potentially partnered* iff the first two conditions above hold. We use pid_i^π and ppid_i^π to denote respectively the partners and the potential partners of the instance Π_i^π. Then it follows by the definitions that $\mathsf{pid}_i^\pi \subseteq \mathsf{ppid}_i^\pi$.

Adversary. The adversary in our model controls all message exchanges in the protocol and can ask participants to open up access to any secrets, either long-term or short-term. These capabilities of the adversary are modeled via various oracles to which the adversary is allowed to make queries. Unlike most previous models for group key exchange, we allow the adversary to query the Test oracle as many times as it wants[4]. This approach was recently suggested by Abdalla et al. [2] for password authenticated key exchange in the three-party setting and was also proved there to lead to a stronger model (for more details, see Lemmas 1 and 2 in Appendix B of [2]). What we found interesting is that allowing multiple Test queries is very useful in proving Theorem 1 which claims the security of our unauthenticated protocol against a passive adversary. We also strengthen the model by incorporating *strong corruption* [5] in which the adversary is allowed

[4] The model in [1] appears to be the first one for group key exchange that does not restrict the adversary to ask only a single Test query.

to ask user instances to release both short-term and long-term secrets. We treat strong corruption in a different manner than [5][5], and follow [36] in that we provide the adversary with an additional oracle called Dump which returns all short-term secrets used by an instance. Other oracles (Execute, Send, Reveal, and Corrupt) are as usual. In the following, we describe these relatively familiar oracles first and then Dump and Test oracles.

- Execute(sid, gid): This query prompts an honest execution of the protocol between a set of instances consisting of one instance for each user in gid, where the instances are all given the session ID sid and the group ID gid as their input. The transcript of the honest execution is returned to the adversary as the output of the query. This models passive attacks on the protocol.
- Send(Π_i^π, M): This query sends message M to instance Π_i^π. The instance Π_i^π proceeds as it would in the protocol upon receiving message M; the instance updates its state performing any required computation, and generates and sends out a response message as needed. The response message, if any, is the output of this query and is returned to the adversary. This models active attacks on the protocol, allowing the adversary to control at will all message flows between instances. A query of the form Send(Π_i^π, sid$\|$gid) prompts Π_i^π to initiate an execution of the protocol using session ID sid and group ID gid.
- Reveal(Π_i^π)[6]: This query returns to the adversary the session key held by Π_i^π. This oracle call captures the idea that exposure of some session keys should not affect the security of other session keys [18]. The adversary is not allowed to ask this query if it has already queried Test(Π_j^ω) for some Π_j^ω in pid_i^π (see below for the description of the Test oracle).
- Corrupt(U_i): This query returns to the adversary all long-term secret information of U_i including the private key SK_i[7]. This models the adversary's capability of breaking into a user's machine and gaining access to the long-term data set stored there. The adversary can issue this query at any time regardless of whether U_i is currently executing the protocol or not. This oracle call captures the idea that damage due to loss of U_i's long-term secrets should be restricted to those sessions where U_i will participate in the future.
- Dump(Π_i^π): This query returns *all short-term secrets* used in the past or currently being used by instance Π_i^π[8]. But, neither the session key computed by Π_i^π nor any long-term secrets of U_i are not returned. This models the adversary's capability to embed a Trojan horse or other form of malicious

[5] In the strong corruption model of [5], the Corrupt oracle returns both long-term and short-term secrets.

[6] While the Reveal oracle does not exist in the so-called ROR model of Abdalla et al. [2], it is available to the adversary in our model and is used to enable a modular approach in the security proof of our protocol. Anyway, allowing Reveal queries causes no harm, but rather provides more clarity.

[7] This definition of the Corrupt oracle corresponds to the so-called *weak corruption model* [5].

[8] This combined with the Corrupt oracle represents strong corruption.

code into a user's machine and then log all the session-specific information of the victim. The adversary is not allowed to ask this query if it has already queried $\mathsf{Test}(\Pi_j^\omega)$ for some Π_j^ω in ppid_i^π.

– $\mathsf{Test}(\Pi_i^\pi)$: This query provides a means of defining security. The output of this query depends on the hidden bit b that the Test oracle chooses uniformly at random from $\{0, 1\}$ during its initialization phase. The Test oracle returns the real session key held by Π_i^π if $b = 1$, or returns a random session key drawn from the key space if $b = 0$. The adversary is allowed to query the Test oracle as many times as necessary. But, the query can be asked only when instance Π_i^π is *fresh* (see Section 2.2 for the definition of freshness). All the queries to the oracle are answered using the same value of the hidden bit b that was chosen at the beginning. Namely, the keys returned by the Test oracle are either all real or all random.

Remark 1. The Dump oracle is essentially similar to the Session-state reveal oracle introduced in the model of Canetti and Krawczyk [14]. But as noted in [36], there is a technical difference between these two oracles. The Session-state reveal oracle can be queried only to obtain the internal state of an incomplete session, whereas the Dump oracle allows the adversary to obtain the recording of local history of an either incomplete or complete session.

Definition 1. *An adversary is called* active *iff it is allowed to access all the oracles described above, and called* passive *iff it is allowed to access all but the* Send *oracle.*

We represent the amount of queries used by an adversary as an ordered sequence of six non-negative integers, $Q = (q_{\text{exec}}, q_{\text{send}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{dump}}, q_{\text{test}})$, where the six elements refer to the numbers of queries that the adversary made respectively to its Execute, Send, Reveal, Corrupt, Dump, and Test oracles. We call this usage of queries by an adversary the *query complexity* of the adversary. Note that by Definition 1, the query complexity of a passive adversary is always represented as a sequence of the form $Q = (q_{\text{exec}}, 0, q_{\text{reve}}, q_{\text{corr}}, q_{\text{dump}}, q_{\text{test}})$.

2.2 Security Definition and Assumptions

Freshness. The notion of *freshness* is used in the definition of security to prohibit the adversary from asking the Test query against an instance whose session key (or some information about the key) can be exposed by trivial means.

Definition 2. *The instance Π_i^π is considered unfresh iff any of the following conditions hold:*

1. $\mathsf{acc}_i^\pi = \text{FALSE}$.
2. *The adversary queried* $\mathsf{Corrupt}(U_j)$ *for some U_j in* gid_i^π *before some instance in* ppid_i^π *accepts.*
3. *The adversary queried* $\mathsf{Dump}(\Pi_j^\omega)$ *for some Π_j^ω in* ppid_i^π*.*
4. *The adversary queried* $\mathsf{Reveal}(\Pi_j^\omega)$ *or* $\mathsf{Test}(\Pi_j^\omega)$ *for some Π_j^ω in* pid_i^π*.*

All other instances are considered fresh.

Remark 2. By "$\mathsf{Test}(\Pi_j^\omega)$" in the fourth condition of Definition 2, we require that for each different set of partners, the adversary should access the Test oracle only once. One may think that this restriction weakens the ability of the adversary. However this is not the case because when all information on partnering is public, obtaining the same data multiple times (from the instances partnered together) is no different than obtaining it once.

Security. The security of a group key exchange protocol P against an adversary $\mathcal{A}$ is defined in terms of the probability that $\mathcal{A}$ succeeds in distinguishing random session keys from real session keys established by the protocol P. That is, the adversary $\mathcal{A}$ is considered successful in attacking P if it breaks the semantic security of session keys generated by P. This notion of security is defined in the context of the following two-stage game, where the goal of adversary $\mathcal{A}$ is to correctly guess the value of the hidden bit b chosen by the Test oracle.

– **Stage 1:** $\mathcal{A}$ makes any allowed oracle queries at will as many times as it wishes.
– **Stage 2:** Once $\mathcal{A}$ decides that Stage 1 is over, it outputs a bit b' as a guess for the value of the hidden bit b used by the Test oracle. $\mathcal{A}$ wins the game if $b = b'$.

In the game above, the adversary can keep querying the oracles even after it asked some Test queries. However, when there was the query $\mathsf{Test}(\Pi_i^\pi)$ asked, the adversary is prohibited from querying $\mathsf{Dump}(\Pi_j^\omega)$ for some $\Pi_j^\omega \in \mathsf{ppid}_i^\pi$ and from querying $\mathsf{Reveal}(\Pi_j^\omega)$ for some $\Pi_j^\omega \in \mathsf{pid}_i^\pi$. This restriction reflects the fact that the adversary can win the game unfairly by using the information obtained via the query $\mathsf{Dump}(\Pi_j^\omega)$ or $\mathsf{Reveal}(\Pi_j^\omega)$.

Given the game above, the advantage of $\mathcal{A}$ in attacking the protocol P is defined as $\mathsf{Adv}_P(\mathcal{A}) = |2 \cdot \Pr[b = b'] - 1|$. Note that this definition is equivalent to say that the advantage of $\mathcal{A}$ is the difference between the probabilities that $\mathcal{A}$ outputs 1 in the following two experiments constituting the game: the *real experiment* where all queries to the Test oracle are answered with the real session key, and the *random experiment* where all Test queries are answered with a random session key. Thus, if we denote the real and the random experiments respectively as $\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A})$ and $\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A})$, the advantage of $\mathcal{A}$ can be equivalently defined as $\mathsf{Adv}_P(\mathcal{A}) = |\Pr[\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A}) = 1]|$, where the outcomes of the experiments is the bit output by $\mathcal{A}$.

We say that the group key exchange protocol P is *secure* if $\mathsf{Adv}_P(\mathcal{A})$ is negligible for all probabilistic polynomial time adversaries $\mathcal{A}$. To quantify the security of protocol P in terms of the amount of resources expended by adversaries, we let $\mathsf{Adv}_P(t, Q)$ denote the maximum value of $\mathsf{Adv}_P(\mathcal{A})$ over all $\mathcal{A}$ with time complexity at most t and query complexity at most Q.

Decisional Diffie-Hellman (DDH) Assumption. Let $\mathbb{G}$ be a cyclic (multiplicative) group of prime order q. Since the order of $\mathbb{G}$ is prime, all the elements of $\mathbb{G}$, except 1, are generators of $\mathbb{G}$. Let g be a random fixed generator of $\mathbb{G}$ and let x, y, z be randomly chosen elements in $\mathbb{Z}_q^*$ where $z \neq xy$.

Informally stated, the DDH problem for $\mathbb{G}$ is to distinguish between the distributions of (g^x, g^y, g^{xy}) and (g^x, g^y, g^z), and the DDH assumption is said to hold in $\mathbb{G}$ if it is computationally infeasible to solve the DDH problem for $\mathbb{G}$. More formally, we define the advantage of $\mathcal{D}$ in solving the DDH problem for $\mathbb{G}$ as $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{D}) = |\Pr[\mathcal{D}(\mathbb{G}, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(\mathbb{G}, g, g^x, g^y, g^z) = 1]|$. We say that the DDH assumption holds in $\mathbb{G}$ (or equivalently, the DDH problem is hard in $\mathbb{G}$) if $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{D})$ is negligible for all probabilistic polynomial time algorithms $\mathcal{D}$. We denote by $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)$ the maximum value of $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{D})$ over all $\mathcal{D}$ running in time at most t.

Signature Schemes. Let $\Sigma = (\mathsf{Kgen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme, where Kgen is the key generation algorithm, Sign is the signature generation algorithm, and Vrfy is the signature verification algorithm. Let $\mathsf{Succ}_{\Sigma}(\mathcal{A})$ denote the probability that $\mathcal{A}$ succeeds in generating an existential forgery under adaptive chosen message attack. We say that a signature scheme Σ is secure if $\mathsf{Succ}_{\Sigma}(\mathcal{A})$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$. We use $\mathsf{Succ}_{\Sigma}(t)$ to denote the maximum value of $\mathsf{Succ}_{\Sigma}(\mathcal{A})$ over all $\mathcal{A}$ running in time at most t.

3 A Scalable Protocol for Unauthenticated Group Key Exchange

This section presents a new group key exchange protocol called SKE (Scalable Key Exchange). Let $\mathcal{G} = \{U_1, U_2, \ldots, U_n\}$ be a set of n users wishing to establish a session key among themselves. As stated in the Introduction, our goal is to design a forward-secure GKE protocol with round complexity $O(1)$ and computation complexity $O(\log n)$. Towards the goal, we arrange the users in a complete binary tree where all the levels, except perhaps the last, are full; while on the last level, any missing nodes are to the right of all the nodes that are present. Fig. 1 shows an example of a complete binary tree of height 3 with 6 leaves and 6 internal nodes. Users in $\mathcal{G}$ are placed at nodes in a straightforward way that U_i has U_{2i} as its left child and U_{2i+1} as its right child. Let N_i denote the node at which U_i is positioned and let $\mathcal{G}_i$ denote the subgroup consisting of all users located in the subtree rooted at node N_i. Each internal node N_i is associated with a node key k_i. In the protocol, the node key k_i is first generated by U_i and then shared as the subgroup key among the users in $\mathcal{G}_i$. Accordingly, k_1 serves as the group key (i.e., session key) shared by all users in $\mathcal{G}$.

3.1 Description of SKE

In describing the protocol, we assume that the following public information has been fixed in advance and is known to all parties in the network: (1) the structure of the tree and the users' positions within the tree, (2) a cyclic multiplicative group $\mathbb{G}$ of prime order q, where the DDH assumption holds, and a generator g of $\mathbb{G}$, and (3) a function I mapping elements of $\mathbb{G}$ to elements of $\mathbb{Z}_q$. A standard

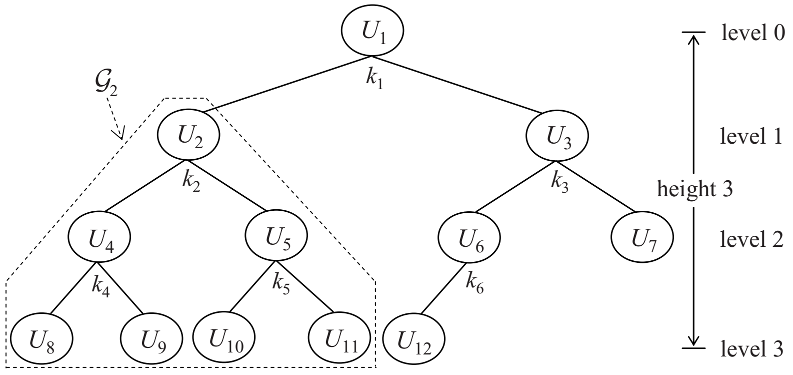


Fig. 1. A complete binary tree for $\mathcal{G} = \{U_1, \ldots, U_{12}\}$

way of generating $\mathbb{G}$ where the DDH assumption is assumed to hold is to choose two primes p, q such that $p = kq + 1$ for some small $k \in \mathbb{N}$ (e.g., $k = 2$) and let $\mathbb{G}$ be the subgroup of order q in $\mathbb{Z}_p^*$. For our purpose, we require that $I : \mathbb{G} \rightarrow \mathbb{Z}_q$ be bijective and (for any element in $\mathbb{G}$) efficiently computable. Whether there are appropriate bijections from $\mathbb{G}$ into $\mathbb{Z}_q$ depends on the group $\mathbb{G}$. If p is a safe prime (i.e., $p = 2q + 1$), then such a bijection I can be constructed as follows:

$$I(x) = \begin{cases} x & \text{if } x \leq q \\ p - x & \text{if } q < x < p. \end{cases}$$

The protocol SKE runs in two communication rounds.

Round 1: All users, except U_1 at the root, send a message to their parent as follows:

- Each user U_i at a leaf node chooses a random $r_i \in \mathbb{Z}_q$, computes $z_i = g^{r_i}$, and sends $M_i^1 = U_i \| 1 \| z_i$ to its parent.
- Each user U_i at an internal node chooses two random $s_i, t_i \in \mathbb{Z}_q$, computes $k_i = g^{s_i t_i}$, $r_i = I(k_i)$ and $z_i = g^{r_i}$, and sends $M_i^1 = U_i \| 1 \| z_i$ to its parent.

Meanwhile, U_1 chooses two random $s_1, t_1 \in \mathbb{Z}_q$ and computes $k_1 = g^{s_1 t_1}$.

Round 2: Each internal user U_i (including U_1) sends a message to its descendants (i.e., the users in $\mathcal{G}_i \setminus \{U_i\}$) as follows:

1. First, U_i computes $x_{2i} = z_{2i}^{s_i}$ and $y_{2i} = k_i x_{2i}^{-1}$. If U_i has the right child (this is the case for all internal users, except possibly for the last one), it also computes $x_{2i+1} = z_{2i+1}^{s_i}$ and $y_{2i+1} = k_i x_{2i+1}^{-1}$.
2. Then, U_i computes $w_i = g^{s_i}$ and sends $M_i^2 = U_i \| 2 \| w_i \| y_{2i} \| y_{2i+1}$ (or $M_i^2 = U_i \| 2 \| w_i \| y_{2i}$ if U_i has only the left child) to its descendants.

Key computation: Using messages from ancestors, each user $U_i \neq U_1$ computes every node key k_j on the path from the parent to the root as follows:

```
while $i \geq 2$
    do  $j \leftarrow \lfloor i/2 \rfloor$
        $k_j = y_i \cdot w_j^{r_i}$
        if $j > 1$
            then $r_j = I(k_j)$
        $i \leftarrow j$
```

Having derived the root node key k_1, all users in $\mathcal{G}$ simply set the session key K equal to k_1.

Consider, for example, the user U_{11} in Fig. 1. (For simplicity, let us exclude user identities and sequence numbers from consideration.) U_{11} sends $z_{11} = g^{r_{11}}$ to U_5 in the first round and receives $w_5\|y_{10}\|y_{11}$, $w_2\|y_4\|y_5$ and $w_1\|y_2\|y_3$ respectively from U_5, U_2 and U_1 in the second round. U_{11} then computes, in sequence, $k_5 = y_{11} \cdot w_5^{r_{11}}$, $r_5 = I(k_5)$, $k_2 = y_5 \cdot w_2^{r_5}$, $r_2 = I(k_2)$ and $k_1 = y_2 \cdot w_1^{r_2}$. Finally, U_{11} sets its session key to k_1.

It can be easily verified that the SKE protocol achieves the complexity bounds claimed in Section 1. Notice in SKE that the users at level ℓ perform about ℓ operations of any kind. This means that the maximum amount of computation done by a user scales linearly with the *height* of the tree, i.e., logarithmically with the number of users in $\mathcal{G}$. Hence, the computation complexity of SKE is $O(\log n)$ as claimed. The message and bit complexities of SKE in a broadcast network are also $O(\log n)$, since the maximum numbers of messages and bits both sent and received by a user increase linearly as the tree height grows. In a point-to-point network, the message and bit complexities rise up to $O(n)$ because the root user has to send a same message $n-1$ times. (Hereafter, for brevity of exposition, all statements regarding message and bit complexities assume a broadcast network.)

Of course, the SKE protocol is not authenticated, and is categorized as a key transport protocol because the session key is generated by one user (i.e., U_1) and then transferred to all other users. In the next section, we will show how to convert this unauthenticated protocol into an authenticated one without compromising the protocol's scalability.

3.2 Security Result for Protocol SKE

The following theorem presents our result on the security of protocol SKE. It says, roughly, that the group key exchange protocol SKE is secure against passive adversaries under the DDH assumption for $\mathbb{G}$.

Theorem 1. *Let $Q = (q_{\text{exec}}, 0, q_{\text{reve}}, q_{\text{corr}}, q_{\text{dump}}, q_{\text{test}})$. Then for any adversary with time complexity at most t and query complexity at most Q, its advantage in breaking the security of protocol* SKE *is upper bounded by:*

$$\mathsf{Adv}_{\text{SKE}}(t, Q) \leq q_{\text{test}} q_{\text{exec}} (2^{\lfloor \log |\mathcal{U}| \rfloor + 1} - 1) \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t'),$$

where $t' = t + O(|\mathcal{U}|q_{\text{exec}}t_{\text{SKE}})$ *and* t_{SKE} *is the time required for execution of protocol* SKE *by any party.*

At a high level, the proof of Theorem 1 proceeds by a mathematical induction on the height of the binary tree used in protocol SKE. Let SKE_h denote the protocol SKE but with the height of its input tree restricted to some fixed value $h > 0$. Namely, SKE_h is exactly the same as SKE, except that it can be run only for those groups such that $2^h \leq n < 2^{h+1}$. Then the basis step is to show that protocol SKE_1 is secure against passive adversaries. The induction step is to prove that for each $h \geq 1$, protocol SKE_{h+1} is secure against passive adversaries under the assumption of the security of protocol SKE_h against passive adversaries. The actual proof of the theorem is omitted here due to lack of space, and will be given in the full version of this paper.

4 A Scalable Protocol for Authenticated Group Key Exchange

Perhaps one of the most pleasing results of research on group key exchange is the one-round compiler presented by Katz and Yung [29] (in short, the KY compiler). The KY compiler shows how we can transform any group key exchange protocol secure against a passive adversary into one that is secure against an active adversary. It certainly is elegant in its scalability, usefulness, and proven security. The transformation itself is quite simple: it first adds an additional round for exchanging nonces among users and then lets all the messages of the original protocol be signed and verified with the nonces. In this section, we convert the unauthenticated protocol SKE into the authenticated protocol SKE^+ by using a modified version of the KY compiler.

4.1 Description of SKE^+

Let again $\mathcal{G}$ be the set of users wishing to establish a common session key. During the initialization phase of SKE^+, each user $U_i \in \mathcal{G}$ generates its long-term verification/signing keys (PK_i, SK_i) by running $\mathsf{Kgen}(1^\kappa)$ and makes the verification key PK_i public. Recall that each user U_i receives as input a pair of session and group IDs $(\mathsf{sid}_i, \mathsf{gid}_i)$ to start to run the protocol. Upon receiving $(\mathsf{sid}_i, \mathsf{gid}_i)$, U_i verifies that (1) sid_i is currently not in use for some active instance of it and (2) there is a bijective mapping between users in gid_i and nodes of the tree to be used. By checking the first condition, U_i is ensuring that the session ID is unique for all its active instances. This means that as far as security is concerned, reusing a session ID previously assigned to a closed session is legal and thus session IDs can be erased once their corresponding sessions have ended. If either of both conditions above is untrue, then U_i declines to participate in the protocol run associated with $(\mathsf{sid}_i, \mathsf{gid}_i)$. Otherwise, U_i performs the protocol SKE^+ as follows:

Round 1: Each user $U_i \in \mathcal{G}$ chooses a random nonce $\phi_i \in \{g^0, g^1, \ldots, g^{q-1}\}$ and sends $\tilde{M}_i^0 = U_i \| 0 \| \phi_i$ to its parent, sibling, descendants, and sibling's descendants. Let ncs_i be an ordered sequence defined as follows:

$$\mathsf{ncs}_i = \begin{cases} ((U_i, \phi_i), (U_{2i}, \phi_{2i}), (U_{2i+1}, \phi_{2i+1})) & \text{if } U_i \text{ has two children} \\ ((U_i, \phi_i), (U_{2i}, \phi_{2i})) & \text{if } U_i \text{ has only the left child} \\ ((U_i, \phi_i)) & \text{otherwise.} \end{cases}$$

Let $\varphi(i) = \lfloor i/2 \rfloor$. Then, after receiving all nonces (from its children, sibling, ancestors, and ancestors' siblings), each U_i computes the ordered sequences ncs_i, $\mathsf{ncs}_{\varphi(i)}$, $\mathsf{ncs}_{\varphi(\varphi(i))}$, ..., ncs_1 as defined above. Notice that the maximum number of nonces received by any single user is at most $2\lfloor \log n \rfloor$.

Round 2: This round proceeds like the first round of protocol SKE, except that users have to sign their outgoing messages:

- Each user $U_i \neq U_1$ computes z_i as specified in SKE and generates a signature $\sigma_i^1 = \mathsf{Sign}_{SK_i}(U_i \| 1 \| \, z_i \| \mathsf{sid}_i \| \mathsf{ncs}_{\varphi(i)} \| \mathsf{ncs}_{\varphi(\varphi(i))} \| \cdots \| \mathsf{ncs}_1)$. Then U_i sends $\tilde{M}_i^1 = U_i \| 1 \| z_i \| \sigma_i^1$ to its parent.
- The operation of U_1 is exactly the same as in SKE. That is, U_1 chooses two random $s_1, t_1 \in \mathbb{Z}_q$ and computes $k_1 = g^{s_1 t_1}$.

Round 3: All users operate as in Round 2 of SKE, but verifying the correctness of incoming messages and signing outgoing messages. We describe this round only for users who have both left and right children; users with left child only behave correspondingly.

- When user U_i receives $\tilde{M}_j^1 = U_j \| 1 \| z_j \| \sigma_j^1$ from U_j for $j = 2i$ and $j = 2i+1$, it first checks that $\mathsf{Vrfy}_{PK_j}(U_j \| 1 \| z_j \| \mathsf{sid}_i \| \mathsf{ncs}_i \| \mathsf{ncs}_{\varphi(i)} \| \cdots \| \mathsf{ncs}_1, \sigma_j^1) = 1$. If any of the verifications fail, U_i aborts the protocol without accepting (i.e., without computing a session key). Otherwise, U_i computes x_{2i}, y_{2i}, x_{2i+1}, y_{2i+1} and w_i as in protocol SKE, generates a signature $\sigma_i^2 = \mathsf{Sign}_{SK_i}(U_i \| 2 \| w_i \| y_{2i} \| y_{2i+1} \| \, \mathsf{sid}_i \| \mathsf{ncs}_i \| \mathsf{ncs}_{\varphi(i)} \| \cdots \| \mathsf{ncs}_1)$, and sends $\tilde{M}_i^2 = U_i \| 2 \| w_i \| y_{2i} \| y_{2i+1} \| \sigma_i^2$ to its descendants.

Key computation: Each user $U_i \neq U_1$, for all messages $\tilde{M}_j^2$ from its ancestors in the tree, checks that $\mathsf{Vrfy}_{PK_j}(U_j \| 2 \| w_j \| y_{2j} \| y_{2j+1} \| \mathsf{sid}_i \| \mathsf{ncs}_j \| \mathsf{ncs}_{\varphi(j)} \| \cdots \| \mathsf{ncs}_1, \sigma_j^2) = 1$. If any of the verifications fail, U_i terminates without accepting. Otherwise, U_i derives the root node key k_1 as in SKE and sets the session key K equal to k_1.

The above transformation from SKE to SKE$^+$ requires round complexity to be increased by a constant factor and the other (bit, message, and computation) complexities by a factor of $\log n$. The latter part of these increases is because the users at (or close to) leaves additionally have to receive about $2 \log n$ nonces and to perform about $\log n$ signature verifications. Consequently, the asymptotic bounds for the complexities remain unchanged between SKE and SKE$^+$. This unchanged scalability well explains why the KY compiler could not be directly applicable to SKE: as soon as we invoke the KY compiler on an arbitrary GKE protocol, the message and bit complexities of the resulting protocol rise at least up to $O(n)$ because the compiler requires each user to receive nonces from all other users.

The primary idea behind the KY compiler is to use the set of n nonces shared by users as a unique session identifier for all time points. Based on this idea, the KY compiler mandates the users to always include their nonce set in signing and verifying protocol messages. In this way, not only the freshness of any exchanged message is guaranteed but also no message can be relayed between user instances holding different sets of nonces. It is this observation that the KY compiler exploits to achieve provable security of the compiled protocol.

Our transformation, though similar in spirit as that by the KY compiler, reduces the number of nonces to be received per user to the order of $\log n$ while achieving provable security of the protocol SKE$^+$. The two main observations underlying this result are that: (1) at a given point of time, each pre-defined session ID is unique for all concurrent runs of SKE$^+$ (see Section 2.1 for the justification of pre-defined session IDs) and (2) even with at most $O(\log n)$ nonces per user, SKE$^+$ is able to guarantee the freshness of the messages exchanged among users. The first observation is clear from the description of SKE$^+$; no two active instances of a user possess a same session ID. The second observation becomes quite obvious once we notice that there is a chain of nonces in SKE$^+$: for all $2 \leq i \leq n$, two ordered sequences ncs_i and $\mathsf{ncs}_{\varphi(i)}$ are linked by the common element ϕ_i. This chain of nonces enables each user U_i to verify the freshness of all messages from its ancestors, even when those messages are not signed with ϕ_i. In other words, the use of the nonce chain ensures that no singed message is replayed between two sessions even with a same session ID. We call this technique *nonce-chained authentication*. These two observations, taken together, suggest that a pre-defined session ID combined with the nonce-chained authentication technique serves as a unique session identifier for all time points and thereby obviates the need for each user to receive n nonces.

4.2 Security Result for Protocol SKE$^+$

Here we claim that the group key exchange protocol SKE$^+$ is secure against active adversaries under the security of protocol SKE against passive adversaries. The following theorem makes this claim precise.

Theorem 2. *Let $Q = (q_{\text{exec}}, q_{\text{send}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{dump}}, q_{\text{test}})$ and $Q' = (q_{\text{exec}} + q_{\text{send}}/2, 0, q_{\text{reve}}, q_{\text{corr}}, q_{\text{dump}} + q_{\text{send}}/2, q_{\text{test}})$. For any adversary with time complexity at most t and query complexity at most Q, its advantage in breaking the security of protocol SKE$^+$ is upper bounded by:*

$$\mathsf{Adv}_{\text{SKE}^+}(t, Q) \leq \mathsf{Adv}_{\text{SKE}}(t', Q') + |\mathcal{U}| \cdot \mathsf{Succ}_{\Sigma}(t') + \frac{q_{\text{send}}^2 + q_{\text{exec}}q_{\text{send}}}{|\mathbb{G}|},$$

where $t' = t + O(|\mathcal{U}|q_{\text{exec}}t_{\text{SKE}^+} + q_{\text{send}}t_{\text{SKE}^+})$ and t_{SKE^+} is the time required for execution of SKE$^+$ by any party.

Proof Idea. We can prove the theorem by finding a reduction from the security of protocol SKE$^+$ to the security of protocol SKE. Assuming an active adversary $\mathcal{A}^+$ who attacks protocol SKE$^+$, we construct a passive adversary $\mathcal{A}$ that uses $\mathcal{A}^+$ in its attack on SKE. As in a typical reductionist approach, the adversary $\mathcal{A}$ simply runs $\mathcal{A}^+$ as a subroutine and answers the oracle queries of $\mathcal{A}^+$ on its own. The idea in constructing $\mathcal{A}$ is to use the fact that in attacking SKE$^+$, the adversary $\mathcal{A}^+$ is able to relay messages only between user instances with the same session ID and the matching nonce sequences. Based on this idea, the adversary $\mathcal{A}$ obtains a transcript T of SKE for each unique combination of session ID and nonce sequences by calling its own $\mathsf{Execute}$ oracle, and generates a transcript T^+ of SKE$^+$ by patching T with appropriate signatures. $\mathcal{A}$ then use the messages of T^+ in answering $\mathcal{A}^+$'s Send queries directed to user instances which have the same session ID and nonce sequences as used in generating T^+. In this way, $\mathcal{A}^+$ is limited to sending messages already contained in T^+, unless signature forgery and nonce repetition occur. In essence, $\mathcal{A}$ is ensuring that $\mathcal{A}^+$'s capability of attacking protocol SKE$^+$ is demonstrated only on the session key associated with the patched transcript T^+ and thus is translated directly into the capability of attacking protocol SKE. Due to space limitations here, we will provide the proof of the theorem in the full version of this paper.

References

1. M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-based group key exchange in a constant number of rounds. *9th International Workshop on Practice and Theory in Public Key Cryptography (PKC '06)*, LNCS vol. 3958, pp. 427–442, 2006.
2. M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. *8th International Workshop on Practice and Theory in Public Key Cryptography (PKC '05)*, LNCS vol. 3386, pp. 65–84, 2005.
3. R. Barua, R. Dutta, and P. Sarkar. Extending Joux's protocol to multi party key agreement. *Progress in Cryptology – INDOCRYPT '03*, LNCS vol. 2904, pp. 205–217, 2003.
4. K. Becker and U. Wille. Communication complexity of group key distribution. *5th ACM Conference on Computer and Communications Security (CCS '98)*, pp. 1–6, 1998.
5. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. *Advances in Cryptology – EUROCRYPT '00*, LNCS vol. 1807, pp. 139–155, 2000.
6. M. Bellare and P. Rogaway. Entity authentication and key distribution. *Advances in Cryptology – CRYPTO '93*, LNCS vol. 773, pp. 232–249, 1993.
7. M. Bellare and P. Rogaway. Provably secure session key distribution — the three party case. *27th ACM Symposium on Theory of Computing (STOC '95)*, pp. 57–66, 1995.
8. S. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. *1992 IEEE Symposium on Security and Privacy*, pp. 72–84, 1992.

9. C. Boyd and J. Nieto. Round-optimal contributory conference key agreement. *6th International Workshop on Practice and Theory in Public Key Cryptography (PKC '03)*, LNCS vol. 2567, pp. 161–174, 2003.
10. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. *Advances in Cryptology – ASIACRYPT '02*, LNCS vol. 2501, pp. 497–514, 2002.
11. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. *8th ACM Conference on Computer and Communications Security (CCS '01)*, pp. 255–264, 2001.
12. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. *Advances in Cryptology – EUROCRYPT '94*, LNCS vol. 950, pp. 275–286, 1995.
13. M. Burmester and Y. Desmedt. Efficient and secure conference-key distribution. *1996 International Workshop on Security Protocols*, LNCS vol. 1189, pp. 119–129, 1997.
14. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. *Advances in Cryptology – EUROCRYPT '01*, LNCS vol. 2045, pp. 453–474, 2001.
15. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. *Advances in Cryptology – EUROCRYPT '02*, LNCS vol. 2332, pp. 337–351, 2002.
16. K.-K. R. Choo. Provably-secure mutual authentication and key establishment protocols lounge. 2006. Available at `http://sky.fit.qut.edu.au/∼choo/lounge.html`.
17. K.-K. Choo, C. Boyd, and Y. Hitchcock. Errors in computational complexity proofs for protocols. *Advances in Cryptology – ASIACRYPT '05*, LNCS vol. 3788, pp. 624–643, 2005.
18. D. Denning and G. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, vol. 24, no. 8, pp. 533–536, 1981.
19. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
20. W. Diffie, P. Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.
21. R. Dutta and R. Barua. Constant round dynamic group key agreement. *8th International Conference on Information Security (ISC '05)*, LNCS vol. 3650, pp. 74–88, 2005.
22. R. Dutta, R. Barua, and P. Sarkar. Provably secure authenticated tree based group key agreement. *6th International Conference on Information and Communications Security (ICICS '04)*, LNCS vol. 3269, pp. 92–104, 2004.
23. S. Hirose and S. Yoshida. An authenticated Diffie-Hellman key agreement protocol secure against active attacks. *1st International Workshop on Practice and Theory in Public Key Cryptography (PKC '98)*, LNCS vol. 1431, pp. 135–148, 1998.
24. I. Ingemarsson, D. Tang, and C. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, vol. 28, no. 5, pp. 714–720, 1982.
25. A. Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, vol. 17, no. 4, pp. 263–276, 2003. A preliminary version was presented at *ANTS IV*.
26. B. Jung, S. Paeng, and D. Kim. Attacks to Xu-Tilborg's conference key distribution scheme. *IEEE Communications Letters*, vol. 8, no. 7, pp. 446–448, 2004.

27. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. *Advances in Cryptology – EUROCRYPT '01*, LNCS vol. 2045, pp. 475–494, 2001.
28. J. Katz and J. Shin. Modeling insider attacks on group key-exchange protocols. *12th ACM Conference on Computer and Communications Security (CCS '05)*, pp. 180–189, 2005.
29. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. *Advances in Cryptology – CRYPTO '03*, LNCS vol. 2729, pp. 110–125, 2003.
30. H.-J. Kim, S.-M. Lee, and D. Lee. Constant-round authenticated group key exchange for dynamic groups. *Advances in Cryptology – ASIACRYPT '04*, LNCS vol. 3329, pp. 245–259, 2004.
31. Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. *7th ACM Conference on Computer and Communications Security (CCS '00)*, pp. 235–244, 2000.
32. Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. *IFIP SEC '01*, pp. 229–244, 2001.
33. H. Krawczyk. HMQV: a high-performance secure Diffie-Hellman protocol. *Advances in Cryptology – CRYPTO '05*, LNCS vol. 3621, pp. 546–566, 2005.
34. S. Lee, Y. Kim, K. Kim, and D.-H. Ryu. An efficient tree-based group key agreement using bilinear map. *1st International Conference on Applied Cryptography and Network Security (ACNS '03)*, LNCS vol. 2846, pp. 357–371, 2003.
35. M. Mayer and M. Yung. Secure protocol transformation via "Expansion": From two-party to groups. *6th ACM Conference on Computer and Communications Security (CCS '99)*, pp. 83–92, 1999.
36. J. Nam, J. Lee, S. Kim, and D. Won. DDH-based group key agreement in a mobile environment. *Journal of Systems and Software*, vol. 78, no. 1, pp. 73–83, 2005.
37. E. Okamoto and K. Tanaka. Key distribution system based on identification information. *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 4, pp. 481–485, 1989.
38. K. Ren, H. Lee, K. Kim, and T. Yoo. Efficient authenticated key agreement protocol for dynamic groups. *5th International Workshop on Information Security Applications (WISA '04)*, LNCS vol. 3325, pp. 144–159, 2004.
39. A. Sherman and D. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 444–458, 2003.
40. V. Shoup. On formal models for secure key exchange. *Cryptology ePrint Archive*, Report 1999/012, 1999. Available at `http://eprint.iacr.org/`.
41. D. Wallner, E. Harder, and R. Agee. Key management for multicast: issues and architectures. RFC 2627, 1999.
42. C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, 2000. A preliminary version was presented at *ACM SIGCOMM '98*.

Preventing Collusion Attacks on the One-Way Function Tree (OFT) Scheme

Xuxin Xu[1], Lingyu Wang[1], Amr Youssef[1], and Bo Zhu[2]

[1] Concordia Institute for Information Systems Engineering
Concordia University
Montreal, QC H3G 1M8, Canada
{xuxin_xu,wang,youssef}@ciise.concordia.ca
[2] Center for Secure Information Systems
George Mason University
Fairfax, VA 22030-4444, USA
bzhu@gmu.edu

Abstract. The one-way function tree (OFT) scheme proposed by Balenson *et al.* is widely regarded as an efficient key management solution for multicast communication in large dynamic groups. Following Horng's claim that the original OFT scheme was vulnerable to a collusion attack, Ku *et al.* studied the collusion attack on OFT and proposed a solution to prevent the attack. The solution, however, requires to broadcast about $h^2 + h$ (h is the height of the key tree) keys for every eviction operation, whereas the original OFT scheme only requires about h keys. This modified OFT scheme thus loses a key advantage that the original OFT has over the logical key hierarchy (LKH) scheme, that is a halving in broadcast size. In this paper, we revisit collusion attacks on the OFT scheme. We generalize the examples of attacks given by Horng and Ku *et al.* to a generic collusion attack on OFT, and derive necessary and sufficient conditions for such an attack to exist. We then show a solution for preventing collusion attacks while minimizing the average broadcast size. Our simulation results show that the proposed solution allows OFT to outperform LKH in many cases.

1 Introduction

Multicast communications can greatly save bandwidth and sender resources in delivering data to groups of recipients. However, cryptographic key management schemes are required to ensure the confidentiality of a multicast communication. More specifically, *backward security* requires that a joining member cannot learn previous messages, and *forward security* requires that an evicted member cannot learn future messages. The adjective *perfect* can be added to the two properties, if they can be satisfied against an arbitrary number of colluding members [2].

To satisfy perfect forward and backward security, the group key must be changed whenever a member is added to or evicted from a group. The new key needs to be conveyed to all members at the minimum communication cost since the group is usually large and dynamically changing. Among other methods, the OFT (one-way function tree) scheme, originally proposed by Balenson *et al.*, is one of the most popular schemes for this purpose [1,2,3,4,17]. A key advantage of OFT over another popular method, the Logical Key Hierarchy (LKH) [6], is that OFT halves the number of bits broadcasted upon adding or evicting a member. Specifically, if a key has k bits and the key tree used by OFT and LKH has a height h, then the broadcast size of OFT is $hk + h$ bits, whereas that of LKH is $2hk + h$ bits. OFT achieves such a halving in broadcast size by deriving its key tree in a bottom-up manner, in contrast to LKH's top-down approach. Consequently, unlike the independently chosen keys in LKH, the keys in an OFT key tree are functionally dependent, and this functional dependency allows OFT to save half of the broadcasted bits.

Unfortunately, the same functional dependency among keys that brings OFT the reduced communication cost also subjects it to collusion attacks. Although OFT was claimed to achieve perfect forward and backward security [2], only the collusion among evicted members was considered. A collusion that includes current members was claimed to be uninteresting, because *a (current) member knows the group key*. However, the claim implicitly assumes the colluding members are trying to learn the current group key, which is not necessarily true. An evicted member may collude with a current member to learn group keys that were used after the former was evicted but before the latter joins the group. In this case, OFT will fail on both forward security and backward security. In 2002, Horng first showed an example of collusion attacks on OFT [16]. In 2003, Ku and Chen provided new attack examples to show that the two assumptions required by Horng's attack were actually not necessary conditions [11]. Ku and Chen also proposed a modified OFT scheme that is immune to the collusion attack. The solution, however, needs to broadcast $(h^2 + h)k$ bits on every member eviction (and hk bits on each member addition). Ku and Chen's scheme thus loses a key advantage which OFT has over LKH, that is a halving in broadcast size. Because their scheme requires a broadcast of quadratic size on evicting any member, it is only suitable for applications where member eviction is rare.

In this paper, we revisit collusion attacks on the OFT scheme. To better understand collusion attacks on OFT, we first generalize the examples of attacks given by Horng and Ku *et al.* to a generic attack. Instead of these examples of two or three members, we study the collusion among arbitrary number of evicted and joining members with arbitrary number of other, non-colluding members leaving or joining in between. Based on this understanding of the general attack, we derive necessary and sufficient conditions for a collusion attack on OFT to exist. These conditions reveal that the solution by Ku *et al.* is unnecessarily conservative. Their solution prevents potential collusion attacks by invalidating any knowledge that is brought out of the group by evicted members. However, our results show that such knowledge is not always useful to a joining member in colluding.

We study a different approach where such leaked knowledge is not immediately invalidated but is recorded by a key manager who is responsible for managing the group. When a member joins the group, the key manager then checks whether it is possible for this new member to collude with previously evicted members. If a potential collusion exists, the key manager will update keys as part of the joining operation such that the collusion becomes impossible. Because additional re-keying is performed only when a collusion is possible, this solution has the advantage of minimizing broadcast size.

Following the discussion of a straightforward stateful method that has an unacceptable storage requirement, we present a modified version of the method whose storage requirement is proportional to the size of the key tree. These methods pose no additional communication cost on evicting a member but may require more broadcasted bits when a member joins. We study the average performance of the scheme using experiments, and the result show that our scheme is more efficient than LKH in many cases.

The contribution of this paper is two fold. First, our study provides a better understanding of the collusion attack on the OFT scheme. The previous work by Horng and Ku *et al.* have only described specific examples of collusion attacks involving two or three colluding nodes but left the general case open [16]. Our results show exactly what can be computed by an arbitrary collection of joining and evicted nodes. Second, the solution we shall propose makes the OFT scheme secure against general collusion attacks while minimizing the communication overhead. Ku and Chen's solution renders OFT strictly less efficient than LKH, whereas our experimental results show that the solution in this paper enables OFT to outperform LKH in small to medium groups. The results also reveal that OFT's approach of using functionally dependent keys actually renders the scheme less efficient in large groups, if collusion attacks are to be prevented. The rest of the paper is organized as follows. Section 2 reviews the OFT scheme and examples of collusion attacks given by Horng and Ku *et al*. Section 3 generalizes these examples to a generic attack and derives the necessary and sufficient conditions for the collusion attack. Section 4 studies a solution that minimizes broadcast size while preventing collusion attack. Section 5 studies the performance of our solution through experiments. Section 6 concludes the paper and gives future directions.

2 Related Work

Various aspects of multicast security, including group key management, have been extensively studied, as surveyed in [5,7,8,12,13]. In [9], an architecture is provided for the management of cryptographic keys for multicast communications. Various security aspects, including ephemeral secrecy, long-term secrecy, and perfect forward secrecy, are outlined in [14]. Popular tree-based group key management schemes include the Logical Key Hierarchy (LKH) scheme [6,15,22], the One-way Function Tree (OFT) scheme [1,2,3,4,17], and the One-way Function Chain (OFC) scheme [8]. Unlike many solutions that depend on a trusted group controller, the authors in [10] propose a group key management scheme based on El Gamal, which only requires a partially trusted controller who does not need accesses to the communication keys. The solution in [19] integrates the one-way key derivation with key trees to reduce the communication overhead of rekeying operations. In the solution, the total number of encrypted keys transmitted during a rekeying operation is reduced by not sending new keys to those members who can derive the keys by themselves. The solution proposed in [20] inherits the architecture of the logical key tree algorithm but rekeys the group using a new algorithm. The batch rekeying scheme in [21] is based on one-way function tree and minimum exact covering.

The LKH scheme is shown to be immune to collusion attacks in [18]. On the other hand, Horng first showed that the OFT scheme is vulnerable to a collusion attack in [16].

This result was later revisited by Ku et al. in [11]. We first review the original OFT scheme in Section 2.1 and then review the examples of collusion attack on OFT given by Horng and Ku *et al.* in Section 2.2. In this paper, we do not address collusion attacks on the OFC scheme [8], which comprises an interesting future work.

2.1 The OFT Scheme

The original OFT scheme is an efficient key management scheme for large, dynamically changing groups [1,2,3,4,17]. A *key manager* maintains a balanced binary key tree for each group. The key trees are computed bottom up using a one-way function $g()$ and a concatenation function $f()$ as follows. First, each leaf node v is assigned a randomly chosen *node key* x_v, and a *blinded node key* is computed from the node key as $g(x_v)$. The node key of each interior node v is then computed by concatenating the blinded node keys of its left child $left(v)$ and right child $right(v)$ as: $x_v = f(g(x_{left(v)}), g(x_{right(v)}))$. For example, the key tree in the left hand side of Figure 1 can be constructed as $x_4 = f(g(x_8), g(x_9))$, $x_2 = f(g(x_4), g(x_5))$,$x_7 = f(g(x_14), g(x_15))$, $x_3 = f(g(x_6), g(x_7))$, and $x_1 = f(g(x_2), g(x_3))$.

Each group member is associated with a leaf node in the key tree, and is given its node key. For each node v on the path from that leaf node to the root, the group member is also given the blinded node key of v's sibling. The group member can thus compute the group key, that is the node key of the root[1]. For example, in the left hand side of Figure 1, a member Alice who is associated with the node 8 will be given the blinded keys $g(x_9)$, $g(x_5)$, and $g(x_3)$. Alice can then compute the group key as: $x_4 = f(g(x_8), y_9)$, $x_2 = f(g(x_4), y_5)$, and $x_1 = f(g(x_2), y_3)$.

A new member always joins at a leaf node closest to maintain the balance of the key tree. After the joining, the existing leaf node becomes the left child of a new interior node and is assigned a new node key. The right child is a new node associated with the joining member. The whole path from the interior node to the root will be updated due to the two new keys, and the updated blinded keys must be conveyed to those members who need them. For example, in Figure 1, the joining member Bob causes the existing node 5 to be split into two nodes, with each assigned a new node key. The node keys of node 5, node 2, and node 1 then need to be updated, and their blinded version will be broadcasted to the members who need them (for example node 8 and 9 will need the updated $g(x_5)$). A similar process applies to the other joining member Candy.

The eviction of a member is similar to the addition with following differences. The sibling of the node associated with the leaving member replaces its parent, and is assigned a new node key. Keys on the path leading that node to the root are then updated and their blinded versions are broadcasted, as in the case of addition. However, if the sibling of the leaving member is an interior node, then we cannot directly change its node key due to the functional dependency among keys. Instead, we need to change the node key of a leaf node in the subtree whose root is that interior node. For example, in Figure 1, the evicted member Alice causes the node 9 to replace the node 4. The node keys of nodes 4, node 2, and node 1 will then be updated, and their blinded version will be broadcasted to those who need them.

[1] In a later version of the scheme, the key used for communication is not the node key itself but is derived from the node key using another one-way function [2].

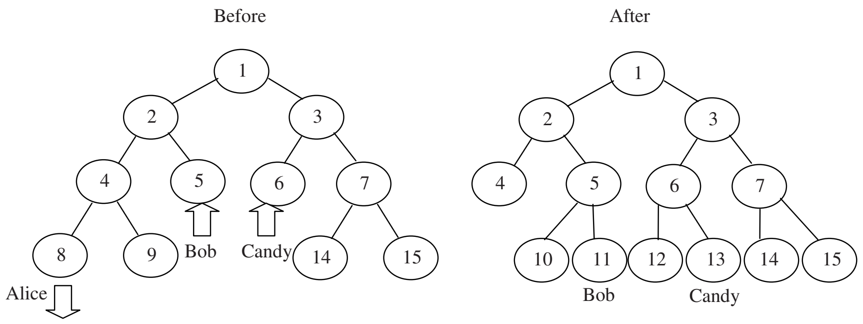


Fig. 1. OFT Key Tree and Collusion Attacks

Let the height of a balanced key tree be h. Then approximately h new blinded keys must be broadcasted on each member addition or eviction (on the other hand, a unicast is used to send the joining member its blinded keys). In addition, h bits are broadcasted to notify members about the position of the joining or eviction. In contrast, the broadcast size of Logical Key Hierarchy (LKH) is $2h$ multiplied by the key size (plus the same h bits for the position of the addition or eviction). The reason that OFT can achieve a halving in broadcast size is that keys in an OFT key tree are functionally dependent, but keys in a LKH key tree are all independent. In OFT, an updated node key is propagated through the sibling of the node, whereas in LKH the key is propagated through the children of the node. The fact that a node has two children but only one sibling explains the difference in the broadcast size of LKH and OFT.

2.2 Examples of Collusion Attack on OFT

Horng observed that the functional dependency among keys in an OFT key tree subjects the OFT scheme to a special collusion attack [16]. Horng gave two conditions for such an attack to exist. Referring to Figure 1, the attack example given by Horng can be described as follows. Suppose Alice, associated with the node 8, is evicted at time t_1, and later Candy joins the group at time t_2 (ignore Bob's joining for the time being). By the OFT scheme, the node key of node 3 is not affected by the eviction of Alice, so Alice knows the blinded version of this key between t_1 and t_2. Moreover, the node key of node 2 is updated when Alice is evicted, and then remains the same even after Candy joins. Candy can thus see the blinded version of this key between t_1 and t_2. Knowing the blinded node key of both node 3 and node 2 between t_1 and t_2, Alice and Candy can collude to compute the group key during that time interval. The OFT scheme thus fails to provide forward security (Alice knows future group key) and backward security (Candy knows previous group key).

Intuitively, the above example is a result of the unchanging keys of the root's children. Horng thus stated two necessary conditions for such an attack to exist, that is the two colluding nodes evicted and joining at different side of the root and no key update happening between time t_1 and t_3 [16]. Later, Ku and Chen showed through two more attack examples that Horng's conditions are actually not necessary [11]. First, referring to Figure 1, if Alice is evicted at time t_1 and Bob joins later at time t_2, then they can collude to compute the node key of node 2 between t_1 and t_2 due to a similar reason. In addition, both Alice and Bob know the blinded node key of node 3 between t_1 and t_2, so they can compute the group key between the same time interval. Second, assume Alice is evicted at time t_1, and Bob and Candy join at time t_2 and t_3, respectively, with $t_1 < t_2 < t_3$. By similar arguments, Alice knows the blinded node key of node 3 between t_1 and t_3, and Candy knows the blinded node key of node 2 between t_2 and t_3. They can thus collude to compute the group key between t_2 and t_3. The two examples show that Horng's two conditions are actually not necessary.

Ku and Chen also provided a solution to prevent the collusion attack on OFT [11]. Intuitively, an evicted member brings out knowledge about some keys that will remain the same for a certain time interval after the eviction. Ku and Chen modify the OFT scheme to change all the keys known by an evicted member upon the eviction. For example, when Alice is evicted in Figure 1, the node key of node 5 and node 3 will be updated (in addition to that of node 4, node 2, and node 1, as required by the original OFT scheme). With this solution, no evicted member can bring out any knowledge about future keys, so a collusion with future joining members is prevented. However, the solution updates the node key of all the h siblings on the path of an evicted node (node 5 and node 3 in above example). Each such update requires the broadcast of h keys (for example, to update the node key of node 3, we must update one of the leaf nodes in the subtree rooted as node 3). The broadcast size is thus h^2 multiplied by the key size plus h bits. Because such a broadcast is required for every eviction, the modified OFT is less efficient than LKH (which broadcast $2h$ keys on an eviction) in most cases, unless member eviction is rare.

3 Generic Collusion Attack on OFT

Section 3.1 first studies a special case, that is an evicted node colludes with another node who joins later. This turns out to be the only interesting case. Section 3.2 then discusses the general case where multiple evicted nodes and joining nodes may collude.

3.1 Collusion Between an Evicted Node and a Joining Node

We first consider the collusion attack between a node A evicted at time t_A and a node C joining the group at time t_C ($t_A < t_C$). Without loss of generality, we assume A is the leftmost node in the key tree, as shown in Figure 2 (notice that this figure actually combines two different key trees at t_A and t_C, which will be justified later in this section). We also need following notations. For any node v, we use $x_{v[t_1,t_2]}$ and $y_{v[t_1,t_2]}$ for its node key and blinded node key between time t_1 and t_2, respectively. We shall also interchangeably refer to a node and the member who is associated with that node. I is the node where the path of A to the root and that of C merges. Let L, R, I', I'' be the left child, right child, parent of I, and parent of I', and let R' and R'' be the right child of I' and I'', respectively. Let B, D, E, and F denote the subtree with the root L, R, $right(I')$, and $right(I'')$, respectively. Let t_{DMIN}, t_{EMIN}, and t_{FMIN} be the time of the first key update after t_A that happens in D, E, and F, respectively. Let t_{BMAX},

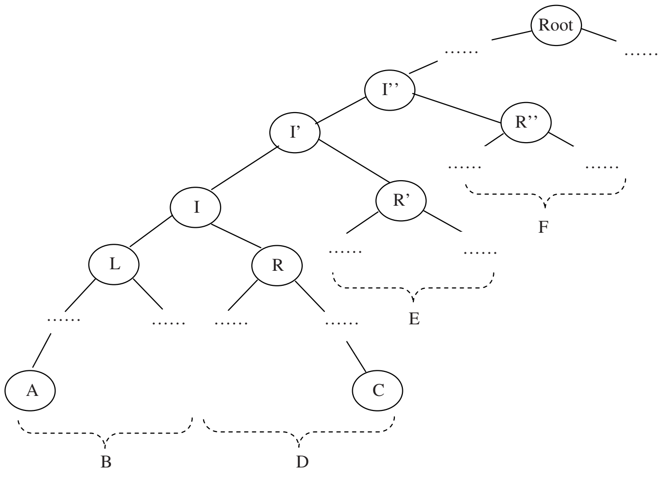


Fig. 2. A Generic Collusion Attack on OFT

t_{EMAX}, t_{FMAX} be the time of the last key update before t_C that happens in B, E, and F, respectively. We then have the following result.

Proposition 1. *Referring to Figure 2, the only node keys that can be computed by A and C when colluding are:*

- x_I *in the time interval* $[t_{BMAX}, t_{DMIN}]$,
- $x_{I'}$ *in* $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMAX}, t_C])$,
- $x_{I''}$ *in* $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMAX}, t_C]) \cap ([t_A, t_{FMIN}] \cup [t_{FMAX}, t_C])$,

and so on, up to the root. Notice that these intervals may be empty.

Proof: When the node A is evicted, it knows the blinded node key of each sibling on its path to the root before the time t_A. This includes $y_{R[-,t_A]}$ and $y_{R'[-,t_A]}$ (recall that the dash means the time when each key is last updated before t_A). By the OFT scheme, the node key of R will not change until a new node joins a node in D (that is, the subtree with the root R) or a node in D leaves, and similarly the node key of R' will not change until a key is updated in E. That is, $y_{R[-,t_A]} = y_{R[-,t_{DMIN}]}$ and $y_{R'[-,t_A]} = y_{R'[-,t_{EMIN}]}$. The node A thus knows these values even after it is evicted. On the other hand, when node C joins, it is given the blinded node key of the siblings on its path to the root. The node C then knows the values $y_{L[t_C,-]}$ and $y_{R'[t_C,-]}$ (recall that the dash here means the time of the next update of these keys after t_C). By the OFT scheme, the node key of L and R' will not be updated when C joins so they have remained the same since the last key update in B and E, respectively. Then we have $y_{L[t_C,-]} = y_{L[t_{BMAX},-]}$ and $y_{R'[t_C,-]} = y_{R'[t_{EMAX},-]}$, which are both known by C.

When A and C colludes, what can be computed depends on the relationship between the timestamps. As shown in Figure 3, A and C can first compute the subgroup key $x_{I[t_{BMAX},t_{DMIN}]} = f(y_{R[-,t_{DMIN}]}, y_{L[t_{BMAX},-]})$. We notice that this statement assumes $t_{BMAX} < t_{DMIN}$. Under this assumption, nodes A and C can compute $y_{I[t_{BMAX},t_{DMIN}]} = g(x_{I[t_{BMAX},t_{DMIN}]})$. This will enable them to further compute another subgroup key I' in two different time intervals. Let $t_{DEMIN} = MIN(t_{DMIN}, t_{EMIN})$ and $t_{BEMAX} = MAX(t_{BMAX}, t_{EMAX})$. Then $x_{I'[t_{BMAX},t_{DEMIN})]}$ can be computed by A and C as $f(y_{I[t_{BMAX},t_{DMIN}]}, y_{R'[-,t_{EMIN}]})$ and $x_{I'[t_{BEMAX},t_{DMIN}]}$ can be computed as $f(y_{I[t_{BMAX},t_{DMIN}]}, y_{R'[t_{EMAX},-]})$. In another word, they can compute the node key of I' in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMAX}, t_C])$. Clearly, this result can be easily extended to the parent of I' and so on, up to the root.

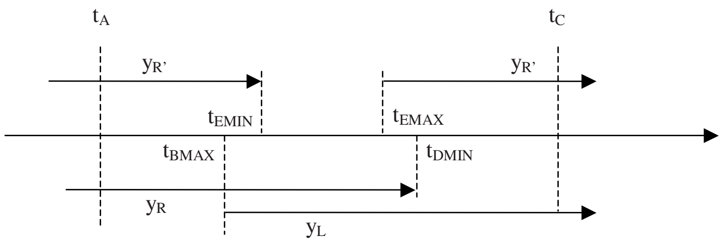


Fig. 3. The Timeline of Collusion Attacks

On the other hand, the above result also depicts all that A and C can compute by colluding. By the OFT scheme, when A is evicted all the node keys along its path to the root are updated, so A no longer knows them. Similarly, C cannot learn any node key on its path to the root prior to its joining. Besides the blinded keys of nodes R, R', and R'' (and all the sibling nodes on the path from I to the root), A may also know the blinded node key of sibling nodes in the subtree B for a time interval after t_A, and similarly C may know about nodes in the subtree D for a time interval before t_C. However, such knowledge does not help them in computing any keys. By the OFT scheme, a node key can only be computed from the blinded key of its two children, but we can never pick a node from the set $B - \{L\}$ and another from $D - \{R\}$ such that they are the children of the same node. $\square$

One subtlety lies in the dynamics of the key tree. The key tree from which A is evicted is different from the one that C joins. Although we show A and C in the same key tree in Figure 2 for simplicity purpose, the tree structure may have been changed after A leaves and before C joins. However, the key facts that our results depend on will not be affected by such changes. First, A knows $y_{R[-,t_{DMIN}]}$ and $y_{R'[-,t_{EMIN}]}$ regardless of any changes that may happen to the subtree with root L, and the definition of t_{DMIN} and t_{EMIN} excludes any change in the subtree with root R and R' to happen before t_{DMIN} and t_{EMIN}, respectively. It is worth noting that the whole subtree with root L may disappear due to evictions, and consequently the node R will replace its parent I (and the node R will be replaced by $right(R)$) by the OFT scheme. In this case, it seems that A will no longer know y_R even when no key update happens in the set D, invalidating the result that A knows $y_{R[-,t_{DMIN}]}$. However, this is not true. When the node R replaces I, the OFT scheme also requires it to be assigned a new node key, which means at least one of the leaf nodes in the set D must change its node key. That is, a key update does happen in D in this operation, and our result still holds. Similarly, C knows the value $y_{L[t_{BMAX},-]}$ regardless of any change in the key tree after the last key update in the set B.

3.2 The General Case

We first consider other cases of collusion between pairs of evicted and joining nodes and show that the above eviction-joining scenario turns out to be the only interesting case, as explained by Proposition 2. We then discuss the collusion among more than two nodes, and we show that it is sufficient to only consider collusion between pairs of nodes, which is stated in Proposition 3.

Proposition 2. *A pair of colluding nodes A and C cannot compute any node key which they are not supposed to know by the OFT scheme, if*

- *A is evicted after C joins.*
- *A and C both join.*
- *A and C are both evicted.*

Proof: First, we consider the joining-eviction case. In Figure 2, suppose C first joins the group and later A is evicted. If A and C collude, then they trivially know all node keys in the intersection of their paths to the root (for example, node I and I') and the siblings (for example, node R') before C joins and after A is evicted, because A is in the group before C joins and C stays in the group after A is evicted. In addition, although A knows the blinded node key of some siblings in the subtree B and C knows the blinded node key of some siblings in the subtree D, these keys cannot be combined to compute any node key since no two nodes share a parent. In summary, two nodes colluding in the joining-eviction case cannot compute any node key besides what they already know.

Next consider the eviction-eviction case. Suppose in Figure 2 A is first evicted at time t_A and later C is evicted at time t_C. Because C stays in the group longer than A does, their knowledge about the shared keys in the intersection of their paths (such as I and I') and the siblings (such as R') is the same as C's knowledge. That is, colluding with A does not help C with respect to these keys. Similar to the above cases, A's knowledge about nodes in the subtree B cannot be combined with C's knowledge about nodes in D to compute any node key. The only exception is their knowledge about L and R, which can potentially be combined to compute I (and consequently I' and so on). However, the OFT scheme updates the node key of R when C is evicted, so A can at best know $y_{R[-,t_A]} = y_{R[t_A,t_C]}$ (if no other key update happens between t_A and t_C), which is useless to C. In summary, two evicted nodes colluding cannot compute any node key in addition to what is already known by the later-evicted node. The joining-joining case is similar to the eviction-eviction case and is omitted. □

Proposition 3. *An arbitrary collection of evicted nodes and joining nodes can collude to compute some node key not already known, if and only if the same node key can be computed by a pair of nodes in the collection.*

Proof: The if part is trivial, and the only if part can be justified as follows. To compute $x_{v[t_1,t_2]}$, the colluding nodes must know both $y_{left(v)}$ and $y_{right(v)}$ for some time intervals that are supersets of $[t_1, t_2]$. Suppose $y_{left(v)}$ is known by m nodes in time period $[t_{ai}, t_{bi}](1 \leq i \leq m)$, and $y_{right(v)}$ is known in $[t_{cj}, t_{dj}](1 \leq j \leq n)$. Because $(\bigcup_{i=1}^{m}[t_{ai}, t_{bi}]) \cap (\bigcup_{j=1}^{n}[t_{cj}, t_{dj}])$ is a superset of the non-empty time interval $[t_1, t_2]$, it cannot be empty, either. Consequently, there must exist a pair of i and j such that $[t_{ai}, t_{bi}] \cap [t_{cj}, t_{dj}] \neq \phi$. The pair of nodes that has such knowledge (no single node can possess this knowledge because we assume $x_{v[t_1,t_2]}$ is not already known by the colluding nodes) can thus collude to compute x_v during the time interval $[t_{ai}, t_{bi}] \cap [t_{cj}, t_{dj}]$. $\square$

We now show that the attack examples given by Ku et al., as described in Section 2.2, are special cases of our generic attack. Referring to Figure 1, the first example says that Alice evicted at t_1 colludes with Bob joining at t_2, and Candy joins at t_3 ($t_1 < t_2 < t_3$). This corresponds to the case where $A = 8, C = 5, I = 2, I' = 1$ (referring to Figure 2), and Candy joins at t_3 in the set E. We thus have $t_{BMAX} = t_1$, $t_{DMIN} = t_2$, and $t_{EMIN} = t_{EMAX} = t_3$. It then follows that Alice and Bob can collude to compute $x_{2[t_1,t_2]}$ and $x_{1[t_1,t_2]}$ (notice that $[t_1, t_2] \cap ([t_1, t_3] \cup [t_3, t_2]) = [t_1, t_2]$). The second example says that Alice evicted at t_1 colludes with Candy joining at t_3, with Bob joining in between at t_2. This corresponds to the case where $A = 8$, $C = 6$, $I = 1$ (I' does not exist), and Bob joins in the set B. We thus have $t_{BMAX} = t_2$ and $t_{DMIN} = t_3$, and consequently Alice colluding with Candy can learn $x_{1[t_2,t_3]}$.

4 A Solution for Preventing Collusion Attacks

The previous section shows that a joining node may collude with previous evicted nodes to compute node keys in certain time intervals, which none of them is supposed to know. However, these results also show that such a collusion is not always possible, and whether it is possible depends on the temporal relationship among joining and evicted nodes. As discussed in Section 2.2, Ku and Chen's solution prevents any evicted node from bringing out knowledge about future node keys. Although it suffices to prevent any collusion attack, this conservative approach has a quadratic broadcast size (in the height of the key tree) on every member eviction and thus is less efficient than the LKH scheme in most cases.

One apparent way to reduce the broadcast size is to update additional keys only when a collusion attack is indeed possible. Unfortunately, this cannot be achieved with Ku and Chen's approach of updating the siblings along the path of an evicted node, because at the time a node is evicted, we do not yet know with whom it may collude in the future. On the other hand, our results in Section 3 make it possible to check whether a joining node can collude with any previously evicted node. If a collusion is possible, we can update a minimum number of additional keys to prevent the joining node from combining its knowledge with the evicted node for that specific collusion. This approach minimizes the communication cost for each joining operation (the eviction operation has no additional communication cost) because a key is updated only when necessary.

We first describe a *stateful* method that explicitly records all the knowledge of evicted nodes. This straightforward method simply applies the results in the previous section to check for possible collusions. However, because the method needs to keep information about all evicted nodes, the storage requirement is proportional to the number of all evicted nodes, which is not acceptable in most applications. Later in this section, we modify this method such that its storage requirement becomes proportional to the size of the key tree. Both methods will eliminate collusion attacks while minimizing the broadcast size.

A Stateful Method. For the stateful method, the key manager tracks all evicted nodes and checks whether a joining node can collude with any previously evicted node. If a collusion is possible, additional key updates are performed to remove the joining node's knowledge about past node keys such that the collusion becomes impossible. The key manager needs to record two kinds of knowledge. First, the knowledge about *future* node keys that each evicted node brings out of the group. Second, the knowledge about *past* node keys that a joining member is given when it joins. For this purpose, the key manager stores a modified key tree as follows. Each node in the OFT key tree is now associated with a pair $< t_u, L >$, where t_u is a timestamp and L is a collection of timestamp pairs $< t_{x1}, t_{y1} >, < t_{x2}, t_{y2} >, \ldots, < t_{xn}, t_{yn} >$.

The OFT scheme will be modified such that the timestamp t_u records the time that the current node was last updated, and each pair $< t_{xi}, t_{yi} >$ records the time interval in which some evicted node knows the blinded node key of the current node. For example, Figure 4 shows such a modified OFT tree. Due to space limitation, only the three nodes I, L, and R have part of their timestamps shown in the figure. In the example, nodes A, B, and D were evicted at time t_A, t_B, and t_D, respectively. Another node C joined at time t_C. The node R was only updated once between t_A and t_B, and the update happened at time t_2. The node I was last updated at time t_1, which is before t_D (t_1 is equal to either t_2 or t_3). In the table attached to R, the timestamp t_2 records the time of its last update. The first pair $< t_A, t_2 >$ records the fact that node A knows the value $y_{R[t_A,t_2]}$. The second pair $< t_B, - >$ records that B knows the value $y_{R[t_B,-]}$ (that is, the value of y_R from t_B until now). In the table attached to I, t_1 is the last update time of I, and $< t_D, - >$ records that node D knows the value $y_{I[t_D,-]}$. In the table of L, the timestamp t_3 records the time of its last update.

The OFT scheme is modified as follows to update the timestamps and to stop collusions when they become possible. When a node v is evicted at time t, the key manager will also insert a pair $< t, - >$ into each sibling node along the path of v to the root. For example, in Figure 4 the pair $< t_B, - >$ is inserted to the table attached to node R when node B is evicted at time t_B because R is a sibling of L and L is on the path of B to the root. After a node v joins the group, the key manager will check if v can collude with any previously evicted node to compute any node key along the path of v to the root. In Figure 4, after the node C joins the group, for each node on the path of C to the root (excluding C), the key manager needs to do the following. Taken R as an example, the key manager will check whether the intersection $[t_3, -] \cap ([t_A, t_2] \cup [t_B, -])$ is empty. If the intersection is not empty, then the node key x_L will be updated, such that C can no longer collude with A and B to compute the node key x_I (in applications where only the root's key needs to be secure, the key manager can ignore the collusion of a subgroup key here).

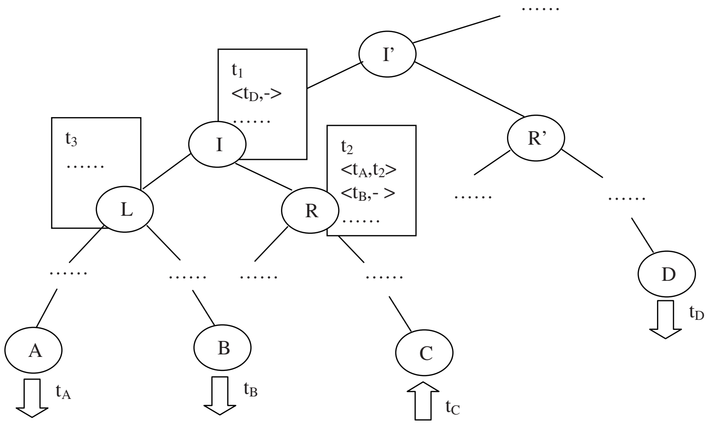


Fig. 4. A Stateful Method for Preventing Collusion Attack

Whenever the key manager updates the node key of a node v, regardless of the reason of this update, it will take following two additional actions. First, it will change the corresponding timestamp t_u associated with v to be the time of the current update. Second, it will scan all pairs of timestamps associated with v and change every dash in these pairs to the current time. The second action records the fact that the key update has invalidated the evicted node's knowledge about this node key. For example, in Figure 4 when the node A leaves, a pair $< t_A, - >$ is inserted into the table attached to R. Later at time t_2 the node key R is updated for some reason, and the dash in $< t_A, - >$ is replaced by the current time t_2, leading to the pair $< t_A, t_2 >$ shown in the figure. This reflects the fact that A no longer knows the new node key of R after time t_2.

An Improved Method With Linear Storage Requirement. The stateful method keeps all necessary information for checking possible collusions. This requires the key manager to build up an infinitely increasing list of evicted nodes, which is not acceptable in most applications. A closer look at the method reveals that it is not necessary to keep the whole list, if no collusion is to be tolerated. Actually for each node, it suffices to only keep at most one pair of timestamps (plus the timestamp for its last update). The storage requirement is thus linear in the size of the key tree, because for each node at most three timestamps need to be stored. Following two observations jointly lead to this result.

First, in Figure 4, if $t_A < t_B < t_2$, then after B is evicted the list of timestamps associated with R will be $< t_A, - >, < t_B, - >$. However, the pair $< t_B, - >$ is redundant and can be removed because $[t_B, -]$ is a subset of $[t_A, -]$. In another word, after the first pair of timestamps with a dash appears in the list, no other pair of timestamps needs to be stored until the next key update happens to the current node. Second, suppose in Figure 4 $t_A < t_2 < t_B$ is true, so none of $< t_A, - >$ and $< t_B, - >$ is redundant. We then have that $t_2 < t_B \le t_3$ ($t_B \le t_3$ holds, because t_3 is the time when x_L is last updated and the eviction of B will update x_L). Now that we know $t_2 < t_3$, the pair $< t_A, t_2 >$ can be safely removed, because the interval $[t_A, t_2]$ will never have a non-empty intersection with $[t_3, -]$.

Based on these two observations, we modify the eviction operation and key update operation of the stateful method as follows. First, when a node v is evicted at time t and a pair of timestamps $< t, - >$ is to be inserted into each sibling node along the path of v to the root, the key manager inserts this pair only if the pair of timestamps already associated with v does not contain a dash. Second, whenever the node key of a node v is updated, the key manager deletes any pair of timestamps associated with the sibling of v that does not contain a dash. For example, in Figure 4 if another node in the subtree with root R' is evicted after t_D but before I is updated, then nothing will be inserted into the table shown in the figure. If I is updated and the dash in $< t_D, - >$ is replaced, then this new pair will stay until the next key update in the subtree with root R'.

5 Empirical Results

This section compares the average communication overhead of our solution, the LKH scheme, the original OFT scheme, and Ku and Chen's modified OFT scheme. Among the four schemes, the original OFT scheme is vulnerable to collusion attacks, and it is included as a baseline to show the additional overhead for preventing collusion attacks. Both our scheme and the modified OFT scheme by Ku and Chen can prevent collusion attacks. The keys in an LKH key tree are independently chosen, so LKH is not vulnerable to the collusion attack discussed in previous sections. We expect our scheme to outperform Ku and Chen's scheme in most cases, because the latter has a quadratic broadcast size for every eviction operation. We also expect our scheme to have a smaller average-case broadcast size than the LKH scheme in some cases.

The communication overhead is measured as the total number of keys broadcasted during a random sequence of joining and eviction operations. We do not consider the unicast of keys to a new member. As discussed in previous sections, collusions depend critically on the order of joining and eviction operations (on the other hand, the specific time duration between these operations is not significant). Starting from an initial key tree of G nodes, a sequence of totally N operations are performed using each of the four schemes. The probability that each operation is the eviction of a member is P (and that of a joining operation $1 - P$). As required by the OFT scheme, the position for each joining operation is chosen to be a leaf node closest to the root. For each eviction operation, the node to be evicted is randomly chosen among all existing leaf nodes.

The left hand side of Figure 5 shows the total broadcast size (the number of keys to be broadcasted) versus the size of the key tree. Totally 20000 operations are performed (about half of them are evictions). As expected, the communication overhead of our solution is much less than that of Ku and Chen's scheme (their scheme broadcasts about five times more keys). Compared to the original OFT scheme, our scheme only has small additional overhead until the key tree size increases over 20000 nodes. The broadcast size of our scheme is also smaller than LKH when the key tree size is smaller than 40000. Table 1 shows a more detailed comparison between the two schemes.

For larger key trees, our scheme is less efficient than LKH. As shown in the second row of Table 1, the broadcast size of our scheme is about double the size of LKH when the key tree has 80000 or more nodes. This can be explained by the fact that more collusions are possible in a larger tree, as shown in Table 1, and the larger height of the tree also increases the number of keys to be broadcasted upon each key update. Ku and Chen's scheme also has a similar trend as ours, which confirms that to prevent collusion attacks, both modified OFT schemes are less scalable than LKH. However, because our scheme only perform additional key updates when necessary, the broadcast size for each operation is already minimal. This indicates an inherent disadvantage of using functionally dependent keys in the face of collusion attacks. For large groups where perfect forward and backward security is important, the LKH scheme will be a better choice.

Table 1. Comparing Our solution to LKH

Key Tree Size	2000	5000	8000	10000	20000	40000	60000	80000	100000
Our Solution/LKH	0.59	0.60	0.62	0.70	0.84	1.08	1.61	2.19	2.24
No. of Collusions	242	1063	2113	5154	10385	18991	38417	54720	61799
Height of The Tree	10	12	12	13	14	15	15	16	16

The right hand side of Figure 5 shows the total broadcast size versus the number of operations, with about half of the operations being evictions, on a key tree with 10000 keys. Because collusion attacks depend on the order of operations but not on the specific time durations, we can also regard the number of operations as the intensity of operations, and Figure 5 thus also shows the broadcast size versus the degree of group dynamics. The broadcast size of all four schemes increases with the number (intensity) of operations. The original OFT scheme, the LKH scheme, and our modified OFT scheme all scale in roughly the same manner, whereas Ku and Chen's scheme is less scalable. The column chart inside Figure 5 shows the total number of collusions. Interestingly, while the number of collusions remains roughly the same when the number of operations goes over 6000, Ku and Chen's scheme still shows a significant increase in the broadcast size, because their scheme requires additional key updates for every eviction operation even when such operation do not cause collusion (in contrast, our scheme scales in the same way as the original OFT).

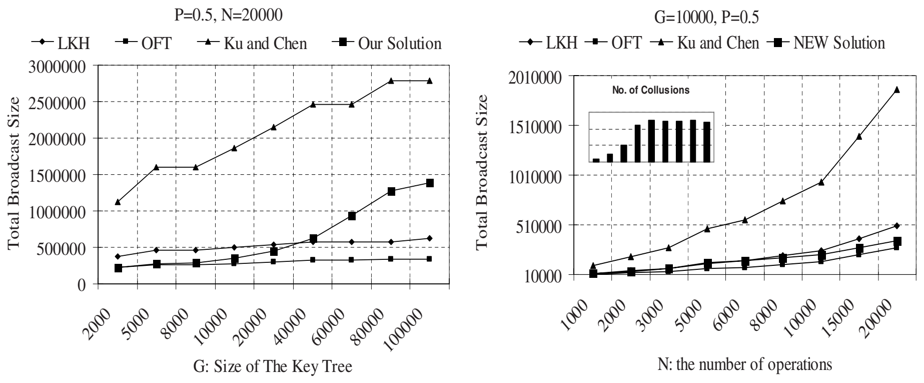


Fig. 5. The Broadcast Size Versus Key Tree Size and The Number of Operations

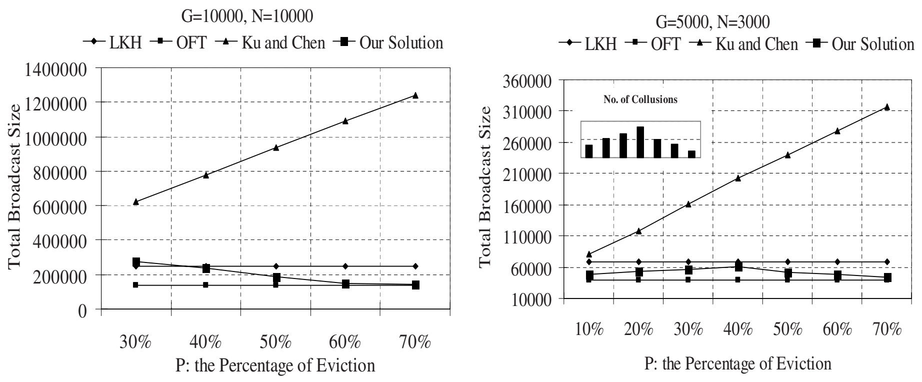


Fig. 6. The Broadcast Size Versus the Ratio of Eviction

Figure 6 shows the total broadcast size versus the ratio of evictions among all operations. The two experiments differ in the key tree size and in the total number of performed operations. In both experiments, the original OFT scheme and the LKH scheme have a constant broadcast size because in both schemes the joining and eviction require the same amount of keys to be broadcasted. The broadcast size of Ku and Chen's scheme increases linearly in the ratio of eviction, because their scheme requires additional key updates and hence additional broadcasted bits on every eviction operation but not on the joining operation. Our scheme shows an interesting pattern. The broadcast size first increases with the eviction ratio and then decreases after the ratio reaches about 40%. This is explained by the column chart inside the figure, which shows the total number of collusions. Because a collusion requires both joining nodes and evicted nodes, the total number of collusions reaches a maximal value when about half of the operations are evictions. The maximal broadcast size shifts a little to the left (40% instead of 50%) because our scheme requires additional key updates for joining nodes, but not for evicted nodes. Each joining node thus contributes to the overall broadcast size slightly more than an evicted node does.

6 Conclusion

We studied collusion attacks on the one-way function tree (OFT) scheme. The OFT scheme achieves a halving in broadcast size in comparison to the LKH scheme. However, OFT's approach of using functionally dependent keys in the key tree also renders the scheme vulnerable to collusion attacks between evicted members and joining members. We have generalized previous observations made by Horng and Ku *et al.* [16] into a generic collusion attack on OFT. This generalization also gave a necessary and sufficient condition for the collusion attack on OFT. Based on this condition, we have proposed a modified OFT scheme. The scheme is immune to the collusion among an arbitrary number of joining and evicted members, and it minimizes the broadcast size for each operation. The scheme has a storage requirement proportional to the size of the key tree. Experiments show that our scheme has smaller communication overhead than the LKH scheme for small to medium groups. For large groups, the increasing number of collusions renders the OFT scheme a less efficient choice than LKH. As future work, we will investigate cases where the compromise of some sub-group keys is an acceptable risk. Such a relaxed security requirement will likely lead to reduced communication overhead.

Acknowledgements. The authors are grateful to the anonymous reviewers for their valuable comments.

References

1. D. McGrew, A. David, T. Alan, and A. Sherman, Key establishment in large dynamic groups using one-way function trees, TIS Report 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, 1998.
2. A.T. Sherman, D.A. McGrew, Key establishment in large dynamic groups using one-way function trees, *IEEE Transactions on Software Engineering*, Volume 29, Issue 5, Pages 444-458, May 2003.
3. D.M. Balenson, D.A. McGrew, and A.T. Sherman, Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization, InternetDraft(work in progress), Internet Engineering Task Force, draft-irtf-smug-groupkeymgmt-oft-00.txt., August 2000.
4. D.M. Balenson, D.A. McGrew, and A.T. Sherman, Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization, InternetDraft(work in progress), Internet Engineering Task Force, draft-balenson-groupkeymgmt-oft-00.txt., February 1999.
5. K. Peter, A survey of multicast security issues and architectures, In *Proceedings of 21st National Information Systems Security Conference*, Pages 408-420, October 1998, Arlington, VA.
6. D. Wallner, E. Harder, R. Agee, Key Management for Multicast: Issues and Architectures, IETF, Request for Comments (RFC) 2627, June 1999.
7. M.J. Moyer, J.R. Rao, P. Rohatgi, A survey of security issues in multicast communications, *IEEE Network*, Volume 13, Issue 6, Pages 12-23, 1999.
8. R. Canetti, J. Garey, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, Multicast security: A taxonomy and efficient constructions, In *Proceedings of IEEE InfoComm'99*, vol. 2, Pages 708-716, Mar. 1999.
9. H. Harney, C. Muckenhirn, and T. Rivers, Group key management protocol architecture, IETF, RFC2093, 1997.
10. H. Khurana, R. Bonilla, A. Slagell, R. Afandi, H.S. Hahm, and J. Basney, Scalable Group Key Management with Partially Trusted Controllers, In *Proceedings of International Conference on Networking*, 2005.
11. W.C. Ku, S.M. Chen, An improved key management scheme for large dynamic groups using one-way function trees, In *Proceedings of 2003 International Conference on Parallel Processing Workshops*, Pages 391-396, October 2003.
12. D. Matthew, J. Moyer, J.R. Rao, and P. Rohatgi, A Survey of Security Issues in Multicast Communications, *IEEE Network Magazine*, November/December 1999.
13. T. Hardjono, and L.R. Dondeti, 2003. Multicast and Group Security. Artech House, Boston, London, ISBN 1-58053-342-6.
14. R. Canetti and B. Pinkas, A taxonomy of multicast security issues, dracanetti-secure-multicast-taxonomy-00.txt, IETF Internet Draft (work in progress), 1998.

15. H. Harney and E. Harder, Logical Key Hierarchy Protocol, Internet Draft (work in progress), draft-harney-sparta-lkhp-sec-00.txt, Internet Engineering Task Force, Mar. 1999.
16. G. Horng, Cryptanalysis of a Key Management Scheme for Secure Multicast Communications, *IEICE Trans. Commun.*, vol. E85-B, no. 5, Pages 1050-1051, 2002.
17. A.T. Sherman, A proof of security for the LKH and OFC centralized group keying algorithms, NAI Labs Technical Report No. 02-043D, NAI Labs at Network Associates, Inc., 2002.
18. J. Fan, P. Judge, M. Ammar, HySOR: Group Key Management with Collusion-Scalability Tradeoffs Using a Hybrid Structuring of Receivers, In *Proceedings of the IEEE International Conference on Computer Communications Networks*, Miami, 2002.
19. J.C. Lin, F. Lai, H.C. Lee, Efficient Group Key Management Protocol with One-Way Key Derivation, In *Proceedings of The 2005 IEEE Conference on Local Computer Networks*, Pages 336-343, 2005.
20. Y. Wang, J. Li, L. Tie, H. Zhu, An efficient method of group rekeying for multicast communication, In *Proceedings of the 6th IEEE Circuits and Systems Symposium*, Pages 273-276, June 2004.
21. S. Xu, Z. Yang, Y. Tan, W. Liu, S. Sesay, An efficient batch rekeying scheme based on one-way function tree, In *Proceedings of The IEEE International Symposium on Communications and Information Technology*, Pages 490- 493, 2005.
22. C.K. Wong, M. Gouda, and S.S. Lam, Secure group communications using key graphs, *ACM Computer Communication Review*, vol. 28, no. 4, Pages 68-79, September 1998.

Bayesian Methods for Practical Traitor Tracing

Philip Zigoris[1] and Hongxia Jin[2]

[1] University of California, Santa Cruz
Santa Cruz, CA 95060, USA
[2] IBM Research, Almaden
San Jose, CA 95120, USA

Abstract. The practical success of broadcast encryption hinges on the ability to (1) revoke the access of compromised keys and (2) determine which keys have been compromised. In this work we focus on the latter, the so-called *traitor tracing* problem. The first method utilizes a Bayesian hierarchical model to replace a crucial step in a well known tracing algorithm. Previously, this step relied on worst case bounds, which often overestimate the number of tests needed to diagnose compromised keys. The second is an adaptive tracing algorithm that selects forensic tests according to the *information gain* criteria. The results of the tests refine an explicit model of our beliefs that certain keys are compromised. In choosing tests based on this criteria, we significantly reduce the number of tests, as compared to the state-of-the-art techniques, required to identify compromised keys.

1 Introduction

Digital piracy is a burning problem for the entertainment industry. After all, digital data can be perfectly and quickly copied. If consumers may freely copy entertainment content and offer that content on the Internet, the market for entertainment content would evaporate. To solve this problem, several frameworks for broadcast encryption [3] have been devised and are in wide use in the market, like CPRM, CPPM, DTCP and AACS [1] [2]. All of these methods are based on encryption of the content: the device manufacturer is given cryptographic keys to decrypt the content, and in return is obligated by the license to follow a set of rules limiting the physical copies that can be made from a single piece of content. Maintaining the secrecy of the cryptographic keys is essential for maintaining the integrity of distribution. scheme. In the event that some keys are compromised and made public, a content protection scheme should be able to revoke their ability to access future content. To combat this eventuality, each device is assigned a set of device keys that can be indirectly used to decrypt the content. The device keys, owned by compliant devices, repeatedly encrypt the content encrypting key (called the media key) in a structure called a media key block (MKB). Each device uses a device key to decrypt the media key block to obtain a valid media key to decrypt the content. The revoked devices, on the other hand, cannot decrypt the MKB and obtain a valid media key to decrypt the content.

To circumvent the content protection scheme, an adversary may break a device, extract the device keys, and build a circumvention device (also known as a clone device or a clone box) comprising the extracted device keys. To identify which original devices (called traitors) have donated their keys to the circumvention device, traitor-tracing technologies [4] are used. Traitor-tracing technology uses carefully crafted media key blocks called *forensic media key blocks*. When a circumvention device is found, the license agency feeds a series of forensic media key blocks to the device. By observing the clone's responses the licensing agency can determine precisely which device keys the circumvention device comprises. The licensing agency can then produce new media key blocks that revoke those compromised device keys such that newly released content cannot be played by the circumvention device.

Since many devices can donate their keys the clone box, it is important that a traitor tracing method is resistant to collusion; this represents the real challenge in traitor tracing. The state-of-art and practice broadcast encryption and traitor-tracing technology comprises a subset-difference scheme, described in [11]. Tracing under this scheme has proven to be theoretically efficient: To determine the compromised keys, the traitor tracing method requires on the order of $T^3 \log(T)$ forensic media key blocks to defeat a circumvention device comprising T sets of compromised device keys. However, this method has not proven to be a completely practical solution. Measures can be taken by the circumvention device to slow down the testing process. For example, each testing iteration may take a minute or more. A circumvention device comprising 100 compromised keys (i.e., $T = 100$) may require over 15 years to determine the device keys the circumvention device has compromised. In effect, such a circumvention device had defeated the content protection system.

The main contribution of this paper is to present two much more efficient, thus, practical traitor tracing schemes. Efficiency is measured by the number of forensic media key blocks required to detect traitors from the circumvention device. The first method, *BayesNNLTrace*, maintains the strict black-box assumptions of previous methods, but uses a Bayesian hierarchical model for diagnosis instead of relying on worst case bounds. The second method, *IGTrace*, assumes the tracing algorithm has some prior knowledge about the behavior of the clone box. It leverages this knowledge as well as the clone box's response to forensic MKBs to infer an explicit model of which keys have been compromised. Not only does this allow for accurate diagnosis, we can also quantify how informative a potential forensic MKB is, which we then use to guide the tracing process. Overall, adaptively choosing the best forensic MKB at each step and continuously updating our beliefs about which keys are compromised allow us to substantially reduce the number of forensic MKBs needed to identify traitors.

In rest of the paper, we will show the state of art subset tracing in more details in Section 2. This section provides the foundation for all subsequent sections. We will then show the first Bayesian approach to improve the subset tracing in Section 3 when no known strategy is known/assumed. We then show another Bayesian approach for subset tracing in Section 5 with a known strategy. Our experimental results in Section 6 shows order of magnitude of improvement in the number of forensic MKBs needed to defeat the clone box.

2 NNL Subset Tracing

Naor, et. al [11] present a broadcast encryption framework using subset covers. Let $\mathcal{D}$ be the set of devices and $\mathcal{K}$ be the set of device keys. Every device $d \in \mathcal{D}$ owns a subset of keys, denoted by $\mathcal{K}_d$. Similarly, associated with every key $k \in \mathcal{K}$ is a set of users $\mathcal{D}_k = \{d \in \mathcal{D} : k \in \mathcal{K}_d\}$.

Suppose we want to broadcast some media M, which, for all intent and purpose, is a binary string. We would like to encrypt M in such a way that a set of legitimate devices $L \subseteq \mathcal{D}$ is able to decrypt and view the media. The first step is to encrypt M with some key K, referred to as the *media key*. We will use the term *key* without a qualifier to refer to device keys. We then find a subset of device keys C such that all legitimate devices are *covered*. That is, C is chosen such that $\bigcap_{k \in C} \mathcal{D}_k = L$. Now, for every $k \in C$ we separately encrypt the media key, giving us $E_k(K)$. Ultimately, the following items are broadcast

- The encrypted media: $E_K(M)$
- The encrypted media key: $\langle E_{k_1}(K), E_{k_2}(K), \ldots, E_{k_{|C|}}(K)\rangle$
- An index of the device keys used to encrypt the media key

These items together are referred to as a *media key block (MKB)*. Every device $d \in L$ will own a key used in the MKB and every device $r \in \mathcal{D}/L$, referred to as a *traitor*, will own none. Hence, it cannot recover the content.

Naor, et.al give a black box tracing algorithm where the only means to diagnosis traitors is to submit tests, sometimes referred to as *forensic MKBs*, to the clone box and observe its response. Their tracing algorithm relies on two things:

1. For every key $k \in \mathcal{K}$ such that $|\mathcal{D}_k| > 1$, there exists keys k_1 and k_2 such that $\mathcal{D}_{k_1} \cup \mathcal{D}_{k_2} = \mathcal{D}_k$ and $D_{k_1} \cap D_{k_2} = \emptyset$. This is referred to as the *bifurcation property*. By this property, we can replace k with k_1 and k_2 and still cover the same set of devices.
2. We have access to a method that, given a set of keys F, finds at least one key in F owned by the clone box. This is referred to as *subset tracing*; the name is due to the fact that each key is associated with a subset.

The algorithm maintains a covering of all legitimate devices $\mathcal{F}$, referred to as the *frontier*. The algorithm proceeds by repeatedly identifying a compromised key $k \in \mathcal{F}$, removing it, and adding to $\mathcal{F}$ k_1 and k_2 satisfying the bifurcation property. If $|\mathcal{D}_k| = 1$ then the single device in $\mathcal{D}_k$ is a traitor. This process is reiterated until the clone box is unable to play the MKB associated with the frontier.

We can formalize subset tracing as follows: the *frontier* $\mathcal{F}$ is a set of keys and the clone box owns a subset $\mathcal{C} \subseteq \mathcal{F}$ of these keys. Let $m = |\mathcal{F}|$. The set $\mathcal{C}$ is, of course, unknown to the tracing algorithm. The task is to determine, within

Algorithm 1.

```
1: **NNLTrace**($\mathcal{F}, a, b, p_{T_a}, p_{T_b}$)
2: **if** $a = b - 1$ **then**
3:    **return** $b$
4: **else**
5:    $c \leftarrow \lceil \frac{a+b}{2} \rceil$
6:    $T_c \leftarrow \{f_{c+1}, \ldots, f_{|F|}\}$
7:    **if** $|p_{T_c} - p_{T_a}| \geq |p_{T_c} - p_{T_b}|$ **then**
8:       **return** NNLTrace($\mathcal{F}, a, c, p_{T_a}, p_{T_c}$)
9:    **else**
10:      **return** NNLTrace($\mathcal{F}, c, b, p_{T_c}, p_{T_b}$)
```

some specified confidence ϵ, at least one key $k \in \mathcal{F}$ that is owned by the clone box. That is, $Pr(k \in \mathcal{C}) > 1 - \epsilon$.

The structure of a forensic MKB, or simply a *test*, is quite simple: we *disable* certain keys by encrypting a random bit string instead of the media key. The remaining keys are said to be *enabled*. This way, all devices that rely on disabled keys are unable to decrypt, or play, the content. There are two caveats. The first is that a device, including a clone box, can determine if a key it owns has been disabled. In this way, its possible for a clone box to know it is under test. The second is that since a clone box may contain both enabled and disabled keys, it can still decrypt content when one of its keys is disabled as well as stop playing when some of its keys are enabled. Its response in these situation constitutes an anti-tracing strategy. We will assume that if all of the keys in a clone box are enabled then it will always play. Of course, if none of the keys are enabled then it is impossible for it to play.

From here on, a test can be simply thought of as a set $T \subseteq \mathcal{F}$ of keys. The keys in T are enabled and the keys in $\mathcal{F}/T$ are disabled. Let p_T be the probability that the clone box plays test T. Since the clone box is assumed stateless, we can treat the outcome of the test as a Bernoulli random variable. If the probability of playing two tests, T and T', are not equal then it must be case that the clone box owns a device key in the exclusive-or of the two sets. Formally,

$$p_T \neq p_{T'} \longrightarrow \exists c \in C, c \in T \otimes T'$$

This motivates *NNLTrace* (Algorithm 1), a binary-search-like method for identifying a compromised key. We initially call the procedure with the arguments $(\mathcal{F}, 0, m, p_F, 0)$. The algorithm proceeds by progressively reducing the interval (a, b) in which we know there must be a compromised device-key. It does this by determining the probability that the midpoint test T_c plays and recursing on the the side with the larger difference in the endpoint probabilities.

Technically, we can recurse on either side as long as the probability of the endpoint tests are not equal. However, this allows for the possibility that the algorithm will become trapped at an interval where the end-point probabilities are arbitrarily close to one another. The challenge in this style of tracing algorithm is that we must estimate p_{T_c} by repeatedly submitting T_c to the clone box. The closer p_{T_a} and p_{T_b} are to one another, the more tests that are needed to confidently decide which is the larger subinterval. The authors show that $O(\log^2(m)\log(\frac{1}{\epsilon})/\Delta^2)$ tests are needed, where $|p_{T_a} - p_{T_b}| \geq \Delta$ (Claim 8). Since our algorithm recurses on the larger side, we have $\Delta > \frac{1}{m}$ and, therefore, the number of tests is upperbounded by $O(m^2 \log^2(m)\log(\frac{1}{\epsilon}))$. Since NNLTrace will recurse $\log(m)$ times, the overall number of tests required for NNLTrace to succeed with probability at least $1-\epsilon$ is upperbounded by $O(m^2 \log \frac{\log m}{\epsilon} \log^3 m)$.

In the *subset difference* key distribution scheme the final frontier size will be $O(2r)$ [11]. Since the size of the frontier can increase by at most one at every iteration of the top-level procedure, the subset tracing procedure is called $O(2r)$ times. Combining this with the bound from the previous paragraph, we have that for r traitors, the entire traitor tracing process will require $O(r^3 \log^3 r)$ tests. As discussed previously, this algorithm, though theoretically efficient, is easily defeated in practice.

3 Bayesian Approach to Subset Tracing

The NNLTrace routine requires the ability to determine, given three tests T_a, T_b, and T_c, if $|p_{T_a} - p_{T_c}| > |p_{T_b} - p_{T_c}|$ (Line 7, Algorithm 1. Henceforth we abbreviate P_{T_a} as P_a. We require that this decision is made with a probability of error less than some specified threshold ϵ. In this section we introduce a principled method for making this decision based on methods from Bayesian statistics. Instead of relying on worst-case bounds to determine the number of times each test is submitted, we model our uncertainty about p_a, p_b, and p_c using a Bayesian hierarchical model. This model is conditioned on the actual responses of the clone box and not a hypothetical worst case. Tests, therefore, are repeated only if we are unable to confidently determine if $|p_a - p_c| > |p_b - p_c|$.

To do this, we think of the probability of playing as a random variable, which we will denote by P_T in order to distinguish it from the actual probability the test plays p_T. For now, assume we have some fixed probability density function (pdf) specifying $Pr(P_T = p)$, abbreviated as $Pr(p)$. This is referred to as the *prior* distribution of P_T since it is not conditioned on any observations. After observing the outcome of a test, $T = t$, we would like to refine our belief about P_T; $t = 1$ implies the clone box was able to decrypt the content, $t = 0$ implies it wasn't. This is provided by Bayes Theorem:

$$Pr(P_T = p|T = t) = \frac{Pr(T = t|P_T = p)Pr(P_T = p)}{Pr(T = t)}$$

We refer to $Pr(P_t = p|T = t)$ as the *posterior distribution*. When there is no risk of confusion we will abbreviate the notation by only writing the observation, i.e. $Pr(P_T = p|T = t) = Pr(p|t)$.

We will model P_T with the Beta distribution, which is defined for p in the interval $[0, 1]$:

$$\text{Beta}(p; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} p^{\alpha-1}(1-p)^{\beta-1}$$

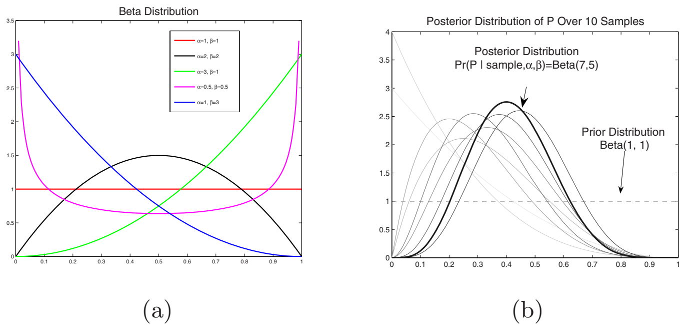


Fig. 1. (a) Beta distribution for different values of (α, β). (b) Example of posterior distribution of P. The gray curves show the posterior distribution of P after receiving each of 10 observations: $S = \{0, 0, 0, 1, 0, 1, 0, 1, 1, 0\}$.

where $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha)+\Gamma(\beta)}$ is the Beta *function* and $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt$ is the Gamma function. The $B(\alpha, \beta)$ term acts as a normalizing constant. The shape of the distribution is controlled by two parameters, α and β. Various settings of these parameters are illustrated in Figure 1(a). The expectation and variance of $P \sim \text{Beta}(\alpha, \beta)$ are, respectively, $\frac{\alpha}{\alpha+\beta}$ and $\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$.

The choice to use the Beta distribution comes from the fact that it is the *conjugate prior* of the Bernoulli and, implicitly, the Binomial distribution. That is, if the prior distribution for P_T is $\text{Beta}(\alpha, \beta)$ then the posterior distribution $Pr(P_T \mid T = t)$ is simply $\text{Beta}(\alpha + t, \beta + (1 - t))$. Given observations $\{T_i = t_i\}$ for $i = 1$ to n, where $\sum_i t_i = k$, the posterior distribution of P_T is $\text{Beta}(\alpha + k, \beta + (n - k))$. These convenient facts make updating the posterior distribution, illustrated in Figure 1(b), extremely simple. For a more general introduction to Bayesian statistics see, e.g., [6].

Ultimately we need to determine if $|p_a - p_c| > |p_b - p_c|$. Let $\mathcal{P} = \{(p, q, r) \in [0, 1]^3 : |p - r| > |q - r|\}$. Then,

$$\begin{aligned} D(\bar{t}) &= Pr(|P_a - P_c| > |P_b - P_c| \mid \bar{t}) \\ &= \int_{(p_a, p_b, p_c) \in \mathcal{P}} Pr(p_a \mid \bar{t}) Pr(p_b \mid \bar{t}) Pr(p_c \mid \bar{t}). \end{aligned} \tag{1}$$

It will also be useful to calculate the probability that the two gaps are equal, within some tolerance δ:

$$\begin{aligned} E_\delta(\bar{t}) &= Pr(\,|\,|P_a - P_c| > |P_b - P_c|\,| < \delta) \\ &= \int_{(p_a, p_b, p_c) \in \mathcal{Q}_\delta} Pr(p_a \mid \bar{t}) Pr(p_b \mid \bar{t}) Pr(p_c \mid \bar{t}). \end{aligned} \tag{2}$$

where $\mathcal{Q}_\delta = \{(p, q, r) \in [0, 1]^3 \;:\; |\,|p - r| > |q - r|\,| < \delta\}$. Finding a closed form solution for Equation 1 and 2 is unlikely. Instead, we use a simple *sample-based* approach to inference. The basic idea is that if we have a large enough sample from a distribution, we can approximate the above integrals. Sampling from a Beta distribution can be done quickly and we do not require approximate sampling methods such as Markov Chain Monte Carlo [10].

After every test, if $D(\bar{t}) > 1 - \epsilon$, $D(\bar{t}) < \epsilon$, or $E_\delta(\bar{t}) > 1 - \epsilon$ then we can recurse on the appropriate subinterval. If not, then we require further testing. In our approach, we repeat the test T with the highest variance in the posterior distribution of P_T.

4 Clone-Box Strategy

It has been conjectured [9] that the following strategy will force the above algorithms to submit the most tests: the clone box tries to decode the media key with one of its device keys chosen uniformly at random. If the key chosen is enabled then the clone box successfully plays the test; if it disabled then it does not. That is,

$$p_T = \frac{|T \cap C|}{|C|}$$

We refer to this as the *uniform choice* strategy. The motivation for this strategy is that it is advantageous for the clone box to minimize the difference between p_a and p_b. However, the subset tracing procedure always recurses on the larger subinterval and so at every step the gap is reduced by, at most, a factor of 2. This gives us the following lower bound on the gap on the last iteration:

$$|p_a - p_{a+1}| \geq \frac{1}{2^{\log_2 |C|}} = \frac{1}{|C|}$$

This lower bound is achieved by the uniform choice strategy.

If we assume that the clone box implements this strategy, then its reasonable to ask if we can leverage this information to reduce the number of tests. In the next section we introduce *IGTrace* which does exactly this. We will see in Section 6 that this method offers a dramatic reduction in the testing time.

5 Subset Tracing with a Known Strategy

In this section, we will be more specific about what defines a *clone-box strategy.* As before, $\mathcal{F}$ and $\mathcal{C}$ denote the frontier and the set of keys in the clone box, respectively. Let $m = |\mathcal{F}|$ and $n = |\mathcal{C}|$. Denote by T the random variable associated with the outcome of a test or a set of enabled keys comprising a test. The intended interpretation should be clear from the context. Let R be the set of possible responses to a test. In previous sections the set of responses R was limited to the set $\{0, 1\}$; it either plays or it does not. The approach we present here can accommodate a richer set of responses, a possibility we will study more closely in the experimental section. Associated with each key $k \in \mathcal{F}$ is a binary random variable F_k; $F_k = 1$ is meant to imply that $k \in \mathcal{C}$. We refer to these random variables, collectively, as F. In many contexts it is useful to treat F as the set of keys for which $F_k = 1$.

For our purposes, a strategy is a *conditional probability distribution* (CPD) specifying the probability the box plays a test conditioned on the fact that it contains a specific subset of keys. Formally, it specifies

$$Pr(T = t \mid F)$$

for all tests $T \subseteq \mathcal{F}$, all possible valuations of F, and all responses $t \in R$. The CPD is also subject to the same constraints as before

$$\text{if } T \cap F = \emptyset \text{ then } Pr(T = 0 \mid F) = 1$$

$$\text{if } T \cap F = F \text{ then } Pr(T = 0 \mid F) = 0$$

where the response 0 indicates the clone box did not play.

The question we would like to answer is: provided with a set of test-response pairs $T_1 = t_1, \ldots, T_N = t_N$, abbreviated as $\bar{T} = \bar{t}$, what is the posterior probability that the clone box contains a particular set of keys F. Applying Bayes rule we have:

$$\begin{aligned} Pr(F \mid \bar{T} = \bar{t}) &= \frac{Pr(\bar{T} = \bar{t} \mid F) Pr(F)}{Pr(\bar{T} = \bar{T})} \\ &= \frac{\prod_i Pr(T_i = t_i \mid F) Pr(F)}{Pr(\bar{T} = \bar{T})} \end{aligned} \tag{3}$$

where the second equality derives from the fact that the results of tests are independent conditioned on F. The goal of the subset tracing procedure is to find a key that is, with high probability, contained by the clone box. Formally, for some threshold ϵ, does there exist $k \in \mathcal{F}$ such that

$$Pr(F_k = 1 \mid \bar{T} = \bar{t}) = \sum_{F \subseteq \mathcal{F} : k \in F} Pr(F \mid \bar{T} = \bar{t}) > 1 - \epsilon$$

The term $Pr(F)$ specifies the prior probability that the clone box contains a set of keys F. Without any background knowledge we choose the uniform distribution. It is possible to embed domain specific knowledge in the prior to give the process a "head start". For example, if we knew that a particular subset of devices were more likely to have their keys compromised then we could increase the prior probability that the clone box contained one of those keys.

The denominator of Equation 3, $Pr(\bar{T} = \bar{t})$, is the marginal probability of observing the responses to the set of tests; it also acts as a normalizing constant and is defined as:

$$Pr(\bar{T} = \bar{t}) = \sum_{F \subseteq \mathcal{F}} Pr(\bar{T} = \bar{t} \mid F).$$

Note that the sum is taken over a set of size 2^m, making this a difficult quantity to calculate directly. Related work on adaptive diagnosis in distributed systems approximate this quantity using the *mini-bucket* algorithm [12]. For our application, there exists methods for efficiently calculating $Pr(\bar{T} = \bar{t})$ exactly, but they are outside the scope of this paper [13].

There is an important difference between this approach and the previous algorithms: here, we directly model our belief that the clone box contains a key. In (Bayes)NNLTrace, this belief was indirectly modeled by the probability that the probability of tests playing were different. In a sense, there was no negative evidence. Two tests playing with different probabilities only supported the existence of certain keys in the clone box. Now, the response of the box can indicate that certain keys are *not* contained in the box. This is one reason why this new approach offers such an improvement over (Bayes)NNLTrace. The other advantage to explicitly modeling the clone box's contents is that we can make a more informed choice about the next test to submit to the clone box. The details of this are described in Section 5.1.

The entire procedure is specified in Algorithm 2. Note that this procedure returns, in addition to the compromised key, the updated beliefs about the clone box. This information can be propagated by the top-level traitor tracing algorithm to subsequent calls to the subset tracing procedure. This, too, can have a significant impact on performance, although we do not document it here.

Algorithm 2. Strategy based subset tracing procedure

```
1: IGTrace($\mathcal{F}, Pr(F)$)
2: if response of clone-box to $T = \mathcal{F}$ is 0 then
3:    return [$\emptyset, Pr(F)$] //in this case, $\mathcal{C} = \emptyset$
4: loop
5:    for all $k \in \mathcal{F}$ do
6:       if $Pr(F_k = 1) > 1 - \epsilon$ then
7:          return [$k, Pr(F)$]
8:    select a test $T$(see Section 5.1)
9:    submit test to clone-box and get response $t$
10:   $Pr(F) \leftarrow Pr(F \mid T = t)$
```

5.1 Test Selection with Information Gain

We can imagine the testing process as a *decision tree* where every node specifies the next test to submit to the clone box. The response of the clone box dictates the subtree on which to recurse. Every leaf in the decision tree will have a key associated with it and reaching it in the recursion implies the associated key has been compromised. Ideally, we could minimized the expected number of tests needed by mapping out an entire testing strategy. However, this task was shown to be NP-Complete [7].

Instead of devising such a decision tree at once, we take a greedy approach to test selection. With a direct model of our beliefs about F, we can ask: how informative is a single test T? That is, how much would our uncertainty about

F decrease provided with a response by the clone-box to T? At every step we simply choose the test that maximizes the decrease in uncertainty.

The first step is to quantify our uncertainty about F. We will measure uncertainty as the *entropy*:

$$H(F \mid \bar{T} = \bar{t}) = -\sum_{F \subseteq \mathcal{F}} Pr(F \mid \bar{T} = \bar{t}) \log_2 Pr(F \mid \bar{T} = \bar{t})$$

Note that if we are certain of the contents of the clone-box then $H(F) = 0$. The entropy is maximized at m when the distribution of $Pr(F)$ is uniform. We measure the quality of a new test T as the *mutual information* between the T and F. In the machine learning literature it is often referred to as the *information gain* of T. It is defined as

$$I(F; T \mid \bar{T} = \bar{t}) = H(F \mid \bar{T} = \bar{t}) - \sum_{t \in R} Pr(T = t \mid \bar{T} = \bar{t}) H(F \mid \bar{T} = \bar{t}, T = t)$$

This is equal to the expected reduction in entropy by seeing the result of test T, taken with respect to the marginal probability of each response $t \in R$. For a more detailed explanation of entropy and mutual information, see, e.g., [10].

Now we can view test selection as solving the following optimization problem:

$$T^* = \text{argmax}_{T \subseteq \mathcal{F}} I(F; T \mid \bar{T} = \bar{t})$$

Of course, this obscures the fact that there are 2^m possible tests to consider. We know of no algorithm for efficiently solving this problem. Therefore, its necessary to approximate this by only considering a small subset of the possible tests. The simplest approach is to pick a set at random. Another approach is described in Algorithm 3. In this approach we maintain a set of tests $\mathcal{S}$ called the *retention set.* At every iteration we try adding a new key to each test in S, creating a new set of tests S'. We then update S to be the top s tests in S'. If after an iteration the retention set does not change, then we return the top test in S. The total number of tests evaluated by this procedure is $O(s^2m^2)$. The proof is as follows: Denote by T_i^j the jth most informative test at iteration i. There are at most sm iterations since $|T_i^j| \leq m$ and at every iteration, at least one of the tests must increase in size. Let $\mathcal{T}_i^j = \{T_i^j \cup \{k\} : k \notin T_i^j\}$ be the set of tests generated in lines 6- 8. Clearly, $|\mathcal{T}_i^j| < m$ so at every iteration we need to evaluate at most sm tests.

Note that there are many alternatives to using entropy to measure uncertainty. For instance, the *Gini impurity*(a generalization of variance) and *misclassification impurity* are popular measures for learning decision trees [5]. We chose entropy because it has proven effective for selecting tests in adaptive diagnosis tasks. Rish, et al. [12] apply a method similar to ours to the task of diagnosing faults in distributed system. A similar approach has been used to play a probabilistic version of the game of Mastermind [8]. Mastermind is a popular board game created in the 70's where two players engage in a query-response loop. In the original formulation, game play is completely deterministic. The goal is for

Algorithm 3. Test selection procedure

```
InfoGainTestSelect
$S \leftarrow \{T = \emptyset\}$
$S_{\text{old}} \leftarrow \emptyset$
while $S \neq S_{\text{old}}$ do
    $S_{\text{old}} \leftarrow S$
    for all $T \in S$ do
        for all $K \notin T$ do
            add $T' = T \cup \{K\}$ to $S$
    sort $S$ by descending information gain
    remove all but the top $s$ tests from $S$
return the first test in $S$
```

one player, the code-breaker, to identify the code put down by the code-maker. In many ways, it is similar to the task we are face with. We can imagine the clone box as a secret code that we are trying to reveal. In our game, though, we are only required to find one bit in the 'code' and the code-maker is under fewer constraints.

6 Experiments

In this section we present empirical work comparing the number of tests needed by the described methods. All experiments were repeated 20 times for random choices of compromised keys. In all experiments, we use $\epsilon = 0.001$ and $R = \{0, 1\}$ Preliminary experiments showed that the number of tests is not particularly sensitive to s, the retention set size, so in all experiments $s = 2$. We only report results for the uniform choice strategy since it is, theoretically, the most challenging strategy to defeat. Preliminary experiments also confirmed this was the case.

All experiments were run using Matlab running under Linux, with 3GB of memory and a 2Ghz Pentium 4 processor. The posterior distribution and information gain were calculated exactly. Both operations scale exponentially with the size of the frontier so we did not evaluate our method on very large frontiers. Elsewhere we describe an efficient algorithm [13], but it is beyond the scope of this paper. The code is optimized to take advantage of Matlab's efficient matrix operations and for a frontier of size 18, test selection takes around 45 seconds.

Table 1 highlights the results of the experiments. It is immediately clear that IGTrace outperforms both other methods in all cases where there is more than 1 compromised key. For larger frontiers and only 1 compromised key, BayesNNL-Trace seems to need slightly fewer tests. But for larger numbers of compromised keys, which is of more interest to us, there is an order of magnitude improvement. Note, too, that there is substantially less variance in the number of tests needed by IGTrace.

BayesNNLTrace compares extremely well with NNLTrace in all cases, as well. One obvious shortcoming of NNLTrace is that it does not automatically take

Table 1. Comparative performance of the three tracing methods for a variety of frontier sizes and clone box sizes. Each entry lists the mean and standard deviation of the number of tests. Column headers indicate the number of compromised keys.

		1	2	4	8
$\|F\| = 8$	*IGTrace*	6±0.8	19±8.0	41±12.6	42±16.1
	BayesNNLTrace	22±3.1	109±19.9	539±168.5	823±231.4
	NNLTrace	271±0.0	605±181.5	919±423.8	1424±160.6
$\|F\| = 10$	*IGTrace*	13±2.0	24±8.4	52±11.1	86±38.7
	BayesNNLTrace	25±4.4	142±34.4	518±270.1	1173±438.1
	NNLTrace	380±54.7	948±299.2	1408±601.2	2632±607.4
$\|F\| = 12$	*IGTrace*	19±2.0	32±9.4	57±18.6	113±36.5
	BayesNNLTrace	28±4.6	148±41.9	784±221.4	1322±602.7
	NNLTrace	481±55.4	1263±415.1	1686±949.3	3562±1223.1
$\|F\| = 14$	*IGTrace*	29±1.0	38±9.5	64±14.7	129±26.1
	BayesNNLTrace	28±3.9	176±34.7	855±244.7	1510±624.5
	NNLTrace	555±51.4	1366±399.4	2326±1221.7	4506±1547.4
$\|F\| = 16$	*IGTrace*	38±1.1	47±9.0	68±14.0	159±41.6
	BayesNNLTrace	29±3.0	188±46.4	914±346.3	2258±923.2
	NNLTrace	637±0.0	1493±478.0	3660±1239.4	4278±1588.7
$\|F\| = 18$	*IGTrace*	49±1.6	57±7.3	86±15.9	150±30.3
	BayesNNLTrace	32±3.0	197±52.0	1203±320.4	1518±705.1
	NNLTrace	728±69.2	1845±488.9	4525±1457.6	6229±3199.4

advantage of situations where the clone box is deterministic. BayesNNLTrace, however, naturally accommodates this scenario, as seen in the first column, where the number of compromised keys is 1. BayesNNLTrace only repeats each test about 7 times.

7 Conclusion

Traitor tracing is an essential component in any broadcast encryption scheme. Previous algorithms, while theoretically efficient, do not offer a practical solution. In this work we have presented two methods, BayesNNLTrace and IGTrace, that significantly improve the number of tests required by the tracing algorithm to defeat a clone box.

In situations where some information about the clone box strategy is obtained, IGTrace can be applied. However, this begs the question: how does one obtain such information? In the future, it is worth investigating methods for learning this strategy along the way, perhaps by representing the clone box strategy as a latent variable.

Similarly, it would also be interesting to investigate the sensitivity of IGTrace to errors in the clone box strategy. We would like it to be such that if the actual strategy deviates from the modeled strategy we are still guaranteed to make accurate diagnoses.

References

1. 4Centity.
2. AACS.
3. A.Fiat and M. Naor. Broadcast encryption. In CRYPTO 93: International Cryptology Conference on Advances in Cryptology, volume 773, pages 480491, London, UK, 1993. Springer-Verlag.
4. A.Fiat B. Chor and M. Naor. Tracing traitors. In CRYPTO 94: International Cryptology Conference on Advances in Cryptology, volume 839, pages 480491, London, UK, 1994. Springer-Verlag.
5. R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification. Wiley-Interscience Publication, 2000.
6. Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. Bayesian Data Analysis, Second Edition. Chapman & Hall/CRC, July 2003.
7. Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. Inf. Process. Lett., 5(1):1517, 1976.
8. Vomlel J. Bayesian networks in mastermind. Proceedings of the 7th Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty., pages 185 190., 2004.
9. Jeff Lotspiech. Personal communication, 2006.
10. David J. C. MacKay. Information Theory, Inference, and Learning Algorithms. Cambridge University Press, 2003. available from http://www.inference.phy.cam.ac.uk/mackay/itila/.
11. Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and tracing schemes for stateless receivers. In CRYPTO 01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, pages 4162, London, UK, 2001. Springer-Verlag.
12. I. Rish,M. Brodie, S.Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez. Adaptive diagnosis in distributed systems. IEEE Transactions on Neural Networks, 16:10881109, 2005.
13. Philip Zigoris. Improved traitor tracing algorithms. Technical report, UC Santa Cruz, 2006.

A New Protocol for Conditional Disclosure of Secrets and Its Applications

Sven Laur[1] and Helger Lipmaa[2]

[1] Helsinki University of Technology, Finland
[2] University College London, UK

Abstract. Many protocols that are based on homomorphic encryption are private only if a client submits inputs from a limited range $\mathcal{S}$. Conditional disclosure of secrets (CDS) helps to overcome this restriction. In a CDS protocol for a set $\mathcal{S}$, the client obtains server's secret if and only if the client's inputs belong to $\mathcal{S}$ and thus the server can guard itself against malformed queries. We extend the existing CDS protocols to work over additively homomorphic cryptosystems for every set from $\mathbf{NP/poly}$. The new construction is modular and easy to apply. As an example, we derive a new oblivious transfer protocol with log-squared communication and a millionaire's protocol with logarithmic communication. We also implement private, universally verifiable and robust multi-candidate electronic voting so that all voters only transmit an encryption of their vote. The only hardness assumption in all these protocols is that the underlying public-key cryptosystem is IND-CPA secure and the plaintext order does not have small factors.

Keywords: Conditional disclosure of secrets, crypto-computing, homomorphic encryption, oblivious transfer, two-party computation.

1 Introduction

Homomorphic encryption is a powerful tool that provides efficient private implementations for many basic operations such as scalar product, oblivious transfer and oblivious polynomial evaluation. However, basic versions of these protocols without zero-knowledge proofs of correctness are secure only in a semihonest model, where all parties submit inputs from a limited range, and are not protected against malicious behaviour. Consequently, a malicious adversary can completely or partially learn the secret inputs. Conditional disclosure of secrets [GIKM00, AIR01], also known as input verification gadget [BGN05], is a protection mechanism against such attacks. Unfortunately, current solutions [AIR01, BGN05] are secure only if the plaintext space has a prime order, whereas most additively homomorphic encryption schemes have a composite plaintext order. We provide the first conditional disclosure of secrets protocol that works in conjunction with *all* currently known additively homomorphic encryption schemes. Hence, we can efficiently and more securely solve many practical problems.

Formally, we consider only two-party protocols between a client and a server, though our results can be extended to the multiparty setting. At the end of such a protocol the client should learn the desired value whereas the server should learn nothing. Our main goal is to achieve *relaxed-security*; that is, the protocol must be secure against malicious clients and semihonest servers. Such a model is widely used in current cryptographic literature [NP99, AIR01] and is well-justified in practical applications: as the number of possible service providers is relatively small compared to the clients, it is possible to force semihonest behaviour with auditing. Moreover, service providers must preserve their reputation and thus they are less likely to act maliciously.

For clarity and brevity, we state our main results in the public key model, where the client is guaranteed to know a valid secret key and the server knows the corresponding public key. The choice of the model is not too restrictive: with a proper initialisation phase all our protocols can be implemented in the standard model, see Sect. 7. On the other hand, such a model enables to prove security of parallel compositions. Composability together with our new basic construction leads to a simpler and more modular way to construct complex protocols. Shortly put, relaxed-security follows directly from the protocol design and there is no need to handcraft the proof. More precisely, we show how to decompose a protocol into elementary tasks that can be efficiently implemented with any additively homomorphic IND-CPA secure cryptosystem, provided that the plaintext order does not have unknown small factors.

In Sect. 3, we establish basic security notions and derive a necessary machinery to analyse parallel compositions. The core results of our papers are presented in Sect. 4. We note that most existing additively homomorphic protocols are based on the possibility of computing the next three basic primitives on ciphertexts: addition of ciphertexts, multiplication with a constant, and *disclose-if-equal* (DIE). In a disclose-if-equal protocol, the server obliviously releases secret β only if the client sends a valid encryption of x, where the coefficient x can be freely chosen by the server. The current cryptographic literature is full of many useful and efficient two-message protocols that are based on these three primitives. Unfortunately, the standard DIE protocol defined say in [AIR01], and then used in many subsequent papers, is secure only if the plaintext space has a prime order and thus can only be used in conjunction with the lifted ElGamal cryptosystem where one has to compute discrete logarithms to decrypt. We provide a new DIE protocol that works in conjunction with *all* currently known additively homomorphic encryption schemes. As a result, we can naturally simplify or extend many protocols that utilise the DIE functionality, e.g. [AIR01, Ste98, Lip05, BK04, FNP04, LLM05].

The rest of the paper provides many useful applications of these generic building blocks. In Sect. 5, we present a two-message protocol for conditional disclosure of secrets (CDS), where the client learns a secret β only if his message $\mathfrak{q}$ is a valid encryption of $\boldsymbol{x} \in \mathcal{S}$, where $\mathcal{S}$ is a publicly known set. Hence, the server can use β as a one-time pad to protect the protocol output, i.e., the client learns nothing unless $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{q}) \in \mathcal{S}$. The latter forms a basis of the CDS transformation that can guard any two-message protocol, where the first message is a vector of ciphertexts, against malicious clients. A slightly extended CDS construction provides an efficient solution to the millionaire problem and conditional oblivious transfer. Another extension of CDS provides a way to implement electronic voting and auctions without non-interactive zero-knowledge proofs in the multi-party setting using threshold-decryption. Finally, we compare our results with conventional cryptographic methods to provide some interesting insights and show the theoretical significance of our results, see Sect. 7.

History. The new DIE protocol, together with the CDS protocol and the CDS transformation date from August 2004 and has been available on eprint since 2005.

2 Cryptographic Preliminaries

Distributions. For a a finite set X, let $\mathcal{U}(X)$ denote the uniform distribution over X and $x \leftarrow X$ denote a uniform draw from X. Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over a discrete support X are statistically ε-close, $\mathcal{D}_1 \overset{\varepsilon}{\sim} \mathcal{D}_2$, if their statistical difference $\max_{S \subseteq X} |\Pr[\mathcal{D}_1 \in S] - \Pr[\mathcal{D}_2 \in S]| \leq \varepsilon$. A shorthand $\mathcal{D}_1 \equiv \mathcal{D}_2$ denotes $\mathcal{D}_1 \overset{0}{\sim} \mathcal{D}_2$.

Homomorphic encryption. A public-key cryptosystem π is defined by three algorithms. A key generation algorithm Gen returns a secret and public key pair $(\mathsf{sk}, \mathsf{pk})$. Corresponding $\mathsf{Enc}_{\mathsf{pk}}(\cdot)$ and $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$ algorithms are used to encrypt and decrypt messages. Let $\mathcal{M}$ and $\mathcal{C}$ denote the corresponding message and ciphertext spaces. Then we require $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(x)) = x$ for every $x \in \mathcal{M}$ and assume that there exists efficient membership test for the ciphertext space $\mathcal{C}$. Privacy of encrypted messages is guaranteed by IND-CPA security. For any *stateful* probabilistic algorithm A, its IND-CPA advantage quantifies the ability to distinguish ciphertexts:

$$\mathsf{Adv}_{\pi}^{\text{IND-CPA}}(A) = 2 \cdot \left| \Pr \left[\begin{matrix} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}, (x_0, x_1) \leftarrow A(\mathsf{pk}), i \leftarrow \{0,1\} \\ c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x_i) : A(x_0, x_1, c) = i \end{matrix} \right] - \frac{1}{2} \right| ,$$

where the probability is taken over coin tosses of all relevant algorithms. A cryptosystem π is (ε, τ)-*IND-CPA-secure* if $\mathsf{Adv}_{\pi}^{\text{IND-CPA}}(A) \leq \varepsilon$ for any τ-time adversary A.

A cryptosystem π is *additively homomorphic*, if $\mathcal{M} = \mathbb{Z}_N$ for some N, and for any $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}$ and valid messages $x_1, x_2 \in \mathcal{M}$ the distribution of products $\mathsf{Enc}_{\mathsf{pk}}(x_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(x_2)$ coincides with the distribution of ciphertexts $\mathsf{Enc}_{\mathsf{pk}}(x_1 + x_2)$. To be precise, the equivalence

$$\mathsf{Enc}_{\mathsf{pk}}(x_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(x_2) \equiv \mathsf{Enc}_{\mathsf{pk}}(x_1 + x_2)$$

must hold for any fixed ciphertext $\mathsf{Enc}_{\mathsf{pk}}(x_1)$. That is, given $\mathsf{Enc}_{\mathsf{pk}}(x_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(x_2)$, even an unbounded adversary learns nothing beyond $x_1 + x_2$. A cryptosystem π is *multiplicatively homomorphic*, if $\mathsf{Enc}_{\mathsf{pk}}(x_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(x_2) \equiv \mathsf{Enc}_{\mathsf{pk}}(x_1 \cdot x_2)$ for any $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}$ and $x_1, x_2 \in \mathcal{M}$, where $\mathcal{M}$ is a multiplicative group where computing the discrete logarithm is hard. In many practical applications, multiplicatively homomorphic cryptosystems Enc are converted to additively homomorphic cryptosystems $\overline{\mathsf{Enc}}$ by using the lifted encryption rule $\overline{\mathsf{Enc}}_{\mathsf{pk}}(x) := \mathsf{Enc}_{\mathsf{pk}}(g^x)$. Such lifted cryptosystems have reduced utility, as the new decryption rule requires computation of discrete logarithms and one can successfully decrypt only a small fraction of ciphertexts.

Many well-known homomorphic cryptosystems are IND-CPA secure under reasonable complexity assumptions, e.g. [Elg85, Pai99, DJ01]. Existing additively homomorphic cryptosystems have a composite plaintext order with large factors. For example, the plaintext order of the Paillier cryptosystem [Pai99] is an RSA modulus and thus its smallest prime factor is approximately $\sqrt{N}$. The Goldwasser-Micali cryptosystem [GM82] is the only known exception, as it is additively homomorphic over $\mathbb{Z}_2$. Such plaintext space is too small for many applications. All known cryptosystems with a large prime plaintext order are multiplicative, e.g., the ElGamal cryptosystem [Elg85].

3 Basic Properties of Two-Message Protocols

Throughout the paper, we consider two-message protocols where a client sends a query $\mathfrak{q}$ to a server that replies with $\mathfrak{a}$, and then the client computes a desired output from $\mathfrak{a}$. The server should learn nothing about the query. The client should learn $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$, where $\boldsymbol{\alpha}$ denotes client's private input vector and $\boldsymbol{\beta}$ denotes server's private input vector. Mostly, we consider the relaxed-security against unbounded clients and computationally bounded servers, but sometimes we consider also the setting where both parties are computationally bounded. A protocol is *correct* if the client always recovers $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$ when both parties are honest. A priori we do not assume correctness from all protocols, as sometimes it is sufficient to know that a client cannot learn anything beyond $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

In the simplest case, the query $\mathfrak{q}$ consists of encrypted inputs $(\alpha_1, \dots, \alpha_m)$ and the server uses properties of additively homomorphic encryption to compose an appropriate reply. We call such protocols *additively homomorphic two-message protocols*. Here, we explicitly assume that the server knows public key pk and thus can efficiently verify that the query consists of valid ciphertexts and ignore malformed queries. Notably, many interesting tasks can be solved with additively homomorphic two-message protocols. Computationally-private information retrieval [AIR01, Ste98, Lip05], solutions to millionaire's problem [BK04, Fis01], and various protocols for privacy-preserving data mining tasks [FNP04, WY04, GLLM04] form only a small set of such protocols.

Relaxed-security in the PKI model. As usual, we define security by comparing the real and ideal model. However, we explicitly assume that the client knows the secret key, the server knows the corresponding public key and only the client can deviate from the protocol specification. Formally, a trusted key generator initially runs the key generation algorithm Gen for a cryptosystem π, and then privately sends $(\mathsf{sk}, \mathsf{pk})$ to the client and pk to the server. In particular, the server knows that pk corresponds to this fixed client. This key pair is then possibly used in many different protocol runs.

Note that the PKI model is normal and even desirable in many applications, e.g. e-voting. Still, we stress that we use the PKI model only for the sake of simplicity of security proofs. In Sect. 7, we show how to replace the trusted key generator by a key transfer protocol with a marginal degradation of security.

Since the server obtains no output and is always semihonest, we can decompose the standard security definition into two orthogonal requirements: client-privacy and server-privacy. A two-message protocol is (ε, τ)-*client-private*, if for any τ-time stateful adversary A, the next inequality holds:

$$2 \cdot \left| \Pr \left[\begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}, (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1) \leftarrow A(\mathsf{pk}), \\ i \leftarrow \{0,1\}, \mathfrak{q} \leftarrow \mathfrak{q}_{\mathsf{pk}}(\boldsymbol{\alpha}_i) : A(\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \mathfrak{q}) = i \end{array} \right] - \frac{1}{2} \right| \leq \varepsilon \ ,$$

where $\mathfrak{q}_{\mathsf{pk}}(\boldsymbol{\alpha}_i)$ denotes the first message computed by the honest client. Server-privacy has a slightly more complicated definition, since we must transform any efficient adversary from the real world to an efficient adversary in the ideal model, where a trusted third party (TTP) computes $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Hence, the definition incorporates a simulator Sim and a distinguisher B and we need to explicitly quantify their efficiency. The simulator

Sim gets $(\mathsf{sk}, \mathfrak{q})$ as an input and can send $\boldsymbol{\alpha}^*$ once to the TTP. Then Sim obtains the value of $f^* = f(\boldsymbol{\alpha}^*, \boldsymbol{\beta})$ and can proceed with the simulation. For brevity, let us define

$$p_{\mathrm{r}} = \Pr\left[(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}, (\boldsymbol{\beta}, \mathfrak{q}) \leftarrow A(\mathsf{sk}), \mathfrak{a} \leftarrow \mathfrak{a}_{\mathsf{pk}}(\mathfrak{q}, \boldsymbol{\beta}) : B(\boldsymbol{\beta}, \mathfrak{q}, \mathfrak{a}) = 1\right] \ ,$$
$$p_{\mathrm{i}} = \Pr\left[(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}, (\boldsymbol{\beta}, \mathfrak{q}) \leftarrow A(\mathsf{sk}), \hat{\mathfrak{a}} \leftarrow \mathsf{Sim}_{\mathsf{sk}}(\mathfrak{q}, f^*) : B(\boldsymbol{\beta}, \mathfrak{q}, \hat{\mathfrak{a}}) = 1\right] \ ,$$

where $\mathfrak{a}(\mathfrak{q}, \boldsymbol{\beta})$ denotes the answer of the honest server with the input $\boldsymbol{\beta}$ to the query $\mathfrak{q}$. A protocol implements $(\tau, \delta, t, \varepsilon)$*-server-privately* a function f, if for any τ-time adversary A there exists a $(t + \delta)$-time simulator Sim such that $|p_{\mathrm{r}} - p_{\mathrm{i}}| \leq \varepsilon$ for any t-time distinguisher B. In the information-theoretical setting, algorithms A, Sim and B are unbounded. A protocol is ε*-server-private* if for any adversary A there exists a simulator Sim such that their output distributions are statistically ε-close. We say that a protocol is $(\varepsilon_1, \tau; \varepsilon_2)$*-relaxed-secure* if it is (ε_1, τ)-client-private and ε_2-server-private. Relaxed-security is widely used standard security assumption, see [NP99, AIR01].

Extractability and simulatability. Usually, the client-privacy follows directly from security assumptions. For example, additively homomorphic protocols are client-private by the construction, provided that the cryptosystem is IND-CPA secure. Proofs of server-privacy can be significantly simplified by considering the following notions of extractability and simulatability. As client can be malicious, the simulator Sim must somehow deduce the intended input $\boldsymbol{\alpha}^*$. In the PKI model, the simulator can use sk to determine the input $\boldsymbol{\alpha}^*$ directly from $\mathfrak{q}$. A two-message protocol is *extractable* if there exists an efficient algorithm $\mathsf{Ext}_{\mathsf{sk}}(\cdot)$ such that $\mathsf{Ext}_{\mathsf{sk}}(\mathfrak{q}_{\mathsf{pk}}(\boldsymbol{\alpha})) = \boldsymbol{\alpha}$ for all valid inputs and $\mathsf{Ext}_{\mathsf{sk}}(\mathfrak{q}) = \perp$ for all invalid queries $\mathfrak{q}$ that do not correspond to any input.

In many protocols, the server's reply can be perfectly or almost perfectly simulated knowing only the corresponding client's output f^* and a secret key sk. We formalise this as simulatability. Consider a protocol transcript $(\mathfrak{q}, \mathfrak{a})$ between the honest client and server. Let $f^* = f(\boldsymbol{\alpha}, \boldsymbol{\beta})$ be the corresponding client's output. Then the server's reply is ε_2*-simulatable* if there exists an efficient algorithm $\mathsf{Sim}^*_{\mathsf{sk}}$ such that the output distributions $(\mathfrak{q}, \mathfrak{a})$ and $(\mathfrak{q}, \hat{\mathfrak{a}})$ are statistically ε_2-close even for a fixed $\mathfrak{q}$, where $\hat{\mathfrak{a}} \leftarrow \mathsf{Sim}^*_{\mathsf{sk}}(\mathfrak{q}, f^*)$. The notion of (t, ε_2)-simulatability is defined analogously. Extractability together with simulatability implies server-privacy:

Theorem 1. *If a two-message protocol is extractable, ε_2-simulatable and the server ignores malformed queries, then the protocol is also ε_2-server-private in the PKI model.*

Proof. We construct a universal simulator Sim as follows. If the query $\mathfrak{q}$ is malformed then the simulator ignores it. Otherwise, Sim extracts the intended input $\boldsymbol{\alpha}^* \leftarrow \mathsf{Ext}_{\mathsf{sk}}(\mathfrak{q})$ and sends $\boldsymbol{\alpha}^*$ to the TTP. Given the reply $f^* = f(\boldsymbol{\alpha}^*, \boldsymbol{\beta})$ from the TTP, the simulator uses $\mathsf{Sim}^*_{\mathsf{sk}}(\mathfrak{q}, f^*)$ to simulate the reply $\hat{\mathfrak{a}}$. Since malformed queries are discarded in both worlds, the distributions $(\boldsymbol{\beta}, \mathfrak{q}, \mathfrak{a})$ and $(\boldsymbol{\beta}, \mathfrak{q}, \hat{\mathfrak{a}})$ are statistically ε_2-close. $\square$

Forked composition. We can use Thm. 1 to prove that a parallel composition of extractable and simulatable protocols preserves server-privacy. It makes sense to consider protocols that share the query phase as we can always merge different queries into a single query. Let two-message protocols $\Pi_1, \ldots, \Pi_s$ share the first message $\mathfrak{q}$. Then the *forked composition* $\mathsf{Forked}[\Pi_1, \ldots, \Pi_s]$ is defined as follows:

1. The client computes the query $\mathfrak{q}$ and sends it to the server.
2. The server uses $\mathfrak{q}$ to compute replies $\mathfrak{a}_1, \dots, \mathfrak{a}_s$ according to $\Pi_1, \dots, \Pi_s$.
3. The server sends $\mathfrak{a}_1, \dots, \mathfrak{a}_s$ to the client.
4. The client computes the private output $(f_1, \dots, f_s)$ according to $\Pi_1, \dots, \Pi_s$.

It is easy to prove that a client can learn nothing beyond $f_1(\boldsymbol{\alpha}, \boldsymbol{\beta}), \dots, f_s(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

Theorem 2. *Let* $\Pi_1, \dots, \Pi_s$ *be extractable and respectively* ε_i*-simulatable implementations of functionalities* f_i*. Then the composition* $\mathsf{Forked}[\Pi_1, \dots, \Pi_s]$ *is an extractable and* $(\varepsilon_1 + \dots + \varepsilon_s)$*-simulatable implementation of the functionality* $f = (f_1, \dots, f_s)$*.*

Proof. Extractability is clear. By the definition of simulatability, there exist simulators $\mathsf{Sim}^*_{\mathsf{sk},i}$ that output simulated replies $\hat{\mathfrak{a}}_i$ such that $(\mathfrak{q}, \mathfrak{a}_i)$ and $(\mathfrak{q}, \hat{\mathfrak{a}}_i)$ are statistically ε_i-close even for fixed $\mathfrak{q}$. Now, define a simulator $\mathsf{Sim}^*_{\mathsf{sk}}$ that given $\mathfrak{q}$ and $f^* = (f_1(\boldsymbol{\alpha}^*, \boldsymbol{\beta}), \dots, f_s(\boldsymbol{\alpha}^*, \boldsymbol{\beta}))$ runs $\mathsf{Sim}^*_{\mathsf{sk},i}(\mathfrak{q}, f^*_i)$ for $i \in \{1, \dots, s\}$ and outputs $\hat{\mathfrak{a}}_1, \dots, \hat{\mathfrak{a}}_s$. By the construction, the distributions $(\mathfrak{q}, \mathfrak{a}_1, \dots, \mathfrak{a}_s)$ and $(\mathfrak{q}, \hat{\mathfrak{a}}_1, \dots, \hat{\mathfrak{a}}_s)$ are statistically $(\varepsilon_1 + \dots + \varepsilon_s)$-close even for a fixed $\mathfrak{q}$ and the simulatability follows. □

Reducing communication further with CPIR. In many two-message protocols, the client must access only a short part of the reply $\mathfrak{a}$ to recover the output $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$ whereas the rest of $\mathfrak{a}$ consists of random noise. Hence, we can significantly decrease the total communication $|\mathfrak{q}| + |\mathfrak{a}|$, if the client could fetch only useful parts of $\mathfrak{a}$. The latter can be done using *computationally private information retrieval* (CPIR). In a 1-out-of-n CPIR protocol, the server maintains a database $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$ of ℓ-bit strings and the client can fetch β_i so that a computationally bounded server learns nothing. The basic properties of CPIR protocols are determined by parameters n and ℓ. It is trivial to achieve communication complexity $\Theta(n\ell)$ just by sending the whole database so one considers only CPIR protocols with sublinear communication. There is a wide range of such protocols. Recent protocols achieve communication that is low-degree polylogarithmic in the database size, see [Lip05, GR05] for further references.

Now, assume that the server's reply has a structure $\mathfrak{a} = (\mathfrak{a}_1, \dots, \mathfrak{a}_n)$ and the client needs to recover at most t elements. Then the client can use t parallel CPIR queries to fetch desired parts $\mathfrak{a}_{i_1}, \dots, \mathfrak{a}_{i_t}$. Note that the CPIR queries can be sent together with the protocol Π messages, provided that the CPIR instance is run independently from Π or joining queries does not decrease client-privacy. Server-privacy cannot decrease, as the replies of CPIR queries are computed from the original reply $\mathfrak{a}$.

4 Three Basic Crypto-computing Primitives

"Crypto-computing" is often used to describe two-message protocols, where a server uses some basic operations on client's garbled inputs to compute reply that reveals only $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$. The first comparative study [SYY99] showed how to crypto-compute predicates with logarithmic circuit depth using the Goldwasser-Micali cryptosystem. Later, this construction was somewhat generalised to compute the greater-than predicate [Fis01]. Here, we provide three basic crypto-computing primitives for additively homomorphic cryptosystems with large factors of the plaintext space. Note that the server can crypto-compute ciphertexts of sums and products with one public factor obliviously from ciphertexts, as

$$\mathsf{Enc}_{\mathsf{pk}}(x_1 + x_2) \equiv \mathsf{Enc}_{\mathsf{pk}}(x_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(x_2) \cdot \mathsf{Enc}_{\mathsf{pk}}(0) \tag{1}$$

$$\mathsf{Enc}_{\mathsf{pk}}(x \cdot y) \equiv \mathsf{Enc}_{\mathsf{pk}}(y)^x \cdot \mathsf{Enc}_{\mathsf{pk}}(0) \tag{2}$$

hold by the definition of additively homomorphic cryptosystems. Here the multiplication by $\mathsf{Enc}_{\mathsf{pk}}(0)$ is necessary to re-randomise the replies.

But there is also a third generic operation that implicitly "tests" whether a ciphertext c is an encryption of x. The existence of this operation depends additionally on the order of the plaintext group. More precisely, a *disclose-if-equal* (DIE) protocol allows releasing of a secret β only if $\mathsf{Dec}_{\mathsf{sk}}(c) = x$ where the server can freely choose x. The idealised functionality of DIE protocol is defined as follows

$$f(\alpha, \beta) = \begin{cases} \beta, & \text{if } \alpha = x \ , \\ \perp, & \text{if } \alpha \neq x \ . \end{cases}$$

The simplest implementation of DIE protocol was given in the paper [AIR01]:

1. The client sends $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\alpha)$ to the server.
2. If $c \in \mathcal{C}$ then the server sends a reply $\mathfrak{a} \leftarrow (c \cdot \mathsf{Enc}_{\mathsf{pk}}(-x))^r \cdot \mathsf{Enc}_{\mathsf{pk}}(\beta)$ for $r \leftarrow \mathcal{M}$.
3. The client outputs $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{a}) = (\alpha - x)r + \beta$.

If the plaintext space has a prime order, then $(\alpha - x)r$ has uniform distribution over $\mathcal{M}$ when $x \neq \alpha$. Consequently, the protocol is perfectly simulatable: if $f(\alpha, \beta) = \perp$ a simulator should output a random encryption $\mathsf{Enc}_{\mathsf{pk}}(m)$ for $m \leftarrow \mathcal{M}$ and $\mathsf{Enc}_{\mathsf{pk}}(\beta)$ otherwise. Therefore, the basic DIE protocol is also relaxed-secure.

On the other hand, the protocol is not correct, since the client obtains a random output when $\mathsf{Dec}_{\mathsf{sk}}(c) \neq x$. If x is public then the correctness is not an issue, as the client knows whether $\mathsf{Dec}_{\mathsf{sk}}(c) = x$ or not. Otherwise, the construction guarantees only that the client learns nothing about β when $\mathsf{Dec}_{\mathsf{sk}}(c) \neq x$. Moreover, if the server sets the first k-bits of β to 0, then the honest client can detect $\alpha \neq x$ with failure probability 2^{-k}, i.e., there is a trade-off between reliability and throughput.

Unfortunately, the basic DIE protocol is not secure if the message space has a composite order. As an example, consider the Paillier cryptosystem, where $N = pq$ is an RSA modulus. If a malicious client sends $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(p + x)$ then $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{a}) = \beta + rp \mod N$ and the client can recover $\beta \mod p$ although $\mathsf{Dec}_{\mathsf{sk}}(c) \neq x$. Since the DIE protocol is a building block in many existing protocols, then such leakage might cause a domino effect that can completely reveal server's input. For example, the circuit CDS protocol in Sect. 5 is extremely vulnerable against such attacks. Therefore, we devise a new DIE protocol that works in conjunction with all currently known additively homomorphic cryptosystems. As a result, we can naturally simplify many protocols [AIR01, Ste98, Lip05, BK04, FNP04, LLM05] that use the lifted ElGamal cryptosystem or zero-knowledge correctness proofs to guarantee security of the DIE protocol.

New general construction for DIE. Server-privacy of the basic DIE protocol hinges on the fact that $\alpha\mathbb{Z}_N = \{\alpha r : r \in \mathbb{Z}_N\} = \mathbb{Z}_N$ for any $\alpha \neq 0$. If the message space

```
Query phase:
    The client sends $\mathfrak{q} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\alpha)$ to the server.
Transfer phase:
    If the ciphertext is invalid $\mathfrak{q} \notin \mathcal{C}$ then the server returns $\perp$.
    Otherwise, the server returns $\mathfrak{a} \leftarrow (c \cdot \mathsf{Enc}_{\mathsf{pk}}(-x))^r \cdot \mathsf{Enc}_{\mathsf{pk}}(\mathsf{encode}(\beta))$ for $r \leftarrow \mathcal{M}$.
Post-processing:
    The client computes $y = \mathsf{Dec}_{\mathsf{sk}}(\mathfrak{a})$ and returns $\mathsf{decode}(y)$.
```

Protocol 1: Disclose-if-equal protocol of ℓ-bit secrets for the constraint $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{q}) = x$

contains non-trivial additive subgroups (ideals) $\{0\} \neq \mathbb{G} \subsetneq \mathbb{Z}_N$ then the client can choose a ciphertext c so that the reply $\mathfrak{a}$ enables to restore the coset $\beta + \mathbb{G}$. Consequently, a malicious client can learn up to $\log_2 N - \log_2 \Phi$ bits of information, where Φ is the minimal size of the non-trivial subgroup $\mathbb{G}$. To seal the leakage, we must use a probabilistic encoding for β such that the total entropy of $\mathbb{G}$ together with the encoding $\mathsf{encode}(\beta)$ is roughly $\log_2 N$. Let us define an encoding for ℓ-bit strings

$$\begin{aligned}\mathsf{encode}(\beta) &= \beta + 2^\ell \cdot t \mod N \quad \text{for} \quad t \leftarrow \mathbb{Z}_T \ , \\ \mathsf{decode}(y) &= (y \mod N) \mod 2^\ell \ ,\end{aligned}$$

where $T = \lfloor 2^{-\ell} \cdot N \rfloor$ and $\ell < \lfloor \log_2 N \rfloor$. As there are no modular wrappings, the decoding is always correct. More importantly, Prot. 4 is now secure for small enough ℓ.

Theorem 3. *Let π be an additively homomorphic cryptosystem such that the smallest factor of the plaintext order is larger than $\gamma > 2$. Then Protocol 4 for transferring ℓ-bit strings is extractable and $(2^{\ell-1}/\gamma)$-simulatable.*

Proof. Extractability is clear and thus we consider only simulatability. If $\alpha \neq x$, then by construction $y = \mathsf{encode}(\beta) + g$ where g is chosen uniformly from a non-zero subgroup $\mathbb{G} \subseteq \mathbb{Z}_N$. If $\mathbb{G} = \mathbb{Z}_N$ then y is uniformly distributed over $\mathbb{Z}_N$. Otherwise $\mathbb{G}$ can be represented as $p\mathbb{Z}_N$, where p is a non-trivial factor of N, and $y \mod p \equiv \beta + 2^\ell \cdot t \mod p$, where $t \leftarrow \mathbb{Z}_T$ and $T = \lfloor 2^{-\ell} \cdot N \rfloor$. Since 2 and p are relatively prime, $\{2^\ell \cdot t : t \in \mathbb{Z}_p\} = \mathbb{Z}_p$ and the term $2^\ell \cdot t \mod p$ covers all elements of $\mathbb{Z}_p$ almost uniformly. More precisely, the elements of $\mathbb{Z}_p$ can be divided into two sets:

$$\begin{aligned}\mathcal{T}_0 &= \left\{c \in \mathbb{Z}_p : \Pr[\beta + 2^\ell \cdot t \mod p = c] = \tfrac{a}{T}\right\} \qquad \text{with} \qquad |\mathcal{T}_0| = p - b \ , \\ \mathcal{T}_1 &= \left\{c \in \mathbb{Z}_p : \Pr[\beta + 2^\ell \cdot t \mod p = c] = \tfrac{a+1}{T}\right\} \qquad \text{with} \qquad |\mathcal{T}_1| = b \ ,\end{aligned}$$

where $a = \left\lfloor \frac{T}{p} \right\rfloor$ and $b = T - ap$. Consequently, the statistical difference between $y \mod p$ and the uniform distribution $\mathcal{U}(\mathbb{Z}_p)$ can be expressed as

$$\varepsilon = \frac{|\mathcal{T}_0|}{2} \cdot \left(\frac{1}{p} - \frac{a}{T}\right) + \frac{|\mathcal{T}_1|}{2} \cdot \left(\frac{a+1}{T} - \frac{1}{p}\right) = \frac{b(p-b)}{Tp} \leq \frac{p}{4T} \leq \frac{N}{4\gamma T} \ ,$$

as $p(p-b) \leq p^2/4$ and $p \leq N/\gamma$. Since $2^{\ell+1} \leq N$ we get $T = \lfloor 2^{-\ell}N \rfloor \geq N/2^{\ell+1}$ and thus $\varepsilon \leq 2^{\ell-1}/\gamma$. Now note that the distributions $\mathsf{encode}(\beta)+\mathcal{U}(p\mathbb{Z}_N)$ and $\mathcal{U}(\mathbb{Z}_N)$ are still ε-close, as we can express

$$\Pr\left[\mathsf{encode}(\beta)+\mathcal{U}(p\mathbb{Z}_N) = c \mod N\right] = \frac{p}{N} \cdot \Pr\left[\mathsf{encode}(\beta) = c \mod p\right] \ .$$

Hence, we can use a simulator $\mathsf{Sim}^*_{\mathsf{sk}}(\mathfrak{q}, f^*)$ that outputs $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{encode}(\beta))$ if $f^* = \beta$ and $\mathsf{Enc}_{\mathsf{pk}}(m)$ for $m \leftarrow \mathbb{Z}_N$ otherwise. □

Corollary 1. *Let π be an (τ, ε_1)-IND-CPA-secure additively homomorphic cryptosystem such that the smallest factor of the plaintext order is larger than $\gamma > 2$. Then Protocol 4 for transferring ℓ-bit strings is $(\tau, \varepsilon_1; \varepsilon_2)$-relaxed-secure for $\varepsilon_2 = 2^{\ell-1}/\gamma$.*

The maximal throughput of DIE protocol. First, note that if we want to achieve ε-server-privacy then we must choose $\ell = \lfloor \log_2(2\varepsilon\gamma) \rfloor$, where γ is the lower bound to non-trivial factors of N. Usually, it is sufficient to take $\varepsilon = 2^{-80}$ and thus N cannot have smaller factors than 2^{80} if the server wants to release Boolean secrets. For the Paillier cryptosystem the smallest factor of N is approximately $\sqrt{N}$, and consequently, one can transfer $\ell = \lfloor \log_2(2\sqrt{N}\varepsilon) \rfloor \approx 0.5 \log_2 N + \log_2 \varepsilon$ bits. For standard 1024-bit RSA modulus and $\varepsilon = 2^{-80}$, one can take $\ell = 433$.

As our DIE protocol is extractable and simulatable, a forked composition of t protocols enables transfer of a $t\ell$-bit secret, where the achieved server-privacy is $\frac{|\beta|}{\ell\gamma} \cdot 2^{\ell-1}$. Smaller values of ℓ increase the maximal length of β but also decrease the ratio between the desired communication $|\beta|$ and the total communication $|\mathfrak{q}| + |\mathfrak{a}|$ and make the protocol less efficient. In other words, a bad encoding $\mathsf{encode}(\beta)$ with a small capacity can significantly decrease efficiency. As our target distribution is $\mathcal{U}(\mathbb{Z}_N)$ then it is straightforward to derive entropy bounds for the capacity: $H(\mathbb{Z}_N) \approx H(\mathsf{encode}(\beta) + p\mathbb{Z}_N) \leq H(\mathsf{encode}(\beta)) + H(p\mathbb{Z}_N) \leq \log_2 |\mathsf{encode}(\beta)| + H(p\mathbb{Z}_N)$, where $|\mathsf{encode}(\beta)|$ denotes the size of the support. As the encoding must be uniquely decodable, the capacity of a single reply $\ell \leq \log_2 \frac{N}{|\mathsf{encode}(\beta)|} \lesssim \min_p H(p\mathbb{Z}_n) = \log_2 \varPhi$, where $\varPhi$ is the smallest prime factor of N. Thus, the encoding is optimal up to a constant additive term $\log_2 \varepsilon$. The result can be generalised for any target distribution using a more detailed analysis.

5 Generic Construction for Conditional Disclosure of Secrets

Many protocols are secure only if client submits inputs $\boldsymbol{\alpha}$ from a limited range $\mathcal{S}$. Cleverly chosen $\boldsymbol{\alpha} \notin \mathcal{S}$ can either partially or completely reveal the server's input $\boldsymbol{\beta}$. Therefore, the server must somehow verify that $\boldsymbol{\alpha} \in \mathcal{S}$. Classically, this is done by a zero-knowledge proof that $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c}) \in \mathcal{S}$. However, this either increases the number of messages or requires a security model with a common reference string or random oracles. A conditional disclosure of secrets (CDS) reaches the same goal without extra messages and exotic assumptions. In a CDS protocol, the client should learn a secret β only if $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{q}) \in \mathcal{S}$, where the query vector $\mathfrak{q}$ consists of ciphertexts $\mathsf{Enc}_{\mathsf{pk}}(\alpha_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(\alpha_m)$ and the set $\mathcal{S}$ is public. Since the server can use β as a one-time pad to encrypt the original reply $\mathfrak{a}$, the client learns nothing about the outputs of the original protocol if $\boldsymbol{\alpha} \notin \mathcal{S}$ and the modified protocol becomes server-private.

A CDS protocol can be straightforwardly constructed as a forked composition of individual DIE protocols for $\{\mathsf{Dec}_{\mathsf{sk}}(c) = x\}_{x \in \mathcal{S}}$ that share the same secret β but such composition is inefficient. Therefore, we show how to use Benaloh-Leichter secret sharing scheme [BL88] together with slightly extended DIE protocols to achieve a more computation and communication efficient CDS protocol (*circuit CDS*).

Conjunctive affine zero tests. First, we present an optimisation for specific sets. Recall that our DIE protocol is secure since $\mathsf{encode}(\beta) + \mathcal{U}(\mathbb{G}) \overset{\varepsilon}{\sim} \mathcal{U}(\mathbb{Z}_N)$ if $\mathbb{G} \neq \{0\}$. Similarly, we can construct CDS protocols for *conjunctive affine zero tests* $\Psi_0(\boldsymbol{\alpha}) = \bigwedge_{j=1}^{v} [\sum_{i=1}^{m} s_{ij}\alpha_i \overset{?}{=} x_j]$, where $\{x_i\}$ and $\{s_{ij}\}$ are public constants:

1. The client sends $\mathfrak{q} = (c_1, \ldots, c_m)$ where $c_i = \mathsf{Enc}_{\mathsf{pk}}(\alpha_i)$.
2. The server halts if some $c_1, \ldots, c_n$ is not a valid ciphertext, otherwise it replies $\mathfrak{a} = \prod_{j=1}^{v} \left(\prod_{i=1}^{m} c_i^{s_{ij}} \cdot \mathsf{Enc}_{\mathsf{pk}}(-x_j)\right)^{r_j} \cdot \mathsf{Enc}_{\mathsf{pk}}(\mathsf{encode}(\beta))$ for $r_1, \ldots, r_v \leftarrow \mathbb{Z}_N$.
3. The client restores $y = \mathsf{Dec}_{\mathsf{sk}}(\mathfrak{a})$ and outputs $\mathsf{decode}(y)$.

As $y = \sum_{j=1}^{v} (\sum_{i=1}^{m} \alpha_i s_{ij} - x_i) r_j + \mathsf{encode}(\beta) = \mathsf{encode}(\beta) + \mathbb{G}_1 + \cdots + \mathbb{G}_v$, then $y = \mathsf{encode}(\beta) + \mathcal{U}(\mathbb{G})$ for a non-zero sub-group $\mathbb{G}$ if some zero-tests do not hold. The latter follows from the fact that $r_1, \ldots, r_v$ are independently chosen. Hence, the claims of Thm. 3 hold also for the CDS protocol given above. Of course, when the plaintext order is prime then there is no need to use probabilistic encoding and we can use the construction given in [AIR01]. Notably, such simplified construction has been used in [BGN05] together with a cryptosystem that has a composite plaintext order. Paradoxically, the latter construction is still computationally secure, as the client must compute arbitrary discrete logarithms to recover a coset $\beta + \mathbb{G}$.

Circuit CDS protocol. For any set $\mathcal{S}$, we can write the predicate $\Psi_{\mathcal{S}}(\boldsymbol{\alpha}) := [\boldsymbol{\alpha} \in \mathcal{S}]$ as a monotonous combination of affine zero tests, i.e., the formula consists of Boolean operations $\wedge$ and $\vee$ together with atomic terms $\Psi_0(\boldsymbol{\alpha}) = \bigwedge_{j=1}^{v} [\sum_{i=1}^{m} s_{ij}\alpha_i \overset{?}{=} x_j]$. For efficiency reasons, we might express the input $\boldsymbol{\alpha}$ as a bit-vector. The server can later use properties of additively homomorphic encryption to restore the original ciphertexts.

First, the server uses the Benaloh-Leichter secret sharing scheme to assign sub-secrets β_i to each leaf test $\Psi_0(\boldsymbol{\alpha})$ so that the client can reconstruct the secret $\beta \in \{0,1\}^{\ell}$ if $\Psi(\boldsymbol{\alpha})$ holds and the secrets of true leaves are revealed. Fig. 1 illustrates how secret β is propagated through the circuit of $\Psi(\alpha) = [\alpha > x]$ without optimisation. Namely, the master secret β is assigned to the topmost gate of the circuit. For every $\vee$-gate, the output secret is just pushed downwards. For every $\wedge$-gate ψ with u children and a secret β_ψ assigned to it, sub-secrets $\beta_1, \ldots, \beta_{u-1} \leftarrow \{0,1\}^{\ell}$ and $\beta_u \leftarrow \beta_\psi - \beta_1 - \cdots - \beta_{u-1} \mod 2^{\ell}$ are assigned to the children. One can also use threshold operations: $\textsc{Thr}_v(x_1, \ldots, x_s) = 0$ if and only if at least v values x_j are equal to 1. For a $\textsc{Thr}_v$ gate, generate a random $(v-1)$-degree polynomial f_ψ with $f_\psi(0) = \beta_\psi$ and assign the secret $f_\psi(i)$ to its ith child. Finally, the server uses a forked composition of CDS protocols for leaf tests Ψ_0 to release sub-secrets associated to each leaf. The client recomputes the secret from leaf values by inversely following the secret generation.

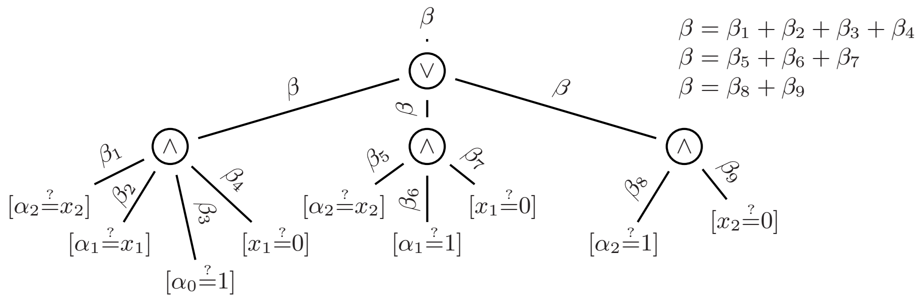

β
$\vee$
$\wedge$
$\beta = \beta_1 + \beta_2 + \beta_3 + \beta_4$
$\beta = \beta_5 + \beta_6 + \beta_7$
$\beta = \beta_8 + \beta_9$
β_1
β_2
β_3
β_4
β_5
β_6
β_7
β_8
β_9
$[\alpha_2 \overset{?}{=} x_2]$
$[\alpha_1 \overset{?}{=} x_1]$
$[\alpha_0 \overset{?}{=} 1]$
$[x_1 \overset{?}{=} 0]$
$[\alpha_1 \overset{?}{=} 1]$
$[\alpha_2 \overset{?}{=} 1]$
$[x_2 \overset{?}{=} 0]$

Fig. 1. An unoptimised circuit for $\Psi(\alpha) = [\alpha > x]$ where secrets are pushed down to DIE leafs. The circuit can be further optimised by replacing $\wedge$-gates with conjunctive affine equality tests.

Theorem 4. *If the leaf CDS protocol is extractable and ε_2-simulatable, then the circuit CDS protocol for $\Psi_{\mathcal{S}}$ is extractable and $\mathcal{L}(\Psi_{\mathcal{S}}) \cdot \varepsilon_2$-simulatable, where $\mathcal{L}(\Psi_{\mathcal{S}})$ is the number of leaves. If the cryptosystem is (τ, ε_1)-IND-CPA secure and $\mathfrak{q}$ consists of m ciphertexts, then the protocol is $(\tau - \mathcal{O}(1), m\varepsilon_1; \mathcal{L}(\Psi_{\mathcal{S}}) \cdot \varepsilon_2)$-relaxed-secure.*

Proof. Given the main secret β it is straightforward to reconstruct the leaf-level secrets. Otherwise, if $\Psi_{\mathcal{S}}(\boldsymbol{\alpha}) = 0$ then the sub-secrets β_i that are assigned to true atoms $\Psi_0(\boldsymbol{\alpha}) = 1$ are independent and are uniformly distributed. Hence, a world with $\mathcal{L}(\Psi_{\mathcal{S}})$ ideally implemented leaf CDS protocols can be perfectly simulated in the world where β is released only if $\boldsymbol{\alpha} \in \mathcal{S}$. Now, the simulatability follows directly from Thm. 2. The second claim follows from Thm. 1 and the basic properties of IND-CPA encryption. □

If the CDS protocol is based on the new DIE protocol, then we can estimate how many bits are needed to transfer ℓ-bit secrets. For the 1024-bit Paillier cryptosystem and 2^{-80}-sever-privacy, a single ciphertext can fit 393 bits provided that the corresponding circuit has less than 2^{40} leaves; the message expansion is roughly $|\mathfrak{a}| / \ell \approx 5.2 \cdot \mathcal{L}(\Psi)$.

As negations can be expressed by conjunctive affine zero tests, then they can appear only in the leaf level, i.e., the formula $\Psi(\boldsymbol{\alpha})$ must be in a negation normal form (NNF). Many practically interesting sets have compact NNF-s, but for some circuits Ψ such normal form is exponentially larger. We can circumvent the problem by using auxiliary inputs $\boldsymbol{w}$. Consider the circuit representation of Ψ that consists of unary $\neg$-gates and binary $\wedge$- and $\vee$-gates. Denote all output wires of logical gates by auxiliary labels w_i. Now, we can represent assignments $w_u \leftarrow w_s \wedge w_t$ and $w_u \leftarrow w_s \vee w_t$ with the formulae

$$[w_u \overset{?}{=} 1] \wedge [w_s \overset{?}{=} 1] \wedge [w_t \overset{?}{=} 1] \vee [w_u \overset{?}{=} 0] \wedge [w_s \overset{?}{=} 0] \vee [w_u \overset{?}{=} 0] \wedge [w_s \overset{?}{=} 0] \ .$$
$$[w_u \overset{?}{=} 0] \wedge [w_s \overset{?}{=} 0] \wedge [w_t \overset{?}{=} 0] \vee [w_u \overset{?}{=} 1] \wedge [w_s \overset{?}{=} 1] \vee [w_u \overset{?}{=} 1] \wedge [w_s \overset{?}{=} 1] \ ,$$

and $w_u \leftarrow \neg w_s$ as $[w_u \overset{?}{=} 0] \wedge [w_s \overset{?}{=} 1] \vee [w_u \overset{?}{=} 1] \wedge [w_s \overset{?}{=} 0]$. Therefore, we can in principle construct a new formula $\overline{\Psi}(\boldsymbol{\alpha}, \boldsymbol{w})$ in NNF such that $\Psi(\boldsymbol{\alpha}) = 1 \iff \exists \boldsymbol{w} : \overline{\Psi}(\boldsymbol{\alpha}, \boldsymbol{w}) = 1$ and the size of $\overline{\Psi}(\boldsymbol{\alpha}, \boldsymbol{w})$ is proportional to the gate count of $\Psi(\boldsymbol{\alpha})$. Consequently, we can always construct efficient circuit CDS protocols for efficiently recognisable sets.

Query phase:
The client sends $\mathfrak{q} = (c_1, \ldots, c_m)$ to the server, where $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\alpha_i)$ for $i \in \{1, \ldots, m\}$.

Transfer phase:
The server computes the reply $\mathfrak{a} = (d_1, \ldots, d_n)$ according to the original protocol Π
The server applies one-time pad $e_i \leftarrow d_i \cdot \mathsf{Enc}_{\mathsf{pk}}(t_i)$ for $t_i \leftarrow \mathbb{Z}_N$ and $i \in \{1, \ldots, m\}$.
The server computes the CDS reply $\mathfrak{a}_{\mathrm{cds}}$ for $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{q}) \in \mathcal{S}$ with secret a $\beta = t_1 \| \ldots \| t_m$.
The server replies $(e_1, \ldots, e_n)$ and $\mathfrak{a}_{\mathrm{cds}}$.

Post-processing:
The client recovers the secrets t_i from $\mathfrak{a}_{\mathrm{cds}}$ and computes $\hat{d}_i \leftarrow e_i \cdot \mathsf{Enc}_{\mathsf{pk}}(-t_i)$.
Next, the client proceeds with the original protocol Π.

Protocol 2: DS transformation for additively homomorphic two-message protocols

CDS transformation. It is straightforward to use a CDS protocol to transform any additively homomorphic two-message protocol that is secure in the semihonest model to a modified two-message protocol that is relaxed-secure. Let the query $\mathfrak{q}$ consist of m ciphertexts $(c_1, \ldots, c_m)$. Protocol Π is secure in the semihonest model, when there exist a set of valid inputs $\mathcal{S}$ such that the client learns only $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$, provided that $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{q}) = (\alpha_1, \ldots, \alpha_m) \in \mathcal{S}$. Let us use a sufficiently long secret β as a one-time pad to decrypt the original reply $\mathfrak{a}$ and release the secret only if $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{q}) \in \mathcal{S}$. Then the corresponding protocol is clearly relaxed-secure. In many cases, the reply $\mathfrak{a}$ consists of re-randomised ciphertexts and we can reduce the length of the secret β, see Prot. 5.

Theorem 5. *If the two-message additively homomorphic protocol Π is correct, (τ, ε_1)-client-private and ε_2-simulatable for $\boldsymbol{\alpha} \in \mathcal{S}$ and the CDS protocol for the set $\mathcal{S}$ is ε_3-simulatable, then Protocol 5 is correct and $(\tau, \varepsilon_1; \max\{\varepsilon_2, \varepsilon_3\})$-relaxed-secure.*

Proof. Due to the re-randomisation, the recovered replies $\hat{d}_i$ have the same distribution as d_i, thus correctness is evident. Client-privacy is evident as both protocols share the query $\mathfrak{q}$. For server-privacy, note that if $\boldsymbol{\alpha} \in \mathcal{S}$, we can first use the original simulator to simulate d_i and then apply the CDS transformation to the simulation output. The corresponding simulation is ε_2-close to the real run, since the original reply is not more than ε_2 away from the simulated one. Otherwise, $(e_1, \ldots, e_n)$ are random ciphertexts and thus perfectly simulatable. Now if we add a simulated CDS reply $\hat{\mathfrak{a}}_{\mathrm{cds}}$, then the aggregated reply $\hat{\mathfrak{a}}_{\mathrm{cds}}, e_1, \ldots, e_n$ is ε_3-close to the real protocol transcript, as the CDS is ε_3-simulatable. The claim follows, as $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{q})$ is either in $\mathcal{S}$ or not. □

Optimisations. If all replied ciphertexts of the original protocol are in the fixed range, i.e., $\mathsf{Dec}_{\mathsf{sk}}(d_i) \in \{0,1\}^{\ell}$ then full recovery of t_i is not necessary. It is sufficient to send $t_i \mod 2^{\ell}$ together with a extra bit needed to indicate a possible wrapping $t_i \geq N - 2^{\ell}$ and the message expansion rate can be less than $\mathcal{L}(\Psi)$. Secondly, note that the communication overhead of the CDS transformation is linear in $|\mathfrak{a}|$. Therefore, the transformation is quite inefficient when $|\mathfrak{a}|$ is long. To get better performance, the server can use symmetric encryption to garble the original reply and a CDS protocol to release the corresponding key. The output is still computationally simulatable and thus we achieve computational server-privacy. A block cipher in counter mode is the best encryption method, as then the client can efficiently decrypt only necessary parts of $\mathfrak{a}$.

6 Practical Applications of Crypto-computing Techniques

In this section, we show how to use additively homomorphic two-message protocols to solve several important cryptographic tasks. Here, the query $\mathfrak{q}$ is a vector of ciphertexts, and the reply is computed by combining the identities (1) and (2) with Prot. 4. Note that the outputs of crypto-computed sums and products are perfectly simulatable provided that the end result is re-randomised. Consequently, client-privacy follows form (τ, ε_1)-IND-CPA security and server-privacy follows from the basic properties of forked composition, see Thm. 1 and 2. Shortly put, the resulting protocols are $(\tau - \mathcal{O}(1), m\varepsilon_1; n\varepsilon_2)$-relaxed-secure, where m is the number of ciphertexts and n is the number of DIE instances, provided that the basic DIE protocol is ε_2-simulatable.

Sometimes we must also prove that knowledge of $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is equivalent to the knowledge of $f_1(\boldsymbol{\alpha}, \boldsymbol{\beta}), \ldots, f_s(\boldsymbol{\alpha}, \boldsymbol{\beta})$, i.e., design a protocol for f based on generic operations. As for 1024-bit Paillier and 2^{-80}-server-privacy, we can transfer 393 bits in the individual DIE reply whenever the number of DIE instances is less than 2^{40}, the resulting protocols are really efficient.

Oblivious transfer. Recall that a 1-*out-of-n oblivious transfer* (OT) protocol implements an ideal functionality $f(\alpha; \beta_1, \ldots, \beta_n) = \beta_\alpha$ if $\alpha \in \{1, \ldots, n\}$ and $\perp$ otherwise. Already in [AIR01], the authors showed that such a protocol can be expressed as a forked composition of n individual DIE protocols:

- release β_1 if $\alpha = 1$,

  $\vdots$
- release β_n if $\alpha = n$.

Therefore, we get a relaxed-secure implementation of oblivious transfer by using Prot. 4 to implement all instances of DIE protocols. Moreover, a client can use any CPIR protocol to obliviously choose the αth reply of the DIE. Hence, we have just described a generic transformation from any CPIR to a relaxed-secure oblivious transfer.

An alternative approach was taken by Chang [Cha04] who proved that the basic DIE protocol from [AIR01] leaks at most $\beta_{\alpha_1} \mod p_1$ and $\beta_{\alpha_2} \mod p_2$ whenever the plaintext order is a product of two primes p_1 and p_2. In the corresponding 1-out-of-n OT protocol an honest client has to encrypt values that depend on the secret key and thus the client-privacy does not follow directly from IND-CPA security.

Millionaire's protocol with logarithmic communication. The millionaire's problem is: given client's private input α and server's private input x, decide whether $\alpha > x$. Although numerous solutions have been proposed for this problem, none of the proposals is completely satisfactory. For example, the two-message protocol of Blake and Kolesnikov [BK04] is server-secure only in the semihonest model since encrypted inputs must be in correct range, it can leak information otherwise. To solve that type of problems, consider a circuit CDS protocol for a public set $\mathcal{S}_x = \{\alpha \in \{0,1\}^m : \alpha > x\}$. Writing α bit by bit $(\alpha_{m-1}, \dots, \alpha_0)$, we obtain

$$\begin{aligned}\Psi_{\mathcal{S}_x}(\alpha) =& ([\alpha_{m-1} \overset{?}{=} 1] \wedge [x_{m-1} \overset{?}{=} 0]) \vee \\ & ([\alpha_{m-1} \overset{?}{=} x_{m-1}] \wedge [\alpha_{m-2} \overset{?}{=} 1] \wedge [x_{m-2} \overset{?}{=} 0]) \vee \\ & ([\alpha_{m-1} \overset{?}{=} x_{m-1}] \wedge [\alpha_{m-2} \overset{?}{=} x_{m-2}] \wedge [\alpha_{m-3} \overset{?}{=} 1] \wedge [x_{m-3} \overset{?}{=} 0]) \vee \cdots \vee \\ & ([\alpha_{m-1} \overset{?}{=} x_{m-1}] \wedge [\alpha_{m-2} \overset{?}{=} x_{m-2}] \wedge \cdots \wedge [\alpha_1 \overset{?}{=} x_1] \wedge [\alpha_0 \overset{?}{=} 1] \wedge [x_0 \overset{?}{=} 0]) \ .\end{aligned}$$

Here, every row corresponds to one conjunctive affine equality test. Fig. 1 depicts the corresponding unoptimised circuit. Now consider the modified protocol where β_0 is a publicly fixed ℓ-bit secret and the server randomly reorders the leaf CDS replies $\mathfrak{a}_1, \dots, \mathfrak{a}_m$. Finally, the client outputs 1 if one of the recovered CDS outputs is β_0. As the formula $\Psi_{\mathcal{S}_x}(\alpha)$ is a disjunction of affine zero tests, then in the ideal world, the client learns a randomly shuffled set $\{\beta_0, \bot, \dots, \bot\}$ if $\alpha > x$ and $\{\bot, \bot, \dots, \bot\}$ otherwise. Hence, the modified protocol is server-private even if x is private and we have obtained a relaxed-secure solution to the millionaire problem that fails with probability $2^{-\ell}$. The total communication of our solution is $2m$ ciphertexts, the client's computation is $\Theta(m)$ and the server's computation is $\Theta(m^2)$, and we only assume that the underlying additively homomorphic cryptosystem is IND-CPA secure. The server's workload can be reduced $\Theta(m)$ as in the Blake-Kolesnikov protocol, if we first crypto-compute a recursion $t_i = (\alpha_i - x_i)r_i + \cdots + (\alpha_{m-1} - x_{m-1})r_{m-1}$ for $r_i \leftarrow \mathbb{Z}_N$ and then re-randomise it by crypto-computing $u_i = t_i s_i$ for $s_i \leftarrow \mathbb{Z}_N$.

Interestingly enough, one can view our solution as an efficient generalisation of the Fischlin protocol [Fis01]. The latter can be alternatively described as a CDS protocol based on additively homomorphic cryptosystem over $\mathbb{Z}_2$. Due to the small message space, the Fischlin's protocol requires a parallel run of ℓ protocols to achieve the same reliability as our protocol, i.e., our protocol is ℓ times more efficient.

Conditional OT. In a *conditional oblivious transfer* protocol for public predicate Ψ, the client has a private input α and the server has a private input (x, β_0, β_1). The client obtains β_1 if $\Psi(\alpha, x) = 1$ and β_0 otherwise. Assume that the master secret β is reconstructed identically for the circuits without witnesses Ψ and $\neg\Psi$ and the reconstruction process and the number of true leaves leaks nothing about x except $\Psi(\alpha, x)$. In particular, assume that the master secret can be reconstructed from randomly shuffled shares. Let $\mathcal{B}_{\Psi(\alpha,x)}$ and $\mathcal{B}_{\neg\Psi(\alpha,x)}$ be the shuffled CDS replies in the ideal world. Then given a shuffled set of sets $\{\mathcal{B}_{\Psi(\alpha,x)}, \mathcal{B}_{\neg\Psi(\alpha,x)}\}$, one can learn only $\beta_{\Psi(\alpha,x)}$ and nothing more, provided that the number of leaf tests is equal $|\mathcal{B}_{\Psi(\alpha,x)}| = |\mathcal{B}_{\neg\Psi(\alpha,x)}|$.

This leads to the following COT protocol. First, the server assigns β_0 to $\neg\Psi$ and β_1 to Ψ and adds trailing zeroes to leaf secrets of one circuit and trailing ones to the remaining sub-secrets. Next, the server constructs replies for each leaf CDS and sends randomly shuffled replies back. Finally, the client restores sets $\mathcal{B}_{\Psi}(\alpha, x)$ and $\mathcal{B}_{\neg\Psi(\alpha,x)}$ and reconstructs $\beta_{\Psi(\alpha,x)}$. The failure probability is bounded by $2^{-k} \cdot \mathcal{L}(\Psi)$ where k is the number of trailing zeroes and ones. Since $[\alpha > x]$ and $[\alpha \leq x]$ have such symmetrical circuits, we can construct a COT protocol for $[\alpha > x]$ and for many other relations.

Electronic voting and auctions without random oracles. E-voting and auction protocols based on homomorphic encryption [CGS97, DJ01, LAN02] are natural extensions of homomorphic two-message protocols, since the secret key is known by the election tallier (or a coalition of talliers) to whom the server forwards the second message. In such protocols, conditional disclosure of secrets can be used to guarantee security of the election authority against malicious voters and a semihonest server. As in [BGN05], consider an electronic voting protocol where every voter sends an encryption $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(v_i)$ to talliers. We assume that the protocol is secure if $v_i \in \mathcal{S}$ for some publicly known set $\mathcal{S}$; this is true in typical e-voting protocols [CGS97, DJ01].

In the existing protocols, it is usually assumed that every voter accompanies his or her vote with a non-interactive zero-knowledge proof that $v_i \in \mathcal{S}$. Instead, the talliers can jointly apply the CDS protocol, with output secret 0, to c_i (this can be done very efficiently if $\mathcal{S}$ is the set of powers of a fixed integer) and then threshold-decrypt the result. If the plaintext is equal to 0, talliers accept the vote as correct. Of course, every step of the talliers has to be accompanied by a zero-knowledge proof of correctness (to each other and to every possible outside observer), but since the number of talliers is significantly smaller than the number of voters, this is doable in practise, see [BGN05].

As the result, we get a voter-private, universally verifiable and robust e-voting scheme where the voters only have to perform one encryption, assuming only that there exists an IND-CPA secure additively homomorphic public-key cryptosystem. The same trick can be used to eliminate the need for random oracles in a similar electronic auction scheme of [LAN02] and in many other similar protocols. Compared to the protocols of [BGN05], our protocols are more efficient since they are based on genuine additive homomorphic cryptosystem whereas [BGN05] uses a lifted version of ElGamal and thus there one has to compute discrete logarithms. Moreover, their cryptosystem is secure under less established security assumptions.

Multiplicative relations and polynomial arithmetic. Finally, we illustrate the power of using auxiliary witnesses. It is well known that multiplicative relation $[z \overset{?}{=} xy]$ does not have a compact NNF. However, we can still construct efficient circuit CDS protocol by introducing a suitable witness $\boldsymbol{w}$. Let $x, y \in \{0,1\}^m$ and $z \in \{0,1\}^{2m}$ be sent to the server by individually encrypting each bit of x, y, z and let $w_0, \ldots, w_{m-1}$ be auxiliary variables such that $w_i = xy_i$. Then $xy = w_0 + 2w_1 + \cdots + 2^{m-1}w_{m-1}$ and the formula $\Psi_{[z=xy]}$ can be expressed as a conjunction of tests: (1) $x_{m-1}, \ldots, x_0 \in \{0,1\}$, (2) $[y_i \overset{?}{=} 0] \wedge [w_i \overset{?}{=} 0] \vee [y_i \overset{?}{=} 1] \wedge [w_i \overset{?}{=} x]$ for $i \in \{0, \ldots, m-1\}$ and x is crypto-computed as $x_0 + \cdots + 2^{m-1}x_{m-1}$, and (3) $[z \overset{?}{=} w_0 + \cdots + 2^{m-1}w_{m-1}]$.

Several papers, see e.g. [KS05], use additively homomorphic two-message protocols in a setting where one encrypts the coefficients of some polynomials, where the important quantity is the set of roots of this polynomial. For example, if F_1 is the set of roots of $f_1(x)$ and F_2 is the set of roots of $f_2(x)$ then $F_1 \cup F_2$ is the set of roots of $f_1(x) \cdot f_2(x)$. Consequently, we can also construct a CDS protocol for the set to prove that $g(x) = f_1(x) \cdot f_2(x)$, as the ith coefficient $g_i = f_{10}f_{2i} + \cdots + f_{1i}f_{20}$. Now, we can also verify that for some sets F_1, F_2 and G, it holds that $F_1 \cup F_2 = G$.

7 Theoretical Implications

Although we stated our results in the PKI model, where a trusted key generator generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}$ and privately transfers $(\mathsf{sk}, \mathsf{pk})$ to the client and pk

to the server, they can be easily implemented in the standard model. Namely, we can eliminate the PKI assumption if the client executes once, separately and in an isolated manner (that is, no other messages of different protocols are sent by the client at the same time), with every server a zero-knowledge proof of knowledge that pk is valid and that he knows the corresponding secret key. This is followed by the real protocol. In the security proof, the simulator extracts the secret key by rewinding and thereafter continues to work as previously. Since we require statistical server-security—and thus can use an unbounded simulator—then it is actually sufficient to have a zero-knowledge proof that the key is correct: the simulator just computes the secret key corresponding to the (correct) public key. It is even irrelevant whether the client computes the public key with a correct distribution, since for the proof we only need the existence of the secret key. Therefore, the amortised message complexity is still two-messages in the standard model, as the verification of a public key must be carried out only once.

It is well known that secure two-party protocols require at least three messages, therefore, it is impossible to obtain full security of two-message protocols in the malicious model. In fact, one cannot achieve more than relaxed-security in two messages even in the PKI model. Consequently, the CDS-transformation presented in Sect. 5 is a universal round-optimal transformation from semihonest model to relaxed-secure model whenever the first message contains only ciphertexts. Moreover, computational and communication resources are linear in the size of the circuit that is needed to test a validity of an input. More formally, assume that for sets $\mathcal{S}_m$ of m-bit strings exists a polynomial-size formula $\Psi(\boldsymbol{\alpha}, \boldsymbol{w})$ such that $\boldsymbol{\alpha} \in \mathcal{S}_m$ iff $\exists \boldsymbol{w} : \Psi(\boldsymbol{\alpha}, \boldsymbol{w}) = 1$. Then there exists also a polynomial-size formula $\overline{\Psi}(\boldsymbol{\alpha}, \overline{\boldsymbol{w}})$ in a negation normal form such that $\boldsymbol{\alpha} \in \mathcal{S}_m$ iff $\exists \overline{\boldsymbol{w}} : \overline{\Psi}(\boldsymbol{\alpha}, \overline{\boldsymbol{w}}) = 1$. Therefore, there exist a family of polynomial-time CDS protocols for an arbitrary set $\mathcal{S}$ in $\mathbf{NP/poly}$. Such protocols can be automatically generated in polynomial time for every set $\mathcal{S}$ that can be described by any $\mathbf{NP}$ relation.

Alternative classical round-preserving methods that guard against malicious clients are based on non-interactive zero-knowledge proofs, i.e., we have to either rely on random oracles or use the *common reference string* (CRS) model. While CRS is a plausible model for protocol design, constructing efficient non-interactive zero-knowledge protocols for $\mathbf{NP}$ in the CRS model has been a long-standing open problem. Thus, our result is also appealing from the complexity-theoretical viewpoint.

As stated already in Sect. 6, the DIE-based OT protocol leads to a general transformation from CPIR to information-theoretically server-private OT, as the client can use the CPIR protocol to fetch only the answer of the αth DIE protocol. In particular, there exists a generic CPIR construction for any IND-CPA secure additively homomorphic cryptosystem [Ste98] with sublinear-but-superpolylogarithmic communication. Therefore, there exists also an OT protocol with comparable communication under the sole assumption that IND-CPA secure additively homomorphic cryptosystems exists. Under the assumption that IND-CPA secure length-flexible additively homomorphic cryptosystem exist, one can construct a CPIR protocol [Lip05] with communication $\Theta(\mathbf{k} \cdot \log^2 n + \ell \cdot \log n)$ where $\mathbf{k}$ is the security parameter. Consequently, we can construct an OT with communication $\Theta(\mathbf{k} \cdot \log^2 n + \ell \cdot \log n)$, if an IND-CPA secure length-flexible additively homomorphic cryptosystem exists. Finally due to the results of Gentry and Ramzan [GR05], there also exists an OT protocol with optimal communication $\Theta(\log n + \ell + \mathbf{k})$, if we assume that Φ-Hiding is hard and that an IND-CPA secure additively homomorphic cryptosystem exists.

Another two-message OT protocol was proposed by Kalai [Kal05]. Her protocol is secure in the standard model, whereas our protocol requires a zero-knowledge proof that the public key is valid. On the other hand, the query of Kalai's protocol does not consist of ciphertexts and thus cannot be used for the CDS protocol. Moreover, Thm. 3 holds even with incorrectly formed pk provided that the corresponding encryption rule is additively homomorphic and it is still possible to detect invalid ciphertexts. Therefore, we can omit the zero-knowledge proofs for pk provided that we can verify that the plaintext order does not have too small factors. For small enough γ and public plaintext order this can be done efficiently by using Lenstra's Elliptic Curve Method, see App. A for further details. Hence, it is possible to achieve two messages as non-amortised round-complexity in the standard model under stronger computational assumptions.

Finally, note that small detectable factors of N can be effectively eliminated. Namely, a server can eliminate a known factor p by multiplying a ciphertext $\mathsf{Enc}_{\mathsf{pk}}(x)$ with $\mathsf{Enc}_{\mathsf{pk}}(pr)$ for $r \leftarrow \mathbb{Z}_N$. Then the client can learn only a coset $x + p\mathbb{Z}_N$, i.e., we have established a new cryptosystem over a new message space $\mathbb{Z}_N / p\mathbb{Z}_n \simeq \mathbb{Z}_{N/p}$.

Acknowledgements. We would like to thank Phil Carmody, Yuval Ishai and Vladimir Kolesnikov for useful comments. The work was partially supported by the Finnish Academy of Sciences and by the Estonian Science Foundation, grant 6848.

References

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, 2001. Springer-Verlag.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *The Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, 2005. Springer Verlag.

[BK04] Ian F. Blake and Vladimir Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals. In *Advances on Cryptology — ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, 2004. Springer-Verlag.

[BL88] Josh Benaloh and Jerry Leichter. Generalized Secret Sharing and Monotone Functions. In *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, 1988. Springer-Verlag, 1990.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, 1997. Springer-Verlag.

[Cha04] Yan-Cheng Chang. Single Database Private Information Retrieval with Logarithmic Communication. In *The 9th Australasian Conference on Information Security and Privacy (ACISP 2004)*, volume 3108 of *Lecture Notes in Computer Science*, 2004. Springer-Verlag.

[DJ01] Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography 2001*, volume 1992 of *Lecture Notes in Computer Science*, 2001. Springer-Verlag.

[Elg85] Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4), 1985.

[Fis01] Marc Fischlin. A Cost-Effective Pay-Per-Multiplication Comparison Method for Millionaires. In *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001*, volume 2020 of *Lecture Notes in Computer Science*, 2001. Springer-Verlag.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient Private Matching and Set Intersection. In *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, 2004. Springer-Verlag.

[GIKM00] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. *Journal of Computer and System Sciences*, 60(3), June 2000.

[GLLM04] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On Private Scalar Product Computation for Privacy-Preserving Data Mining. In *Information Security and Cryptology - ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, 2004. Springer-Verlag.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, 1982. ACM.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In *The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, 2005. Springer-Verlag.

[Kal05] Yael Tauman Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. In *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, 2005. Springer-Verlag.

[KS05] Lea Kissner and Dawn Song. Privacy-Preserving Set Operations. In *Advances in Cryptology — CRYPTO 2005, 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, 2005. Springer-Verlag.

[LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In *Financial Cryptography — Sixth International Conference*, volume 2357 of *Lecture Notes in Computer Science*, 2002. Springer-Verlag.

[Len87] Hendrik W. Lenstra, Jr. Factoring integers with Elliptic Curves. *Annals of Mathematics*, 126(2), 1987.

[Lip05] Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In *The 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, 2005. Springer-Verlag.

[LLM05] Sven Laur, Helger Lipmaa, and Tanel Mielikäinen. Private Itemset Support Counting. In *Information and Communications Security, 7th International Conference, ICICS 2005*, volume 3783 of *Lecture Notes in Computer Science*, 2005. Springer-Verlag.

[NP99] Moni Naor and Benny Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, 1999. ACM Press.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, 1999. Springer-Verlag.

[Ste98] Julien P. Stern. A New and Efficient All or Nothing Disclosure of Secrets Protocol. In *Advances on Cryptology — ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, 1998. Springer-Verlag.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-Interactive CryptoComputing For NC^1. In *40th Annual Symposium on Foundations of Computer Science*, 1999. IEEE Computer Society.

[WY04] Rebecca N. Wright and Zhiqiang Yang. Privacy-Preserving Bayesian Network Structure Computation on Distributed Heterogeneous Data. In *Proceedings of The Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004. ACM.

[ZD06] Paul Zimmermann and Bruce Dodson. 20 Years of ECM. In *The Algorithmic Number Theory Symposium*, volume 4076 of *Lecture Notes in Computer Science*, 2006. Springer-Verlag.

[Zim06b] Paul Zimmermann. Optimal Parameters for ECM. Available at `http://www.loria.fr/~zimmerma/records/ecm/params.html`, as of May, 2006.

A Non-interactive Partial Public Key Validation

Next, we propose another technique to transform the proposed protocols to be secure in the the standard model. It does not need extra messages but needs an extra amount of computations by an honest server. Namely, Thm. 3 holds even with incorrectly formed pk provided that the corresponding encryption rule is additively homomorphic and it is still possible to detect invalid ciphertexts. In particular, the Paillier cryptosystem is homomorphic even if a public modulus N is incorrectly formed. Thus, the verification of pk can just consist of computing a lower bound γ on factors of N. For small enough γ this can be done efficiently by using Lenstra's Elliptic Curve Method [Len87] which works in time $\exp((\sqrt{2}+o(1))\sqrt{\ln p \cdot \ln\ln p})$ where p is the smallest factor of N [ZD06]. If we want the server's computation to be polynomial in $\log N$ then we have to take a sufficiently small ℓ. To provide some concrete numbers note that ECM allows "efficient" detection of 88-bit factors. Assume that the desired server-privacy level is 2^{-40}. Such a choice of ε_2 is most probably sufficient in practise. Then, in the case of the DIE protocol, one has $\ell = 47$, which is sufficient for several applications. In Spring 2006, we verified this approach by using the suggested optimal parameters from [Zim06b], on an AMD Athlon 64 3000+ processor by using the GMP-ECM software. As an example, if $N = pq$, where p is an 88-bit prime and q is an $(1024-88)$-bit prime then one has to run the ECM algorithm on an expected 206 curves with bounds $B1 = 50\,000$ and $B2 = 5\,000\,000$. Testing on one curve with these parameters takes approximately 2.5 seconds, and thus testing that the smallest factor is greater than 2^{89} takes 9 minutes on average. On the other hand, if q is an 66-bit prime then it takes an expected 77 curves with bounds $B1 = 11\,000$ and $B2 = 1\,100\,000$. On the same platform, testing one curve with these parameters takes approximately 0.66 seconds and checking the bound 2^{67} takes 51 seconds on average. Given the advances in the ECM, we would expect the quoted timings to decrease dramatically over the next few years.

An Unconditionally Secure Protocol for Multi-Party Set Intersection[⋆]

Ronghua Li[1,2] and Chuankun Wu[1]

[1] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, P.R. China
[2] Graduate School of Chinese Academy of Sciences, Beijing 100039, P.R. China
{lirh,ckwu}@is.iscas.ac.cn

Abstract. Existing protocols for private set intersection are based on homomorphic public-key encryption and the technique of representing sets as polynomials in the cryptographic model. Based on the ideas of these protocols and the two-dimensional verifiable secret sharing scheme, we propose a protocol for private set intersection in the information-theoretic model. By representing the sets as polynomials, the set intersection problem is converted into the task of computing the common roots of the polynomials. By sharing the coefficients of the polynomials among parties, the common roots can be computed out using the shares. As long as more than $2n/3$ parties are semi-honest, our protocol correctly computes the intersection of n sets, and reveals no other information than what is implied by the intersection and the secrets sets controlled by the active adversary. This is the first specific protocol for private set intersection in the information-theoretic model as far as we know.

Keywords: Secure multi-party computation, privacy-preserving set intersection, unconditional security.

1 Introduction

This paper studies the following problem: n parties each with a secret set want to compute the intersection of these sets without leaking anything else about the secret sets. This problem is a specific case of secure multi-party computation, which is introduced by Yao [10], and extended by Goldreich, Micali and Wigderson [5]. The goal of secure multi-party computation is to design a protocol for the parties each with a secret input to compute securely a public function of their inputs. The protocol should be correct and private. Correctness means that every party believes the correctness of the output. Privacy means that no party can learn more than what is implied by its output and its own input. For the set intersection problem, the inputs are the parties' secret sets and the output is the intersection of these sets (all the parties get the intersection and nothing

[⋆] Supported by National Natural Science Foundation of China under Grant NO.90304007 and the National Grand Fundamental Research 973 Program of China under Grant NO.2004CB318004.

about other parties' secret sets). Protocols for private set intersection are useful in online recommendation services, online dating services, medical databases, data mining, etc., as pointed out by Freedman, Nissim and Pinkas [4].

Related work. The set intersection problem is studied in [4,7,8], where several protocols are presented in the cryptographic model based on homomorphic public-key encryption. In the information-theoretic model, many general secure computation protocols are proposed (e.g. [3,1,6,2]). Solutions to the set intersection problem can be derived from these general protocols from the theoretical point of view, however, how to modify the general protocols to solve the set intersection problem is not known.

Our work. This paper presents a protocol for the set intersection problem in the information-theoretic model. The protocol uses the idea of representing a set as a polynomial [4,7,8] and the two-dimensional verifiable secret sharing used in the general unconditionally secure protocols (e.g. [3,2]). Our protocol follows the share-compute-recover paradigm for general constructions in the information-theoretic model. By representing the parties' secret sets as polynomials, the set intersection problem is converted into the task of computing the common roots of these polynomials. Then the coefficients of the polynomials are shared using two-dimensional secret sharing scheme and the common roots of the polynomials are computed out using the shares. This is the first specific protocol for the intersection problem in the information-theoretic model as far as we know.

Assuming that an active adversary corrupts less than $n/3$ parties, the proposed protocol enables n parties each with a secret set to compute privately the intersection of these sets and leaks no other information than what is implied by the intersection and the secret sets controlled by the adversary. The security of the protocol is proved and efficiency is analyzed in Section 4.

The proposed protocol is efficient in terms of communication complexity. The protocol demands 6 rounds of communication and exchanges $\mathcal{O}(n^4k^2)$ elements in E, where k is the size of the secret sets and E is a large finite field. The most efficient general protocol in the information-theoretic model is due to [2]. The protocol uses circuit randomization technique (for details, see[1]) and consists of preparation phase, input phase, and computation phase. If the protocol is applied to the set intersection problem, then at least nk random numbers should be shared in the pre-processing phase, which needs $\mathcal{O}(n^4k)$ elements to be exchanged. As no article considers applying the general unconditionally secure protocols to the set intersection problem before, so we cannot estimate the total communication complexity, thus only a partial comparison is provided here.

2 Preliminaries

2.1 Adversary and Communication Models

Adversary model. A cheating party in a protocol is modelled as an adversary who can corrupt parties. There are two kinds of adversaries. A passive adversary learns the information hold by the corrupted parties, but the corrupted parties still act correctly according to the protocol. An active adversary not only learns the information hold by the corrupted parties but also takes full control of them. The two models are called passive model and active model respectively.

Communication model. There are two communication models in secure computation: cryptographic model and information-theoretic model. The first model assumes that parties are connected by public channels, and parties have bounded computation power (probabilistic polynomial time) and the adversary can see all the messages exchanged among the parties. The second model assumes that there are pair-wise secure channels among parties, and parties have unbounded computation power and the adversary can not learn the messages exchanged between the honest (i.e. uncorrupted) parties.

In the paper, we assume both passive and active adversaries, and we assume the information-theoretic model.

2.2 Polynomial Representation of Sets

Notation. Let E denote a large finite field. $E[x]$ consists of all the polynomials whose coefficients are chosen from E, and the probability that a random polynomial (i.e. the coefficients are chosen randomly from E) in $E[x]$ represents elements of U is negligible.

We use the technique of representing sets as polynomials [4]. Let S be a set of size k. A polynomial of degree k can be constructed: $f(x) = a_0 + a_1x + ... + a_kx^k$, $f(a) = 0$ if and only if $a \in S$, where the coefficients of $f(x)$ are chosen from E.

In [7,8], Kissner and Song convert the set intersection problem into the task of computing the roots of a polynomial by use of the polynomial representation of sets. This paper follows the idea of the protocols for set intersection in [7,8]. We briefly review their idea as follows.

Let $r(x)$ be a random polynomial in $E[x]$. If a is the root of $f(x)$ then a is the root of $f(x)r(x)$ too, that is, $f(x)r(x)$ preservers all the roots of $f(x)$. Let $S_1, ...S_n$ be n sets, and the polynomials $f_1(x), ..., f_n(x)$ represent $S_1, ...S_n$ respectively. Let $r_1(x), ..., r_n(x)$ be n random polynomials in $E[x]$. Then the roots of $f_1(x)r_1(x) + ... + f_n(x)r_n(x)$ represent the intersection of $S_1 \cap ... \cap S_n$.

2.3 Two-Dimensional Verifiable Secret Sharing

Secret sharing is firstly introduced by Shamir [9]. A secret s can be shared with a polynomial of degree t: $f(x)$ such that $f(0) = s$. Let $\alpha_1, ..., \alpha_n$ be public parameters chosen in E. P_i's share is $Piece_i(s) = f(\alpha_i)$, $i = 1, ..., n$. Shamir's secret sharing scheme is extended to two-dimensional secret sharing [3,6]. Each secret value is shared among the players with a polynomial of degree t, and each share is again shared among the parties with a polynomial of degree t. Let $Piece_i$ denote a share, and $Piece_{ij}$ (sometimes written as $Piece_{i,j}$, e.g. $Piece_{i,2k}(s)$) denote a share-share.

Sharing. To share a secret s, a party chooses a random two-dimensional polynomial $p(x, y) = \sum_{i,j=0}^{t} r_{ij}x^iy^j$ such that $p(0, 0) = s$, and sends to P_i the two polynomials $f_i(x) = p(x, \alpha_i), \tilde{f}_i(y) = p(\alpha_i, y)$, $i = 1, ..., n$. P_i's shares are $Piece_i(s) = f_i(0)$, $Piece_{ji}(s) = \tilde{f}_i(\alpha_j) = p(\alpha_i, \alpha_j)$, $j = 1, ..., n$. We say that the party t-shares s or s is t-shared.

Verifying the correctness of sharing. For $i = 1$, ..., n, each party P_j $(j \neq i)$ sends to P_i the share-share $Piece_{ij}(s) = f_j(\alpha_i)$, and P_i checks if the received value is equal to $\tilde{f}_i(\alpha_j)$.

Linear function. Let a, b be two secrets shared among the parties, and r be a constant integer, parties can compute locally the sharing of the new secrets ra and $a + b$.

Multiplication of two secrets. Let a, b be two secrets shared among the parties, the goal is to produce the t-sharing of a new secret $c = ab$ among the parties. The computation procedure develops in two steps: local computation and degree reduction.

The degree reduction phase involves mainly a sub-protocol: re-share protocol. The re-share protocol allows parties to produce t-shares of a secret given its t'-shares. In the active model, when a party t-shares his share $Piece_i(s)$, he proves that the shared value is indeed $Piece_i(s)$. The details for re-sharing protocol can be found in [6].

Reconstruction. Let s be t-shared among the parties. To reveal the secret s to a designated party, all other parties send their shares of s to the designated party, then the party recovers the secret using Lagrange Interpolation.

Error correction. In the active model, a party may receive wrong shares from corrupted parties when reconstructing a secret. Parties can perform locally error correction to get n correct shares, and recover the secret (see [3] for details).

3 The Proposed Protocol for Private Set Intersection

A protocol in the passive model is constructed first. The protocol in the active model is constructed based on the protocol in the passive model. The two protocols develop similarly except that some verifications and proofs are added to the second protocol.

3.1 Construction in the Passive Model

Let $P_1, ..., P_n$ be n parties connected with pair-wise secure channels. Assume parties are computationally unbounded and a passive adversary is allowed to corrupt $t < n/2$ parties. $P_1, ..., P_n$ have secret sets $S_1, ..., S_n$ respectively. Let k be the size of the sets and $S[j]$ denote the j-th element of S. Then a set S can be represented as a polynomial of degree k in $E[x]$: $f(x) = (x - S[1])...(x - S[k])$=$a_0 + a_1x + ... + a_kx^k$.

The main idea. Each party represents his secret set as a polynomial of k degree. Let $f_1(x) = a_{10} + a_{11}x + ... + a_{1k}x^k$, $f_2(x) = a_{20} + a_{21}x + ... + a_{2k}x^k$,

..., $f_n(x) = a_{n0} + a_{n1}x + ... + a_{nk}x^k$ be the n parties' polynomials respectively. Let $r_1(x)$, ..., $r_n(x)$ be n random polynomials of degree k. If the coefficients of $f_i(x), r_i(x)$, $i = 1, ..., n$, are shared, then the shares of the coefficients of the polynomial $F(x) = f_1(x)r_1(x) + f_2(x)r_2(x) + ... + f_n(x)r_n(x)$ can be computed. Each party publishes his shares of the coefficients of $F(x)$, then every party can recover the polynomial $F(x)$ and evaluate it at each element of his secret set. If the function value equals to zero then the element belongs to the intersection, else the element does not belong to the intersection. Thus the parties compute out the intersection of the sets. The protocol is composed of three phases: input phase, computation phase and output phase.

Input Phase

Step 1. Each party represents his secret set as a polynomial and shares the coefficients of the polynomial among the parties.

- For $i = 1, ..., n$, P_i represents his secret set as a polynomial $f_i(x)$.
- For $i = 1, ..., n$, P_i t-shares the coefficients of his polynomial (e.g. P_i uses $f_{ij}(x)$ to share a_{ij}, $j = 0, ..., k$).

Step 2. The parties produce jointly the t-sharing of n random polynomials of degree k.

- For $i = 1, ..., n$, P_i produces n random polynomials of degree k: $r_{i1}(x) = b_{i10} + b_{i11}x + ... + b_{i1k}x^k$, $r_{i2}(x) = b_{i20} + b_{i21}x + ... + b_{i2k}x^k$, ..., $r_{in}(x) = b_{in0} + b_{in1}x + ... + b_{ink}x^k$.
- For $i = 1, ..., n$, P_i t-shares the coefficients of the n random polynomials among the parties. E.g. P_i uses $r_{ij0}(x)$, $r_{ij1}(x)$, ..., $r_{ijk}(x)$ to share b_{ij0}, b_{ij1},..., b_{ijk} respectively, $j = 1, ..., n$.
- For $i = 1, ..., n$, P_i computes the t-shares of the following n polynomials of degree k $r_j(x) = \sum_{j'=1}^{n} r_{j'1}(x) = w_{j0} + w_{j1}x + ... + w_{jk}x^k$, $j = 1, ..., n$, as below: $Piece_i(w_{j0}) = \sum_{j'=1}^{n} Piece_i(b_{j'10})$, $Piece_i(w_{j1}) = \sum_{j'=1}^{n} Piece_i(b_{j'11})$, ..., $Piece_i(w_{jk}) = \sum_{j'=1}^{n} Piece_i(b_{j'1k})$, $j = 1, ..., n$.

Computation Phase

Let $f_j(x)r_j(x) = z_{j0} + z_{j1}x + ... + z_{j,2k}x^{2k}$, $j = 1$, ..., n, and $F(x) = f_1(x)r_1(x)$ + $f_2(x)r_2(x)$ +... + $f_n(x)r_n(x)$= $z_0 + z_1x + ... + z_{2k}x^{2k}$.

Step 1. Compute the $2t$-shares of the coefficients of $f_1(x)r_1(x)$, $f_2(x)r_2(x)$, ..., $f_n(x)r_n(x)$.

- For $i = 1, ..., n$, P_i computes locally the following values: $Piece_i(a_{j0})Piece_i(w_{j0})$, $Piece_i(a_{j0})Piece_i(w_{j1})$, ..., $Piece_i(a_{jk})Piece_i(w_{jk})$, $j = 1, ..., n$.

Step 2. Call the re-sharing protocol, and convert the $2t$-shares of the coefficients of $f_1(x)r_1(x)$, $f_2(x)r_2(x)$, ..., $f_n(x)r_n(x)$ into t-sharing.

- For $i = 1, ..., n$, P_i t-shares the following values: $Piece_i(a_{j0})Piece_i(w_{j0})$, $Piece_i(a_{j0})Piece_i(w_{j1})$, ..., $Piece_i(a_{jk})Piece_i(w_{jk})$, $j = 1, ..., n$.

- For $i = 1, ..., n$, P_i reconstructs the t-shares of $a_{j0}w_{j0}$, ..., $a_{jk}w_{jk}$: $Piece_i(a_{j0}w_{j0})$, $Piece_i(a_{j0}w_{j1})$, ..., $Piece_i(a_{jk}w_{jk})$, $j = 1, ..., n$.
- For $i = 1, ..., n$, P_i computes the t-shares of the coefficients of $f_1(x)r_1(x)$, $f_2(x)r_2(x)$, ..., $f_n(x)r_n(x)$ as below: $Piece_i(z_{j0}) = Piece_i(a_{j0}w_{j0})$, $Piece_i(z_{j1}) = Piece_i(a_{j0}w_{j1}) + Piece_i(a_{j1}w_{j0})$, ..., $Piece_i(z_{j,2k}) = Piece_i(a_{jk}w_{jk})$, $j = 1, ..., n$.

Step 3. Parties compute the t-shares of the $2k + 1$ coefficients of $F(x)$.

- For $i = 1, ..., n$, P_i computes the following $2k + 1$ values: $Piece_i(z_j) = \sum_{j'=1}^{n} Piece_i(z_{j'j})$, $j = 0, ..., 2k$.

Output Phase
Step 1. Each party sends his t-shares of the coefficients of $F(x)$ to all other parties.

Step 2. Find out the intersection.

- For $i=1, ..., n$, P_i reconstructs the $2k+1$ coefficients of $F(x)$ using $Piece_1(z_j)$, $Piece_2(z_j)$, ..., $Piece_n(z_j)$, $j = 0, ..., 2k$.
- For $i = 1, ..., n$, P_i evaluates $F(x)$ at each element of his set. If the evaluation is zero then the element belongs to the intersection, else the element does not belong to the intersection. All the elements at which the evaluation is zero form the intersection $S_1 \cap ... \cap S_n$.

In the above protocol, it needs to compute the shares of the coefficients of the multiplication of two polynomials given the shares of the coefficients of these two polynomials. For example, let $f(x)$, $g(x)$ be two polynomials of degree k, whose coefficients are shared among parties, we want to compute the sharing of $h(x) = f(x)g(x)$. Let c_j, $j = 0, ..., k$ be the coefficients of $f(x)$, d_j, $j = 0, ..., k$ be the coefficients of $g(x)$, and let u_j, $j = 0, ..., 2k$ be the coefficients of $h(x)$. We have: $u_0 = c_0d_0$, $u_1 = c_0d_1 + c_1d_0$, ..., $u_k = c_0d_k + c_1d_{k-1} + ... + c_kd_0$, $u_{k+1} = c_1d_k + c_2d_{k-1} + ... + c_kd_1$, ..., $u_{2k} = c_kd_k$. In these $2k + 1$ expressions, there are $(k + 1)(k + 2)$ items of the form c_id_j, and the t-shares of c_id_j can be computed out using the t-shares of c_i, d_j. Then the t-shares of u'_j, $j' = 0, ..., 2k$ can be computed out as below: $Piece_i(u_0) = Piece_i(c_0d_0)$, $Piece_i(u_1) = Piece_i(c_0d_1) + Piece_i(c_1d_0)$, ..., $Piece_i(u_k) = Piece_i(c_0d_k) + Piece_i(c_1d_{k-1}) + ... + Piece_i(c_kd_0)$, $Piecei(u_{k+1}) = Piece_i(c_1d_k) + Piece_i(c_2d_{k-1}) + ... + Piece_i(c_kd_1)$, ..., $Piece_i(u_{2k}) = Piece_i(c_kd_k)$.

3.2 Construction in the Active Model

In the protocol described above, an active adversary can disrupt the security in two ways. One is pointed at [7,8]: the adversary can use zero-polynomial $f(x) = 0$ to replace the the polynomial used to represent a set. The other is that the adversary can send wrong shares to parties when sharing a secret.

If a party uses zero-polynomial to represent his set, then it is equivalent to that the party's set is the union set and he can compute the intersection of other parties' sets. In order to prevent the parties from using zero-polynomial, it is enough to define 1 as the default value of the coefficients of the k-degree items.

In input phase and computation phase, whenever a secret is shared the correctness of sharing should be verified using the share-shares. Additionally, a party should prove that he indeed shares the required value (not a random value) in the re-sharing protocol.

In output phase, each party publishes his shares in order to compute out the coefficients of $F(x)$. For each coefficient of the polynomial $F(x)$, each party gets n shares among which at most t shares are from the adversary (there are at most t wrong shares). In case there are wrong shares, parties can use error correction procedure to correct errors and reconstruct the coefficients correctly.

Using the above methods of preventing cheating activities, the protocol in Section 3.1 can be made secure against an active adversary as below.

Input Phase

Step 1. Each party represents his secret set as a polynomial and shares the coefficients of the polynomial among the parties.

- For $i = 1, ..., n$, P_i represents his secret set as a polynomial $f_i(x)$.
- For $i = 1, ..., n$, P_i two-dimensionally shares the coefficients of $f_i(x)$.
- For $i = 1, ..., n$, P_i checks the correctness of each of the received shares using the corresponding share-shares under the help of other parties.

Step 2. Parties produce jointly the t-sharing of n random polynomials of degree k.

- For $i = 1, ..., n$, P_i chooses randomly n polynomials of degree k: $r_{ij}(x) = b_{ij0} + b_{ij1}x + ... + b_{ijk}x^k$, $j = 1, ..., n$.
- For $i = 1, ..., n$, P_i two-dimensionally t-shares the coefficients b_{ij0}, b_{ij1}, ..., b_{ijk}, $j = 1, ..., n$.
- For $i = 1, ..., n$, P_i verifies the correctness of each of the received shares under the help of other parties.
- For $i = 1, ..., n$, P_i computes the t-shares of the following n polynomials of degree k $r_j(x) = \sum_{j'=1}^{n} r_{j'1}(x) = w_{j0} + w_{j1}x + ... + w_{jk}x^k$, $j = 1, ..., n$, as below: $Piece_i(w_{j0}) = \sum_{j'=1}^{n} Piece_i(b_{j'10})$, $Piece_i(w_{j1}) = \sum_{j'=1}^{n} Piece_i(b_{j'11})$, ..., $Piece_i(w_{jk}) = \sum_{j'=1}^{n} Piece_i(b_{j'1k})$, $j = 1, ..., n$.

Computation Phase

Let $f_j(x)r_j(x) = z_{j0} + z_{j1}x + ... + z_{j,2k}x^{2k}$, $j = 1, ..., n$, and $F(x) = f_1(x)r_1(x) + f_2(x)r_2(x) + ... + f_n(x)r_n(x) = z_0 + z_1x + ... + z_{2k}x^{2k}$.

Step 1. Compute the $2t$-shares of the coefficients of $f_1(x)r_1(x)$, $f_2(x)r_2(x)$, ..., $f_n(x)r_n(x)$.

- For $i = 1, ..., n$, P_i computes locally the following values: $Piece_i(a_{j0})Piece_i(w_{j0})$, $Piece_i(a_{j0})Piece_i(w_{j1})$, ..., $Piece_i(a_{jk})Piece_i(w_{jk})$, $j = 1, ..., n$.

Step 2. Call the re-sharing protocol, and convert the $2t$-shares of the coefficients of $f_1(x)r_1(x)$, $f_2(x)r_2(x)$, ..., $f_n(x)r_n(x)$ into the t-shares.

- For $i = 1, ..., n$, P_i two-dimensionally t-shares the values: $Piece_i(a_{j0})Piece_i(w_{j0})$, $Piece_i(a_{j0})Piece_i(w_{j1})$, ..., $Piece_i(a_{jk})Piece_i(w_{jk})$, $j = 1, ..., n$.
- For $i = 1, ..., n$, P_i checks the correctness of each of the received shares using the corresponding share-shares under the help of other parties.
- For $i = 1, ..., n$, P_i proves that the shared values are indeed the multiplication of his shares: $Piece_i(a_{j0})Piece_i(w_{j0})$, $Piece_i(a_{j0})Piece_i(w_{j1})$, ..., $Piece_i(a_{jk})Piece_i(w_{jk})$, $j = 1, ..., n$.
- For $i = 1, ..., n$, P_i reconstructs the t-shares of $a_{j0}w_{j0}$, ..., $a_{jk}w_{jk}$: $Piece_i(a_{j0}w_{j0})$, $Piece_i(a_{j0}w_{j1})$, ..., $Piece_i(a_{jk}w_{jk})$, $j = 1, ..., n$.
- For $i = 1, ..., n$, P_i computes the t-shares of the coefficients of $f_1(x)r_1(x)$, $f_2(x)r_2(x)$, ..., $f_n(x)r_n(x)$ as below: $Piece_i(z_{j0}) = Piece_i(a_{j0}w_{j0})$, $Piece_i(z_{j1}) = Piece_i(a_{j0}w_{j1}) + Piece_i(a_{j1}w_{j0})$, ..., $Piece_i(z_{j,2k}) = Piece_i(a_{jk}w_{jk})$, $j = 1, ..., n$.

Step 3. Parties compute the t-shares of the $2k+1$ coefficients of $F(x)$.

- For $i = 1, ..., n$, P_i computes the following $2k+1$ values: $Piece_i(z_j) = \sum_{j'=1}^{n} Piece_i(z_{j'j})$, $j = 0, ..., 2k$.

Output Phase
Step 1. Each party sends his t-shares of the coefficients of $F(x)$ to all other parties.

Step 2. Find out the intersection.

- For $i = 1, ..., n$, P_i performs the error correction procedure to get n correct shares for each of the $2k+1$ coefficients of $F(x)$.
- For $i = 1, ..., n$, P_i reconstructs the $2k+1$ coefficients of $F(x)$ using the correct shares.
- For $i = 1, ..., n$, P_i evaluates $F(x)$ at each element of his set. If the evaluation is zero then the element belongs to the intersection, else the element does not belong to the intersection. All the elements at which the evaluation is zero form the intersection $S_1 \cap ... \cap S_n$.

4 Security and Efficiency Analysis of the Proposal

4.1 Correctness and Security

Theorem 1. *In the passive model, suppose at most $t < n/2$ parties collude, the protocol in Section 3.1 is a solution to the private set intersection problem.*

Proof. The correctness of the protocol is based on the polynomial $F(x) = f_1(x)r_1(x) + f_2(x)r_2(x) + ... + f_n(x)r_n(x)$, and the privacy is based on the randomness of $r_1(x)$, $r_2(x)$, ..., $r_n(x)$ and the properties of the secret sharing scheme.

Correctness. Firstly, all the parties get the correct polynomial $F(x)$. When the input phase ends, all the coefficients of $f_1(x)$, ..., $f_n(x)$, $r_1(x)$, ..., $r_n(x)$ are shared correctly among the parties. When the computation phase ends, the coefficients of $F(x)$ are shared correctly among the parties. In the output phase, each party publishes his shares and all the party can compute out the coefficients of $F(x)$, which means that, all the parties learn the polynomial $F(x)$.

Secondly, all the parties learn the correct intersection $S_1 \cap ... \cap S_n$ from $F(x)$. If the element a belongs to the intersection $S_1 \cap ... \cap S_n$, then $f_1(a) = 0$, ..., $f_n(a) = 0$ and $f_1(a)r_1(a) = 0$, ..., $f_n(a)r_n(a) = 0$, and $F(a) = f_1(a)r_1(a) + ... + f_n(a)r_n(a) = 0$. It is to say that, if a belongs to $S_1 \cap ... \cap S_n$, then each party learns $F(a) = 0$ when evaluating $F(x)$ at the element a. If a does not belong to $S_1 \cap ... \cap S_n$, then at least there is one set, i.e., S_i does not include a and $f_i(a) \neq 0$), $i \in \{1, ..., n\}$, so the probability that $f_i(a)r_i(a) = 0$ is negligible (remember that the probability that $r_i(x)$ represents the elements of sets is negligible) and the probability that $F(a) = f_1(a)r_1(a) + ... + f_n(a)r_n(a) = 0$ is negligible. Thus if a does not belong to $S_1 \cap ... \cap S_n$, then the probability that a party whose set includes a determines wrongly that $a \in S_1 \cap ... \cap S_n$ is negligible. This completes the correctness proof.

Privacy. Before output phase, the protocol leaks no information about coefficients of the polynomials, or equivalently, the secret sets. In computation phase, the parties need to reconstruct some secrets when running the re-sharing protocol. In the re-sharing protocol, a party is allowed to reconstruct $Piece_i(c)$, a share of the multiplication of two shared secrets, say a, b. As there are at most t parties collude and t parties cannot recover the shared secrets, so the facts that parties reconstruct the $Piece_i(c)$s does not leak anything information about c, not even to say the secrets a or b.

In output phase, the parties only learn $S_1 \cap ... \cap S_n$ from $F(x)$. The polynomials $r_1(x)$, ..., $r_n(x)$ are random due to the fact that at least $n - t$ parties honestly choose random polynomials from $E(x)$ when jointly producing n random polynomials. The randomness of $r_1(x), ..., r_n(x)$ hides the elements at which the evaluation of $F(x)$ is nonzero. This completes the privacy proof. □

Theorem 2. *In the active model, suppose at most $t < n/3$ parties collude, the protocol in Section 3.2 is a solution to the private set intersection problem.*

Proof. An active adversary cannot disrupt the privacy of the protocol due to two reasons. First, less than $t + 1$ colluding parties cannot recover a shared secret before output phase according to the properties of verifiable secret sharing scheme. Second, the protocol allows at most t parties to collude. So an active who corrupts t parties cannot recover the shared secrets. The proof of privacy is similar to that of Theorem 1, and is omitted here.

Correctness. An active adversary cannot affects the correctness of the protocol. In input phase, the coefficients of $f_1(x), ..., f_n(x)$ are correctly t-shared, since verification of correct sharing is performed whenever a secret is shared. Similarly, the coefficients of the random polynomials $r_1(x), ..., r_n(x)$ are correctly t-shared among the parties. In the computation phase, the parties communicate only when the re-sharing protocol is recalled. In the re-sharing protocol, when a party, say P_i, re-shares his share, all parties jointly verify the correctness of sharing, and P_i is asked to prove that the shared value is indeed the shares that he holds.

So the computation phase is correct. The correctness of the output phase is similar to the proof of Theorem 1 and is omitted here. □

4.2 Efficiency

The protocol only uses simple operation like addition and multiplication, so we only consider the communication complexity of the protocol. The communication complexity is measured in elements in E.

The protocol in the passive model requires 3 rounds of communication. In input phase, the sharing in step 1 and step 2 can be executed in parallel in 1 round. In computation phase, the re-sharing of secrets needs 1 round of communication. In the output phase, parties publish their shares, which requires 1 round of communication.

The communication complexity of the protocol in the passive model is $\mathcal{O}(n^3k^2)$. The communication complexity is dominated by computation phase, where each party needs to share $n(k+1)(k+2)$ secret values and the total communication is $\mathcal{O}(n^3k^2)$.

In the active model, the protocol requires 6 rounds of communication. In input phase, sharing of secrets requires 1 round and verification of correct sharing requires 1 round. In computation phase, re-sharing of shares needs 1 round, verification of correct sharing needs 1 round, and proof that the shared value is indeed the required share needs 1 round. In output phase, publishing the shares needs 1 round.

The communication complexity of the protocol in the active model is $\mathcal{O}(n^4k^2)$. The communication complexity is dominated by computation phase. In computation phase, each party re-shares $n \times (k+1)^2$ secret values, which needs $n \times (k+1)(k+2) \times n^2$ elements to be sent. So the communication complexity of the computation phase is $\mathcal{O}(n^4k^2)$.

5 Conclusion

This paper considers the private set intersection problem in the information-theoretic model. We adopt the technique of polynomial representation of sets used in the previous protocols in the cryptographic model. By representing sets as polynomials, the set intersection problem is converted into the problem of computing the common roots of polynomials. This paper follows the share-compute-recover paradigm for the general protocols in the information-theoretic model, and presents a protocol based on the two-dimensional secret sharing scheme. The protocol consists of input phase, computation phase, and output phase. The protocol allows an active adversary to corrupt $t < n/3$ parties and demands 6 rounds of communication and $\mathcal{O}(n^4k^2)$ elements in a large finite field to be exchanged. It is interesting to consider whether the protocol can be improved with the "dispute control" technique [2] to allow $t < n/2$ colluding parties. In the information-theoretic model, other problems like cardinality set intersection, set union, etc. are also worthy of consideration.

Acknowledgments

The first author would like to show gratitude to Sujing Zhou for the helpful comments on the writing of this paper.

References

1. Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.): Advances in Cryptology - CRYPTO'91. Lecture Notes in Computer Science, Vol. 576. Springer-Verlag Berlin Heidelberg (1992) 420–432
2. Beerliová-Trubíniová, Z., Hirt, M.: Efficient Multi-Party Computation with Dispute Control. In: Halevi, S., Rabin, T. (eds.): the third Theory of Cryptography Conference (TCC 2006). Lecture Notes in Computer Science, Vol. 3876. Springer-Verlag Berlin Heidelberg (2006) 305–328
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In: 20th Annual Symposium on the Theory of Computing (STOC'88). ACM Press (1988) 1–10
4. Freedman, M. J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J. (eds.): Advances in Cryptology - EUROCRYPT'04. Lecture Notes in Computer Science, Vol. 3027. Springer-Verlag Berlin Heidelberg (2004) 1–19
5. Goldreich, O., Micali, S., Wigderson, A.: How to Play Any Mental Game — A Completeness Theorem for Protocols with Honest Majority. In: 19th ACM Symposium on the Theory of Computing (STOC'87). ACM Press (1987) 218–229
6. Hirt, M., Maurer, U., Przydatek, B.: Efficient Secure Multi-Party Computation. In: Okamoto, T. (ed.): Advances in Cryptology - ASIACRYPT'00. Lecture Notes in Computer Science, Vol. 1976. Springer-Verlag Berlin Heidelberg (2000) 143–161
7. Kissner, L., Song, D.: Privacy-Preserving Set Operations. In: Shoup, V. (ed.): Advances in Cryptology - CRYPTO'05. Lecture Notes in Computer Science, Vol. 3621. Springer-Verlag Berlin Heidelberg (2005) 241–257
8. Kissner, L., Song, D.: Private and Threshold Set-Intersection. Technical Report CMU-CS-05-113, Carnegie Mellon University, 2005.
9. Shamir, A.: How to Share a Secret. Communications of the ACM, Vol. 22. ACM Press (1977) 612–613
10. Yao, A. C.: Protocols for Secure Computations. In: 23rd IEEE Symposium on Foundations of Computer Science (FOCS'82). IEEE Computer Security (1982) 160–164

Privacy-Preserving Set Union

Keith Frikken

Department of Computer Science and Systems Analysis
Miami University
Oxford, OH 45056

Abstract. Recently there has been a significant amount of work on privacy-preserving set operations, including: set intersection [14,6,21,9], testing set disjointness [17], multi-set operations [18], and set union [16,1,18]. In this paper, we introduce novel protocols for privacy-preserving set union in the malicious adversary model. More specifically, each participant inputs a set of values, and at the end of the protocol, each participant learns the items that are in at least one participant's set without learning the frequency of the items or which participant(s) contributed specific items. To our knowledge our protocol is the most efficient privacy-preserving set union protocol for the malicious adversary model to date.

1 Introduction

Recently there has been a significant amount of work on privacy-preserving set operations, including: set intersection [14,6,21,9], testing set disjointness [17], multi-set operations [18], and set union [16,1,18]. In this paper, we introduce a new protocol for privacy-preserving set union ($\mathcal{PPSU}$) in the malicious adversary model. We are only aware of one other $\mathcal{PPSU}$ protocol that is secure in the malicious adversary model, which was introduced in [18], and our new protocol is more efficient than this protocol. An application of this work is the following scenario: Suppose several hospitals treat a rare disease and that a research group needs various information about patients with the disease in order to develop alternative treatments. Because cases of the disease are so sparse, the research group needs more than a single hospital's data. Unfortunately, some patients may have gone to multiple hospitals and these patients' information will taint the quality of the collected data. By using a secure set union protocol the research group can gather the information from multiple hospitals while omitting duplicate patients without learning the identity of the patients. Furthermore, $\mathcal{PPSU}$ has been used as a building block in some privacy-preserving data mining protocols [16,23] and privacy-preserving graph algorithm protocols [1], and thus we believe that improved protocols for $\mathcal{PPSU}$ will be useful in many application domains.

Contributions: The contributions of this work are not only in a preliminary protocol for $\mathcal{PPSU}$, but also in several extensions of this protocol. We now summarize our results:

- *Preliminary Protocols:* Our preliminary protocols securely compute the set union for: i) two parties in the honest but curious adversary model with $O(n)$ communication, ii) multiple parties in the honest-but-curious adversary model with $O(k^2 n)$ communication, and iii) multiple parties in the malicious adversary model with $O(n^2 k^2 + k^3 n)$ communication (for sets of size n and k participants). This is described in section 5. Note that for the complexities given in this paper there is always an implicit security parameter.
- *Padding:* Our preliminary protocols reveal the number of items in each participant's sets. In section 6.1 we introduce modifications to our preliminary protocol that allow participants to pad their sets with "dummy" items in order to obfuscate this information. This is described in section 6.1.
- *Cardinality:* For some applications computing the size of the set union without revealing the actual union is preferred. In section 6.2 we introduce a protocol that reveals only the cardinality of the set union.
- *Empty-set attack:* One problem with any set union protocol is that a dishonest participant can use the empty-set as their input set. This is particularly damaging if there are few participants; e.g., when there are only two participants then this reveals the honest party's set. In section 6.3 we introduce counter-measures against this attack.
- *Over-threshold set union:* As discussed in [18] there are some situations where participants want to know all items that are in at least t sets. In section 6.4 we introduce protocols that compute this "over-threshold" set union.

Outline: The rest of this paper is organized as follows. In section 2 we formally describe the set union problem and the security models considered in this paper. In section 3 we provide a detailed summary of previous work in privacy-preserving set operations. In section 4, we introduce several building blocks that are used in our protocols, but which are not contributions of this paper. In section 5 we introduce preliminary protocols for privacy-preserving set union. In section 6, we describe several extensions to the preliminary protocols. Finally, we conclude our paper in section 7.

2 Problem Definition

There are k participants, labeled $P_1, \ldots, P_k$, that have respective sets $S_1, \ldots, S_k$ that are drawn from a universe of items $\mathcal{U}$. As a shorthand notation, we use η_i to represent $|S_i|$. To denote the specific items in a set S_i we use the notation $(S_i)_1, \ldots (S_i)_{\eta_i}$. Note that the above notation implies an ordering on the items, but this is just for ease of notation; that is, we do not assume the items are in any specific order. To simplify the notation when giving the complexity analysis of our protocols we use $n = \max_{i=1}^{n} \eta_i$ and we assume that every party has n items. The desired output of the protocol is that the participants learn $\cup_{i=1}^{k} S_i$.

We define security in the standard way (see [10] for more details), that is we define an ideal model (using a trusted third party) and show that any polynomial-time adversary in our protocol can be simulated by a polynomial-time adversary in the ideal model.

Ideal Model: In the ideal model the participants send their sets to a trusted third party T, and then this party broadcasts the union of the sets along with the size of each individual set to all of the participants. Note that this reveals slightly more than the set union, specifically each participant's set size is revealed.

Honest-But-Curious Adversaries: In this adversary model the participants will faithfully follow the prescribed protocol, but will try to learn additional information after the protocol. To prove security in this model, we show that the transcript that is produced by our protocol could be simulated by an adversary that has the output of the protocol. More specifically, we require that the simulator be able to generate a transcript that is computationally indistinguishable from the real transcript.

Malicious Adversaries: In the malicious adversary model the adversary will deviate from the protocol in an arbitrary fashion. The purpose of this deviation can be several things, including: i) to learn more information about honest participants' values, ii) to change the result of the protocol, or iii) to terminate the protocol prematurely. In this paper we do not consider early termination to be a problem (although our protocols could be modified to prevent this using standard techniques such as [11]). Thus to show our protocol is secure in this model we show that: i) any transcript generated by the protocol can be simulated given the results of the protocol and ii) that any result-changing action by an adversary could be achieved by changing the adversary's inputs in the ideal protocol.

3 Related Work

The $\mathcal{PPSU}$ problem can be solved with the generic results of secure multiparty computation [24,11]. While recent advances in malicious circuit evaluation [3] show that it is possible to simulate a circuit efficiently in the malicious model, the communication complexity of such a the scheme will still be the number of gates in the circuit times a security parameter times a polynomial of k (the specific polynomial depends on the scheme being used). The straight-forward circuit for set union has $O(k^2 n^2 \log |\mathcal{U}|)$ gates (when given k parties whose sets each contain n elements).

As mentioned earlier, many privacy-preserving protocols have been introduced for set operations other than set union, including: set intersection [14,6,21,9], testing set disjointness [17], and multi-set operations [18]. One might think that, because of DeMorgan's Law, a secure protocol for set union follows directly from a secure protocol for set intersection. Specifically, one can compute $\cup_{i=1}^{n} S_i$, by computing $\overline{\cap_{i=1}^{n} \overline{S_i}} = \mathcal{U} - (\cap_{i=1}^{n}(\mathcal{U} - S_i))$. While this method does correctly compute the set union, when $\mathcal{U}$ is significantly larger than $\cup_{i=1}^{n} S_i$ this method is inefficient. Thus, the results of this paper are most beneficial for applications where the sets are chosen from large domains.

There have also been several protocols that privately compute set union [16,1,18]. However, the previous solutions have not been fully satisfactory solutions. In [16], the protocol reveals superfluous information, such as the cardinality of the intersection between some participants' sets, in order to improve efficiency. The protocols in [1] was proven secure only in the honest-but-curious adversary model (this protocol was for two parties and required $O(n \log |U|)$ communication). Finally, the protocol given in [18] is secure in the malicious model, but according to our analysis[1] the communication complexity of their honest-but-curious approach is $O(k^3n^2)$, whereas our scheme has communication complexity $O(k^2n^2 + k^3n)$.

4 Building Blocks

In this section we outline the building blocks that are used by our protocols.

4.1 Homomorphic Encryption

In this paper we use a public-key semantically-secure [12] additively homomorphic encryption scheme, such as [22]. Throughout this paper we will denote the encryption and decryption functions by E_{pk} and D_{sk} respectively. Recall that it is possible to add the plaintexts of two encrypted values by multiplying the ciphertexts; that is, when given the encryptions $E_{pk}(x)$ and $E_{pk}(y)$, we can compute $E_{pk}(x + y)$ by computing $E_{pk}(x) * E_{pk}(y)$. Also, when given $E_{pk}(x)$ it is possible to compute $E_{pk}(c*x)$ for any constant c by computing $E_{pk}(x)^c$. Finally, we use homomorphic schemes where it is possible to re-encrypt a ciphertext value to generate another ciphertext with the same plaintext value.

We utilize a threshold version of Paillier's scheme throughout this paper, such as the one presented in [2,4,8]. More specifically, we require a (k, k)-threshold decryption algorithm, that is, the decryption key is distributed among all k players, and the participation of all k players are required to decrypt a value. We use the same model as [18], and we assume that the threshold keys have already been distributed amongst the participants. The communication required to perform a joint decryption is $O(k)$.

4.2 Polynomial Representation of Sets

Several previous set operation results use polynomials to represent sets or multisets [9,18]. Specifically, to represent a multi-set $S = \{s_1, \dots, s_n\}$ we use the polynomial $(x - s_1)(x - s_2) \cdots (x - s_n)$, which we denote by $f_S(x)$. An important property of this representation is that a value y is in the set S if and only if $f_S(y) = 0$.

To hide the value of a polynomially-represented set, it is often useful to encrypt the set's polynomial using homomorphic encryption. Suppose we are given a polynomial $f(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0$, then the encryption of

[1] Based on our analysis of the THRESHOLD-PERFECT-HBC protocol given in [18]. An explicit protocol for the malicious model was not given, but by adding zero-knowledge proofs to the steps such a protocol could be constructed.

f, denoted by $E_{pk}(f)$, is the encryption, using E_{pk} on the coefficients of f, i.e., $E_{pk}(a_n), \ldots, E_{pk}(a_0)$. As described in previous work [9,18] when given $E_{pk}(f)$ it is possible to perform many operations on f. In this paper, we use the following operations:

1. *Polynomial Evaluation:* Given $E_{pk}(f)$, the public parameters of E_{pk} and a value x it is possible to compute $E_{pk}(f(x))$.
2. *Polynomial Addition:* Given $E_{pk}(f)$ and $E_{pk}(g)$ for two polynomials f and g, then it is possible to compute $E_{pk}(f+g)$.
3. *Polynomial Multiplication:* Given $E_{pk}(f)$ and g for two polynomials f and g it is possible to compute $E_{pk}(f * g)$.
4. *Polynomial Derivation:* Given $E_{pk}(f)$ it is possible to compute the encrypted polynomial of the derivative of f, i.e. it is possible to compute $E_{pk}(f')$.

4.3 Zero Knowledge Proofs

To extend our protocols to the malicious adversary model, we utilize zero knowledge proofs. In what follows, we outline the proofs of knowledge that are used in this paper. These proofs are similar to those used in [18] and can be efficiently realized using [2] and [5]. These proofs can be made non-interactive using the Fiat-Shamir heuristic [7].

In what follows, E_{pk} is a threshold additive-homomorphic encryption scheme.

1. $POPK(E_{pk}(x))$ represents a proof that the prover knows the plaintext x (i.e., it is a proof of plaintext knowledge). Furthermore, this proof can be done with $O(1)$ communication complexity.
2. *Proof of Correct Multiplication:* Given $E_{pk}(x)$ (where x may be unknown to the prover) and a value y, it is possible to publish values $E_{pk}(y)$ and $E_{pk}(z)$ and prove that $z = xy$. We denote this proof by $ZKPK(y|a = E_{pk}(y) \wedge b = E_{pk}(z) \wedge z = x * y)$. Furthermore, this proof can be done with $O(1)$ communication complexity.
3. *Proof of Correct Polynomial Evaluation:* When the prover is given $E_{pk}(f)$ for some polynomial f, then the prover can generate values $E_{pk}(x)$ for a known value x and $E_{pk}(z)$ along with a proof that $z = f(x)$. If the polynomials have degree n then, this proof requires $O(n)$ proofs of correct multiplication. We denote this proof by $ZKPK(x|y = E_{pk}(f(x)) \wedge z = E_{pk}(x))$.
4. *Proof of Correct Polynomial Multiplication:* When a prover is given $E_{pk}(f)$ for some polynomial f and another polynomial g, then the prover can generate $E_{pk}(g)$ and $E_{pk}(h)$ along with a proof that $h = f * g$. If the polynomials f and g have respective degree m and n, then this protocol requires $O(mn)$ proofs of correct multiplication. We denote this proof by $ZKPK(f|h = f * g \wedge y = E_{pk}(f))$.
5. *Proof of Correct Polynomial Construction:* If a prover has posted encryptions of a list of values $E_{pk}(x_1), \ldots, E_{pk}(x_n)$. Then the prover can post the encrypted polynomial $(x - x_1) \cdots (x - x_n)$ along with a proof that it was constructed properly. This requires $O(n^2)$ proofs of correct multiplication.

4.4 Mixes and Shuffles

In our protocols we use a cryptographic protocol for shuffling (e.g., mixing) a list of encrypted values. Specifically, a shuffle protocol uses list of encrypted values $E_{pk}(x_1), \ldots, E_{pk}(x_n)$ as input and the output of the protocol is another list of encrypted values $E_{pk}(y_1), \ldots, E_{pk}(y_n)$. Furthermore, the y-list is a permutation of the original x-list and any group of participants that is not a quorum (of the threshold encryption scheme) cannot associate a specific y-value back to a specific x-value. In our protocols we utilize a robust shuffling protocol where the participants obtain proof(in zero-knowledge) that the shuffle was performed correctly. Such a robust mix can be made from standard protocols [20,19,15]. We actually use a slight variation of a standard mix, in that our protocols shuffle tuples. That is, the input to the protocol is a list of tuples of encrypted values and the output is a permuted list of re-encrypted tuples that have a different tuple order, but the individual values inside of a tuple are in the same order. This can be achieved with techniques from [13]. The communication required to robustly mix n tuples with k participants is $O(k^2 n)$.

4.5 Bulletin Board

Our protocols use a bulletin board abstraction (i.e., all parties post information to a common area) that can be constructed using standard cryptographic techniques. We measure the communication requirements of our protocols as the amount of information posted on the bulletin board. It is worth noting that we do not need a robust bulletin board for our application (we are not trying to be secure against adversaries that try to force early termination). Thus our protocols could just utilize standard broadcast techniques.

5 Preliminary Protocols

In this section we give preliminary protocols for the honest-but-curious adversary model and the malicious adversary model. As a starting point we introduce a protocol for the two-party honest but curious adversary model, then we extend this to multiple parties, and we then extend it to the malicious model. It is worth noting that the malicious protocol is similar to the honest-but-curious protocol, but the protocols are presented as two separate protocols to enhance readability. As not to clutter this initial presentation, we postpone the discussion of several extensions to these protocols until the next section.

5.1 Two-Party Honest But Curious

In this section we propose a two-party protocol for set union in the honest but curious adversary model where only one party learns the result. This protocol is the most efficient such protocol to date that the authors are aware of, and requires only $O(n)$ communication. The main idea of this technique is as follows: Suppose that a participant has the encrypted polynomial, denoted by $E_{pk}(f_S)$

for some set S, then this participant can blindly evaluate this polynomial on each of his set items; we will denote a specific such value by $E_{pk}(f_S(s))$. Now $f_S(s) = 0$ if and only if $s \in S$ (with high probability). So if we create a tuple of the form: $(E_{pk}(f_S(s) * s) ; E_{pk}(f_S(s)))$, then this tuple will be $(0 ; 0)$ if $s \in S$ and otherwise s can be recovered from the decrypted tuple values.

Thus a high level protocol for the two-party case is as follows: Participant P_1 encrypts his set as a polynomial and sends it to P_2 (using a homomorphic scheme that is chosen by P_1). P_2 then computes the above tuples and sends them back to P_1 in a random order. P_1 then decrypts the tuples to learn the values in S_2 that are not in S_1, these values are then added to the items in S_1 to produce the output of the protocol. The full description of the protocol is given in Figure 1.

Setup: Participant P_1 chooses a homomorphic encryption scheme, and denote the public encryption function by E_{pk}.

1. P_1 sends $E_{pk}(f_{S_1})$ along with the public parameters of E_{pk} to P_2.
2. For each value $s \in S_2$, P_2 chooses a random value r (chosen uniformly) and computes a tuple $(E_{pk}(f_{S_1}(s) * s * r) ; E_{pk}(f_{S_1}(s) * r))$. P_2 randomly permutes all of the tuples and sends them to P_1.
3. P_1 initially sets the output set to be S_1. For each tuple $(E_{pk}(x) ; E_{pk}(y))$ from the previous step, P_1 decrypts x and y. If both values are 0, then P_1 continues to the next tuple. Otherwise, P_1 adds $x * y^{-1}$ to the output set.

Fig. 1. Two-party HBC protocol for Set Union

Complexity Analysis: Step 1 of the protocol requires $O(n)$ communication and computation. It requires $O(n)$ computation to compute the value $f_{S_1}(s)$ for a single value s, and so Step 2 requires $O(n^2)$ computation. However, Step 2 requires only $O(n)$ communication. Finally, Step 3 requires $O(n)$ computation. Thus this protocol requires $O(n^2)$ computation, $O(n)$ communication, and $O(1)$ rounds. The computation can be reduced to $O(n \log\log n)$ using the bucketing techniques of [9][2].

Security Analysis: We must show that the communication transcript from this protocol is simulatable from the results of the protocol alone along with one of the participant's inputs. This is trivial to do in the case of P_2 since all communication from P_1 is encrypted with a semantically-secure homomorphic encryption scheme (where the private key is known only to P_1). The proof to show that the protocol is secure against P_1 is also straightforward, but is not as trivial. Suppose that a simulation algorithm has: $|S_2|$, S_1, and $S_1 \cup S_2$. Clearly, the simulator can compute $S_2 - S_1$ from the above information. The simulator then proceeds as follows: i) it computes $|S_2| - |S_2 - S_1|$ tuples of the form $(E_{pk}(0) ; E_{pk}(0))$, ii) it computes a tuple $(E_{pk}(s * r) ; E_{pk}(r))$ for a randomly chosen value r for each item $s \in S_2 - S_1$, and iii) it randomly permutes the tuples from the previous

[2] This requires minor modifications to the protocol.

two steps and outputs this as the simulated transcript. It is straightforward to show that this simulated transcript is indistinguishable from the values sent to P_1 in Step 2 of the protocol.

5.2 Multi-party Honest But Curious

In this section we introduce the preliminary protocol for the honest-but-curious adversary model for multiple participants. To compute the union of the individual sets, the protocol computes the multi-set union (using techniques of [18]) for each of the following sets of participants $\{P_1\}, \{P_1, P_2\}, \ldots, \{P_1, \ldots, P_n\}$. Then each participant P_i reports every item in S_i that is not in the multi-set union of $S_1, \ldots, S_{i-1}$ using the reporting technique from the previous section. The tuples are then mixed, to hide the source of each tuple, and then are decrypted to reveal the items in the set union. The full description of the protocol is given in Figure 2.

Complexity Analysis: In what follows we list the communication requirements of each step of the protocol:

1. *Step 1.a:* This requires $O(n)$ communication for each participant, and thus this requires $O(kn)$ total communication.
2. *Step 1.b:* This requires $O(in)$ communication for participant P_i, and thus this requires $O(k^2n)$ total communication.
3. *Step 1.c*: Each participant has to post $O(n)$ tuples, and thus this requires $O(kn)$ total communication.
4. *Step 2.b:* Each participant has to post $O(kn)$ tuples, and thus this requires $O(k^2n)$ total communication.
5. *Step 3:* Since we are mixing $O(kn)$ values in a non-robust manner this requires $O(k^2n)$ total communication.
6. *Step 4:* Each decryption requires $O(k)$ communication, and so this requires $O(k^2n)$ total communication.

In summary, this protocol requires $O(k^2n)$ communication and $O(k)$ rounds.

Security Analysis: We must show that the communication transcript from this protocol is simulatable from the results of the protocol alone. This is a relatively straight-forward simulation, and so we are a bit informal throughout this discussion. Suppose that a simulation algorithm is given $\eta_1, \ldots, \eta_k$ along with $\cup_{i=1}^{k} S_i$. Now the simulation algorithm can easily create k simulation sets $SS_1, \ldots, SS_k$ such that $|SS_i| = \eta_i$ and $\cup_{i=1}^{k} SS_i = \cup_{i=1}^{k} S_i$. Now the simulation algorithm simply mimics the protocol in Figure 2 (for the shuffling phase it acts as all of the mix servers) and outputs the transcript of this session. We claim that this simulated transcript is indistinguishable from transcript generated by the real protocol to any adversary that does not establish a quorum of participants. First, in steps 1 and 2 everything is encrypted with a semantically-secure cryptosystem so these values will be indistinguishable. Step 3 is indistinguishable because both are runs of a mix protocol. Finally, Step 4 is indistinguishable because both are randomly permuted lists tuples, and we claim that decrypted tuples that reveal a set value

Setup: The participants have agreed on a threshold Homomorphic encryption scheme, and denote the public encryption function by E_{pk}.

1. *Build tuples:*
   (a) *Post polynomial representation of sets:* Participant P_i posts $E_{pk}(f_{S_i})$ to the bulletin board.
   (b) *Post polynomials:* Participant P_i (for $i = 2, \ldots, k$) posts $E_{pk}(\prod_{j=1}^{i}(f_{S_j}))$ to the bulletin board.
   (c) *Post tuples:* Participant P_i posts the following tuples to the bulletin board:
      i. P_1 posts $(E_{pk}(s)\ ;\ (E_{pk}(1))$ for each value $s \in S_1$.
      ii. Participants P_i (for $i = 2, \ldots, k$) posts tuples for every $s \in S_i$ as follows: $\left(E_{pk}(\prod_{j=1}^{i-1}(f_{S_j}(s) * s)\ ;\ E_{pk}(\prod_{j=1}^{i-1}(f_{S_j}(s))\right)$.
2. *Randomize tuple values:* For each tuple in $(E_{pk}(x)\ ;\ E_{pk}(y))$ that was posted by any participant in the previous step, the participants do the following (Note that this step can be done in parallel for all tuples):
   (a) Each participant, P_i chooses a non-zero random values r_i chosen uniformly.
   (b) The participants multiply the tuple's values by their random value. That is, P_i posts $\left(E_{pk}(x * \prod_{j=1}^{i} r_j)\ ;\ E_{pk}(y * \prod_{j=1}^{i} r_j)\right)$ to the bulletin board. Note that P_i must wait until P_{i-1} has posted his tuples before P_i can post his tuples.
3. *Shuffle:* The parties engage in a secure shuffle protocol for all of the tuples generated by P_k in the previous step.
4. *Decrypt results:* For each tuple $(E_{pk}(x)\ ;\ E_{pk}(y))$, the parties jointly decrypt x and y. If both values are 0, then the parties continue to the next tuple. Otherwise, the parties add $x * y^{-1}$ to the output set.

Fig. 2. Multi-party HBC protocol for Set Union

s are indistinguishable from a tuples $(s * r\ ;\ r)$ for some random value r. This follows from Step 2 of the protocol, because as long as one participant is honest the values will be multiplied by a random value unknown to the adversary.

5.3 Malicious Model

In this section we introduce the protocol for the malicious model. This protocol is similar to the honest-but-curious protocol, however there are a few crucial differences. The main difference is that the parties commit to their set values and then at each step of the protocol the parties prove in zero knowledge that they are following the protocol correctly. The full description of the protocol is in Figure 3.

Complexity Analysis: In what follows we list the communication requirements of each step of the protocol:

1. *Step 1:* Each participant has to post $O(n)$ values, and thus the total communication is $O(kn)$.

Setup: The participants have agreed on a threshold Homomorphic encryption scheme, and denote the public encryption function by E_{pk}.

1. *Commit to sets:* Participant P_i posts the following values to the bulletin board $E_{pk}((S_i)_1), \ldots, E_{pk}((S_i)_{\eta_i})$ along with a proof of plaintext knowledge.
2. *Build tuples:*
   (a) *Post polynomial representation of sets:* Participant P_i posts $E_{pk}(f_{S_i})$ to the bulletin board along with a proof of correct polynomial construction (using the commitments from the previous step).
   (b) *Post polynomials:* Participant P_i (for $i = 2, \ldots, k$) posts $E_{pk}(\prod_{j=1}^{i}(f_{S_j}))$ to the bulletin board along with a proof of correct polynomial multiplication.
   (c) *Post tuples:* Participant P_i posts the following tuples to the bulletin board:
      i. P_1 posts $(E_{pk}(s)\ ;\ (E_{pk}(1))$ for each value $s \in S_1$ along with a proof of correct construction.
      ii. Participants P_i (for $i = 2, \ldots, k$) posts tuples for every $s \in S_i$ as follows: $\left(E_{pk}(\prod_{j=1}^{i-1}(f_{S_j}(s) * s)\ ;\ E_{pk}(\prod_{j=1}^{i-1}(f_{S_j}(s))\right)$ along with zero knowledge proofs that this tuple is formed correctly. Specifically, the participant posts a proof of correct polynomial evaluation and a proof of correct multiplication.
3. *Randomize tuple values:* For each tuple in $(E_{pk}(x)\ ;\ E_{pk}(y))$ that was posted by any participant in the previous step, the participants do the following (Note that this step can be done in parallel for all tuples):
   (a) Each participant, P_i chooses a random values r_i. P_i also posts a commitment of this random value $E_{pk}(r_i)$ to the bulletin board along with a proof that r_i is non-zero.
   (b) The participants multiply the tuple's values by their random value. That is, P_i posts $\left(E_{pk}(x * \prod_{j=1}^{i} r_j)\ ;\ E_{pk}(y * \prod_{j=1}^{i} r_j)\right)$ to the bulletin board along with a proof of correct construction. Note that P_i must wait until P_{i-1} has posted his tuples before P_i can post his tuples.
4. *Shuffle:* The parties engage in a secure shuffle protocol for all of the tuples generated in the previous step by P_k.
5. *Decrypt results:* For each tuple $(E_{pk}(x)\ ;\ E_{pk}(y))$, the parties jointly decrypt x and y. If both values are 0, then the parties continue to the next tuple. Otherwise, the parties add $x * y^{-1}$ to the output set.

Fig. 3. Malicious protocol for Set Union

2. *Step 2.a:* Each participant has to post $O(n^2)$ data (the size of the correct polynomial construction proof), and thus the total communication is $O(kn^2)$.
3. *Step 2.b*: This requires $O(in^2)$ communication for participant P_i (who must do a proof of correct polynomial multiplication between a polynomial of size $O(in)$ and a polynomial of size $O(n)$), and thus this requires $O(k^2n^2)$ total communication.
4. *Step 2.c:* This requires $O(in^2)$ communication for participant P_i (who must do a proof of correct polynomial evaluation for a polynomial of size $O(in)$ on $O(n)$ values), and thus this requires $O(k^2n^2)$ total communication.

5. *Step 3.a:* This requires each participant to post $O(kn)$ communication, and thus the total communication is $O(k^2n)$.
6. *Step 3.b:* This requires each participant to post $O(kn)$ communication, and thus the total communication is $O(k^2n)$.
7. *Step 4:* This requires a robust mix of $O(kn)$ values with k participants. Thus this requires $O(k^3n)$ total communication.
8. *Step 5:* Each decryption requires $O(k)$ communication, and so this requires $O(k^2n)$ total communication.

In summary, this protocol requires $O(k^2n^2 + k^3n)$ communication and $O(k)$ rounds.

Security Analysis: In this section we give an argument for security for the malicious model protocol. All of the communication in steps 1-4 are either values encrypted with a homomorphic encryption scheme or are zero knowledge proofs. It is easy to verify that the zero knowledge proofs do not reveal anything other than predicates of the form: was this step done properly. Thus all of these steps can easily be simulated in a way that is indistinguishable from the real transcript. However, we must show that the decrypted values in Step 5 of the protocol can be simulated from the output of the protocol in a manner that is indistinguishable from the values in the protocol (even if a group of participants deviates from the protocol). If any of the zero knowledge proofs fail, then the protocol terminates and Step 5 is not reached, and so in what follows we assume that the zero knowledge proofs have all passed.

In Step 1 of the protocol the participants submit of list of committed values. Clearly, this same set of values could be injected into the ideal model (assuming that the ideal model allows multi-set inputs). We must show that the simulator with the result from the ideal model will produce a list of values that are indistinguishable from these values. To do this, we first define the simulation algorithm. Suppose that $S = \bigcup_{i=1}^{k} S_i$, and that $N = \sum_{i=1}^{k} |S_i|$. The simulation then proceeds as follows: i) it produces $N - |S|$ tuples of the form $(0\ ;\ 0)$ and for each value $s \in S$ it creates a tuple of the form $(s * r\ ;\ r)$ for a randomly chosen value r. The simulation algorithm randomly permutes these N tuples and outputs this as the transcript for Step 5.

The following list enumerates the state of the N tuples that are produced in each of the steps of the protocol

1. *Step 2:* Participant P_i's jth tuple will be $(0\ ;\ 0)$ if $(S_i)_j$ is in one of the sets $S_1, \ldots, S_{i-1}$. Otherwise, it will be $\Big((S_i)_j * v\ ;\ v\Big)$ for a value v that may reveal information.
2. *Step 3:* This step multiplies the values in each tuple by a random non-zero value using a standard protocol (i.e., everyone multiplies it by their own random non-zero value). Thus participant P_i's jth tuple will be $(0\ ;\ 0)$ if $(S_i)_j$ is in one of the sets $S_1, \ldots, S_{i-1}$. Otherwise, it will be $\Big((S_i)_j * r\ ;\ r\Big)$ for a randomly chosen value r. Thus at the end of this protocol there will be $N - |S|$ tuples of the form $(0\ ;\ 0)$ and for each value $s \in S$ it creates a

tuple of the form $(s * r\ ;\ r)$ for a randomly chosen value r. This is just like the simulation algorithm except that the order of the tuples in the protocol at the end of this step reveals information.

3. *Step 4:* The tuples are randomly shuffled with a robust mix. Thus, the tuples will be the same as in the previous step but are in a random order.

6 Extensions

In this section we introduce various extensions of the preliminary protocols in the previous section, including: padding sets, computing the cardinality, countering the empty-set attack, and computing the over-threshold set union.

6.1 Padding

As mentioned earlier, the preliminary protocols leak the number of values that each party has in their set. In this section we introduce a method for padding a list to obfuscate this value, and thus all that is revealed is an upper bound on the cardinality of each party's set. Suppose that a participant P_i wants to report a set S_i and a size η_i where $\eta_i \geq |S_i|$. In the HBC protocol P_i would make the following changes: i) The participant would use a polynomial $g_{S_i,\eta_i} = f_{S_j} * x^{\eta_i - |S_i|}$ and would use this as his polynomial in Step 2 and ii) when reporting his "dummy" values in Step 3 of the protocol the participant posts $(E_{pk}(0)\ ;\ E_{pk}(0))$.

The changes to the malicious protocol are a little more involved (to prevent the participants from failing a zero knowledge proof). The main change is that the subject will commit to a larger set of values where the dummy values set to a value not in $\mathcal{U}$ and a dummy item's random hiding factor in Step 3 is set to 0. It is also required that the proofs that a random values are non-zero are removed for a participant's own values. The rest of the protocol then remains unchanged.

6.2 Cardinality

There are some situations where the goal is to reveal only the cardinality of the set union. It is relatively straight-forward to modify the preliminary protocols to compute the cardinality. In Step 1 (of the HBC protocol) and Step 2 (in the malicious protocol) all that needs to be changed is that when creating the tuples, participant P_i used 1 instead of $(S_i)_j$. Now, when the tuples are decrypted duplicate items will still be $(0\ ;\ 0)$ but first-time items will be $(r\ ;\ r)$ (for some random value r). Thus, the number of tuples that are not $(0\ ;\ 0)$ is the cardinality of the set union.

6.3 Empty-Set Attack

It is possible for a malicious party to set their set to the empty-set. This is particularly damaging when there are only two participants, as this will reveal the honest participant's exact set. We now outline several strategies for countering this attack:

1. A simple solution is to require that each participant's set is not the empty-set. One way of doing this is to make sure that the leading coefficient of the polynomial is non-zero. However, this has limited effectiveness, because the adversary can use n items that are very infrequent as their input set. Or worse, the adversary could use a non-decomposable polynomial (i.e., one that is never 0, for the protocol).
2. A more complicated approach would be to make sure that each party's set has certain properties. However, these requirements would vary from domain to domain, and so each new domain would require a separate protocol for determining if a set is valid. Thus, this is not a general solution.
3. Another approach is for participants to require some overlap between the other participant's set and their own. Thus in the two party-case the participants would first engage in a cardinality of set intersection protocol and would continue only if this cardinality was over a threshold. This check can be done without revealing the actual cardinality. This is not a perfect approach, because an adversary can still test if an honest participant has a specific item (or small set of items), but it does help prevent complete revelation of the honest party's set in a single run of the protocol.

6.4 Over-Threshold Set Union

In this section we introduce a protocol for computing all items that appear t or more times. This protocol does not reveal how many times an item appears (even if it is in the result). A similar protocol was given in [18], but the given protocol reveals how many times an item in the result appeared. Of course, many of the ideas from [18] could be combined to make such a protocol, we believe that the following protocol will be more efficient than such a protocol. However, some of the ideas in the following protocol were also presented in [18] (specifically taking the $(t-1)$th derivative to determine if an item has appeared t or more times). We present this protocol for the honest-but-curious model in Figure 4; a malicious model protocol will be given in the full version of the paper[3].

The main idea behind this protocol is that each participant first learns which of its items appears t or more times, and then these values are used in a standard set union protocol to merge the items and to hide the multiplicity of the items. Clearly, this intermediate result is simulateable from the output alone (i.e., given all items that appear t or more times a participant can compute which items in its set appear t or more times). To compute which items are in t or more sets the participants compute the $(t-1)$th derivative of the polynomial for the multi-set union of every participant's set. Note that when this polynomial evaluates to 0 for a specific set item then the item will have appeared t or more times with high probability.

[3] It is worth noting that the protocol in Figure 4 would not work by simply adding zero knowledge proofs, because this would not prevent a party from submitting multiple copies of the same value.

Setup: The participants agree on a threshold Homomorphic encryption scheme, and denote the public encryption function by E_{pk}.

1. *Compute multi-set union*: Using Step 1 of the protocol in Figure 2 the participants compute $E_{pk}(\prod_{j=1}^{k}(f_{S_j}))$, which we denote my $E_{pk}(m)$.
2. The participants then compute the value $E_{pk}(m^{(t-1)})$ (i.e., the $(t-1)$th deriviative of the multi-set union).
3. For each value $s \in S_i$ participant P_i posts the following tuple $\Big(E_{pk}(m^{(t-1)}(s))\ ;\ E_{pk}(s)\Big)$ to the bulletin board.
4. Using Step 2 of the protocol in Figure 2 the participants multiply the first value of each tuple by a random value and post the new values to the bulletin board.
5. For each tuple $\Big(E_{pk}(m^{(t-1)}(s) * r)\ ;\ E_{pk}(s)\Big)$ that was posted in the previous step (the participants compute $E_{pk}((m^{(t-1)}(s) * r) + s)$ and jointly decrypt this value so that only the participant that contributed the value will learn the plaintext.
6. For each item that a participant posted in Step 3 they obtain either the value itself or a random value. Each participant then builds a new set of the items that survived elimination and pads the list to its original size. The participants then engage in a privacy-preserving protocol for Set Union with these new sets.

Fig. 4. HBC protocol for Over-Threshold Set Union

7 Conclusions

In this paper we introduced protocols for privacy-preserving set union that are more efficient than previous such protocols. We believe that these new protocols will have many applications in data mining and other domains. We also introduced a new over-threshold set union protocol.

Acknowledgements

The author would like to thank the anonymous reviewers for their comments and useful suggestions.

References

1. J. Brickel and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Proceedings of Advances in Cryptology - ASIACRYPT '05*, volume 3788 of *Lecture Notes in Computer Science*, pages 236–252, 2005.
2. R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proceedings of Advances in Cryptology - EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299, 2001.
3. I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Proceedings of Advances in Cryptology - CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–411, 2005.

4. I. Damgard and M. Jurik. Efficient protocols based on probabilistic encryption using composite degree residue classes, 2000.
5. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136, London, UK, 2001. Springer-Verlag.
6. A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222, New York, NY, USA, 2003. ACM Press.
7. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, 1986.
8. Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. *Lecture Notes in Computer Science*, 1962: 90–104, 2001.
9. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Proceedings of Advances in Cryptology - EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, 2004.
10. O. Goldreich. *Foundations of Cryptography: Volume II Basic Application.* Cambridge University Press, 2004.
11. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.
12. S. Goldwasser and S. Micali. Probabilistic encryption, 1984.
13. P. Golle and M. Jakobsson. Reusable anonymous return channels. In *WPES '03: Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, pages 94–100, New York, NY, USA, 2003. ACM Press.
14. B. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 78–86, New York, NY, USA, 1999. ACM Press.
15. M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of USENIX'02*, pages 339–353, 2002.
16. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, September 2004.
17. A. Kiayias and A. Mitrofanova. Testing disjointness of private datasets. In *Proceedings of Financial Cryptography*, volume 3570 of *Lecture Notes in Computer Science*, pages 109–124, 2005.
18. L. Kissner and D. Song. Privacy-preserving set operations. In *Proceedings of Advances in Cryptology - CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, 2005. Full version appears at `http://www.cs.cmu.edu/ leak/`.
19. C.A. Neff. A verifiable secret shuffle and its application to e-voting. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, New York, NY, USA, 2001. ACM Press.
20. W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In *Proceedings of ICICS '97*, volume 1334 of *Lecture Notes in Computer Science*, pages 440–444, 1997.

21. C. O'Keefe, M. Yung, L. Gu, and R. Baxter. Privacy-preserving data linkage protocols. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 94–102, New York, NY, USA, 2004. ACM Press.
22. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 573–584, 1999.
23. J. Vaidya and C. Clifton. Privacy - preserving top-k queries. In *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, pages 545–546, 2005.
24. A. Yao. How to generate and exchange secrets. In *Proceedings of FOCS*, pages 162–167, 1986.

Universal Accumulators with Efficient Nonmembership Proofs

Jiangtao Li[1], Ninghui Li[2], and Rui Xue[3]

[1] Intel Corporation
jiangtao.li@intel.com
[2] Purdue University
ninghui@cs.purdue.edu
[3] LOIS, The Institute of Software, CAS
rxue@is.iscas.ac.cn

Abstract. Based on the notion of accumulators, we propose a new cryptographic scheme called universal accumulators. This scheme enables one to commit to a set of values using a short accumulator and to efficiently compute a membership witness of any value that has been accumulated. Unlike traditional accumulators, this scheme also enables one to efficiently compute a nonmembership witness of any value that has not been accumulated. We give a construction for universal accumulators and prove its security based on the strong RSA assumption. We further present a construction for dynamic universal accumulators; this construction allows one to dynamically add and delete inputs with constant computational cost. Our construction directly builds upon Camenisch and Lysyanskaya's dynamic accumulator scheme. Universal accumulators can be seen as an extension to dynamic accumulators with support of nonmembership witness. We also give an efficient zero-knowledge proof protocol for proving that a committed value is not in the accumulator. Our dynamic universal accumulator construction enables efficient membership revocation in an anonymous fashion.

1 Introduction

Accumulators were first introduced by Benaloh and de Mare [4] as a method to condense a set of values into one short accumulator, such that there is a short witness for each value that has been accumulated. In the mean time, it is infeasible to find a witness for a value that has not been accumulated. Barić and Pfitzmann [3] proposed a construction of a collision-resistant accumulator under the strong RSA assumption. Camenisch and Lysyanskaya [10] further proposed a dynamic accumulator in which elements can be efficiently added into or removed from the accumulator. Accumulators have been used in many applications [3,4,10,20], including membership testing, time stamping, authenticated directory, and certificate revocation.

None of the existing accumulator schemes provide nonmembership witnesses for values that have not been accumulated. This feature of nonmembership witnesses is highly desirable in many applications. The following are two examples where a nonmembership witness is important.

1. Suppose a credit report agency compiles a list of users who have gone into bankruptcy within the last three years, and it also publishes the accumulator for this list. When Alice applies an auto loan from a bank, the bank wants Alice to prove that she is not in the bankruptcy list. In a similar application, suppose a Certificate Authority (CA) revokes a number of certificates before their expiration dates. A certificate user may need to efficiently prove that her certificate has not been revoked when using her certificate.
2. Suppose the Center for Disease Control and Prevention maintains a list of patients who have a certain infectious disease (e.g., Measles or Cholera). In some applications, a patient needs to prove that she has the disease in order to purchase discounted medicines from the local pharmacy stores. Whereas in other applications, one needs to prove that she does not have the disease.

In this paper, we propose the notion of *universal accumulators*, which enables a trusted group manager to condense a list of values into a short accumulator. For each value in the list, there is a short membership witness; and for each value not in the list, there exists a short nonmembership witness. It is computationally infeasible to find a membership witness for a value that was not accumulated or to find a nonmembership witness for a value that was accumulated. The notion of universal comes from the fact that each possible value in the input domain has a witness (either a membership witness or a nonmembership witness). Using universal accumulators, one can easily solve the problems in the aforementioned applications.

We further propose the notion of *dynamic universal accumulators*, which allow one to dynamically add and delete inputs, such that the the cost of an addition or a deletion is independent of the size of the accumulated set. We construct an efficient dynamic universal accumulator under the strong RSA assumption. Dynamic universal accumulators enable efficient membership revocation: A group manager issues a credential for each group member. The group manager also maintains a credential revocation list using a dynamic universal accumulator. To revoke a member, the group manager simply inserts the serial number of the revoked credential into the revoked list. To prove membership, a valid group member first shows her group credential, then shows that the credential's serial number is not in the revocation list by presenting the nonmembership witness.

We also develop an efficient zero-knowledge proof protocol such that, if a value is stored in a cryptographic commitment, then one can prove that the value is not accumulated in a zero-knowledge fashion. This enables membership revocation in an anonymous setting. To prove membership anonymously, the group member first proves that she has a valid group credential, then proves that the credential's serial number is not in the revocation list.

Note that many applications that require nonmembership proofs (such as ones in Application 1) can be solved using membership-proof techniques. Take the certificates revocation as an example, instead of proving the nonmembership of a revocation list, one can prove the membership of a legitimate user list to show that her certificate has not been revoked. This idea was used by, e.g., Camenisch and Lysyanskaya [10]. However, even though proving membership is efficient in [10], the maintenance overhead of the witness could be very expensive if the user list is huge and frequently-changing. For example, consider a certificate system that has thousands or millions of users. Suppose the list of valid users increases every hour (i.e., each hour there are several new users added into the list). And suppose the list of revoked users is small in size and relatively static (e.g., changes every month). Using the scheme in [10], a legitimate user may have to update her witness every hour, whereas using our scheme, she only need to update her nonmembership witness once a month.

1.1 Our Contribution

The contributions of this paper are as follows.

– We introduce the notion of universal accumulators, which support short witnesses for both membership and nonmembership.
– We construct an efficient dynamic universal accumulator based on the strong RSA assumption. The update of witness in our scheme can be efficiently performed without the help of the trusted group manager. Proofs of membership or nonmembership can be achieved with a constant number of modular exponentiations.
– We give an efficient zero-knowledge proof protocol to prove that a committed value was not accumulated. This enables efficient membership revocation in anonymous credential systems, group signature schemes, and direct anonymous attestation schemes. Universal accumulators may be of interest in other applications as well.

1.2 Organization of This Paper

The rest of this paper is organized as follows. We first give notations and security assumptions in section 2. In section 3, we give a formal definition of universal accumulators and present our construction. In section 4, we present the notion of dynamic universal accumulators and describe the corresponding construction. In section 5, we present a zero-knowledge proof protocol to prove that a committed value has not been accumulated in an accumulator. In section 6, we discuss several applications of dynamic universal accumulators to membership revocations in the anonymous setting, we also compare our solution with other existing membership revocation techniques.We conclude our paper in section 7.

2 Notations and Assumptions

2.1 Notations

We use $\phi(\cdot)$ to denote the Euler totient function. Let $n = pq$ be a RSA modulus, we use $\mathbb{Z}_n^*$ to denote the set of all positive integers that are less than n and relative prime to n. We use QR_n to denote the set of quadratic residues modulo n.

A negligible function, denoted by $\text{neg}(\cdot)$, represents a positive function that vanishes faster than the inverse of any fixed positive polynomial. That is, for every polynomial $p(\cdot)$ and for every large enough integer n, $\text{neg}(n) < 1/p(n)$. If S is a probability space, then the probability assignment $x \leftarrow_R S$ means that an element x is chosen at random according to S. If F is a finite set, then $x \leftarrow_R F$ denotes that x is chosen uniformly from F. If p is a predicate and $S_1, S_2, \ldots, S_m$ are probability spaces, then the notation $\texttt{Pr}\left[x_1 \leftarrow_R S_1, x_2 \leftarrow_R S_2, \ldots x_m \leftarrow_R S_m : p(x_1, x_2, \cdots, x_m)\right]$ denotes the probability that $p(x_1, \cdots, x_m)$ will be true after the ordered execution of the probabilistic assignments $x_1 \leftarrow_R S_1, \ldots, x_m \leftarrow_R S_m$. Let A and B be interactive Turing machines, we use $(a \leftarrow A(\cdot) \leftrightarrow B(\cdot) \rightarrow b)$ to denote that a and b are two random variables corresponding to the outputs of A and B as a result of their joint computation.

We use the notation used by Camenisch and Stadler in [14] for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$\texttt{PK}\{(\alpha, \beta) : y = g^{\alpha} h^{\beta} \wedge (u \leq \alpha \leq v)\}$$

denotes a zero-knowledge proof of knowledge of integers α and β, such that $y = g^{\alpha} h^{\beta}$ holds and $u \leq \alpha \leq v$.

2.2 Security Assumptions

The security of our construction is based on the strong RSA assumption, which assumes that it is infeasible to solve the following problem: Given an RSA modulus n and a random $x \leftarrow_R \mathbb{Z}_n^*$, find $e > 1$ and $y \in \mathbb{Z}_n^*$ such that $y^e = x \bmod n$. The strong RSA was introduced by Barić and Pfitzmann [3] and has been used in proving the security of many cryptographic schemes (e.g., [19,18,16]). It can be formally stated as follows:

Assumption 1 (strong RSA assumption). For every probabilistic polynomial-time algorithms A,

$$\texttt{Pr}\left[n \leftarrow G(1^k), x \leftarrow_R \mathbb{Z}_n, (y, e) \leftarrow A(n, x) : y^e = x \pmod{n} \wedge 1 < e < n\right] = neg(k)$$

where $G(1^k)$ is a algorithm that generates a RSA modulus n of size k, and $neg(k)$ is a negligible function.

Our security proofs also use the following lemma ([23,16]):

Lemma 1. *For any integer n, given integers $u, v \in \mathbb{Z}_n^*$ and $a, b \in \mathbb{Z}$, such that $u^a = v^b \bmod n$ and $\gcd(a, b) = 1$, one can efficiently compute $x \in \mathbb{Z}_n^*$ such that $x^a = v \bmod n$.*

To see the correctness of this lemma, observe that, as $\gcd(a, b) = 1$, one can use the extended Euclidian algorithm to find $c, d \in \mathbb{Z}$ such that $bd = 1 + ac$. Let $x = (u^d v^{-c} \bmod n)$, then

$$x^a = u^{ad} v^{-ac} = (u^a)^d v^{-ac} = (v^b)^d v^{-ac} = v \pmod{n}.$$

3 Universal Accumulators

We now give a formal definition of universal accumulators and present our construction.

3.1 Definition of Universal Accumulators

Definition 1. Let k be a security parameter, a secure universal accumulator for a family of input $\{\mathcal{X}_k\}$ is a family of functions $\{\mathcal{F}_k\}$ with the following properties:

- *Efficient Generation*: There is an efficient probabilistic polynomial time algorithm G that on input 1^k produces a random function f of $\mathcal{F}_k$. Additionally, G also outputs some auxiliary information about f, denoted as $\mathtt{aux}_f$.
- *Efficient Evaluation*: Each $f \in \mathcal{F}_k$ is a polynomial time function, on input $(g, x) \in \mathcal{G}_f \times \mathcal{X}_k$, outputs a value $h \in \mathcal{G}_f$, where $\mathcal{G}_f$ is the input domain for the function f, and $\mathcal{X}_k$ is the input domain for the elements to be accumulated.
- *Quasi-Commutative*: For all $f \in \mathcal{F}_k$, for all $g \in \mathcal{G}_f$, and for all $x_1, x_2 \in \mathcal{X}_k$, $f(f(g, x_1), x_2) = f(f(g, x_2), x_1)$. If $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}_k$, we use $f(g, X)$ to denote $f(f(\ldots(g, x_1), \ldots), x_m)$.
- *Membership Witness*: For each $f \in \mathcal{F}_k$, there is a membership verification function ρ_1. Let $c \in \mathcal{G}_f$ and $x \in \mathcal{X}_k$. A value w_1 is called membership witness if $\rho_1(c, x, w_1) = 1$.
- *Nonmembership Witness*: For each $f \in \mathcal{F}_k$, there is a nonmembership verification function ρ_2. Let $c \in \mathcal{G}_f$ and $x \in \mathcal{X}_k$. A value w_2 is called nonmembership witness if $\rho_2(c, x, w_2) = 1$.
- *Security*: A universal accumulator scheme is secure if, for all probabilistic polynomial-time adversary A_k,

  $$\mathtt{Pr}\begin{bmatrix} f \leftarrow G(1^k); g \leftarrow_R \mathcal{G}_f; (x, w_1, w_2, X) \leftarrow A_k(f, \mathcal{G}_f, g) : \\ x \in \mathcal{X}_k; X \subset \mathcal{X}_k; \rho_1(f(g, X), x, w_1) = 1; \rho_2(f(g, X), x, w_2) = 1 \end{bmatrix} = \mathrm{neg}(k)$$

  In other words, it is computationally infeasible to find both a valid membership witness and a valid nonmembership witness for any x in $\mathcal{X}_k$. Note that this is equivalent to say that, given any set $X \in \mathcal{X}_k$, it is computationally infeasible to find $x \in X$ with a valid nonmembership witness or find $x \in \mathcal{X}_k \backslash X$ with a valid membership witness.

The preceding definition is similar to the one of Camenisch and Lysyanskaya [10], the main difference is that our definition requires witnesses for nonmembership elements.

3.2 Our Construction

Our construction builds upon the construction of Camenisch and Lysyanskaya [10]. The difference is that we give an efficient solution for nonmembership witness.

Construction 1 (Universal Accumulators). Let k be a security parameter. Let $\ell = \lfloor k/2 \rfloor - 2$. We use $\mathcal{X}_k$ to denote the set of all primes in $\mathbb{Z}_{2^\ell}$. $\mathcal{X}_k$ is the input domain for the elements to be accumulated.[1] We use $\mathcal{F}_k$ to denote the family of functions corresponding to safe-prime products of length k. The construction takes the following steps.

[1] As in [10], the input domain $\mathcal{X}_k$ has to be primes. If the required input domain is the set of all possible strings, we need to map arbitrary strings to prime numbers. A number of approaches for doing this have been proposed in the literature, see, e.g., [3,19,20].

- The generation algorithm G takes 1^k as input and outputs a random modulus n of length k that is a safe prime, that is, $n = pq$, where $p = 2p' + 1$, $q = 2q' + 1$, p and q have equal length, and p, q, p', q' are all prime number.
- We use f_n to denote the function corresponding to modulus n. The auxiliary information aux_f for f_n is the factorization of n. The input domain $\mathcal{G}_f$ for f_n is defined as $\mathcal{G}_f = \{g \in QR_n : g \neq 1\}$ where QR_n denotes the group of quadratic residues modulo n.
- For $f = f_n$, $f(g, x) = g^x \bmod n$.
- For $f = f_n$, the membership verification function ρ_1 is defined as $\rho_1(c, x, c_x) = 1$ if and only if $(c_x)^x = c$, where $c_x \in \mathcal{G}_f$ is the membership witness for x. The nonmembership verification function ρ_2 is defined as $\rho_2(c, x, a, d) = 1$ if and only if $c^a = d^x g \pmod n$, where $(a, d) \in \mathbb{Z}_{2^\ell} \times \mathcal{G}_f$ is the nonmembership witness for x.

It is easy to see that, given (g, x), we can efficiently compute $f(g, x)$. It is also easy to see that f is quasi-commutative, that is, $f(f(g, x_1), x_2) = f(f(g, x_2), x_1) = g^{x_1 x_2} \bmod n$. Note that membership and nonmembership witnesses can be computed with or without the auxiliary information. It is much more expensive to compute witness without the auxiliary information. In our application, we do not need to compute witness using the auxiliary information; but they may be useful in other settings.

One difference between our construction and Camenisch and Lysyanskaya's accumulator scheme [10] is that their construction does not require to publish g. In our construction, g needs to be public for nonmembership queries. Note that a party that knows g and the value of the accumulator, and suspects that $x_1, \ldots, x_n$ are the values used to compute the accumulator, can easily verify its guess. This is not really a problem because (1) it is not required to hide the accumulated values, and (2) it is easy to prevent this attack by adding a random value x_0 to the set of values used to compute the accumulator.

How to compute witness without the auxiliary information. Suppose $X = \{x_1, \ldots, x_m\}$ is a subset of $\mathcal{X}_k$ and g is a random value in QR_n. Let u denote $\prod_{i=1}^m x_i$. By definition $f(g, X) = g^u \bmod n$. As in the previous accumulator schemes [4,3,10], for any $x \in X$, we can compute the membership witness for x as $c_x = g^{u/x} \bmod n$. To verify the witness, one checks that $x \in \mathcal{X}_k$ and $(c_x)^x = c \bmod n$.

For any $x \in \mathcal{X}_k \backslash X$, since $x, x_1, \ldots, x_m$ are distinct prime numbers, $\gcd(x, u) = 1$. We can find $a \in \mathbb{Z}_{2^\ell}$ and $b \in \mathbb{Z}$ such that $au + bx = 1$. The value a and b can be computed as follows: we first use Euclid algorithm to find a' and b' such that $a'u + b'x = 1$. As x is a positive integer in $\mathbb{Z}_{2^\ell}$, we can always find an integer k such that $a' + kx \in \mathbb{Z}_{2^\ell}$. Observe that $(a' + kx)u + (b' - ku)x = 1$, therefore $a = a' + kx$ and $b = b' - ku$. Let $d = g^{-b} \bmod n$, the nonmembership witness for x is (a, d). To verify the witness, one checks that $x \in \mathcal{X}_k$, $a \in \mathbb{Z}_{2^\ell}$, and $c^a = d^x g \pmod n$, which holds because $c^a = g^{ua} = g^{1-bx} = g^{-bx} g = d^x g \pmod n$.

How to compute witness with the auxiliary information. The membership witness and nonmembership witness can be computed efficiently given the auxiliary information aux_f. Suppose there is a trusted group manager who knows aux_f, maintains the set X, and has already computed the accumulator $c = f(g, X)$, the group manager can compute (non)membership witness for any $x \in \mathcal{X}_k$ with *one* short modular exponentiation.

For $x \in X$, the group manager first checks whether $x \in X$, then computes $a = x^{-1} \mod \phi(n)$, and finally computes $c_x = c^a \mod n$. The membership witness for x is c_x. It is easy to verify the correctness of the witness as $(c_x)^x = (c^a)^x = c^{x^{-1} \cdot x \bmod \phi(n)} = c \pmod n$.

For $x \in \mathcal{X}_k \backslash X$, let $u' = u \mod \phi(n)$, the group manager first checks whether $\gcd(x, u') = 1$.

- If $\gcd(x, u') = 1$, the group manager finds a and b such that $au' + bx = 1$, and sets the nonmembership witness for x as $(a, g^{-b} \mod n)$. The nonmembership witness is correct because $c^a = (g^u)^a = (g^{u'})^a = g^{u'a} = g^{1-bx} = g^{-bx}g = d^x g \pmod n$.
- If $\gcd(x, u') \neq 1$, the group manager finds a and b such that $au + bx = 1$, then computes $b' = b \mod \phi(n)$, and sets the nonmembership witness for x as $(a, g^{-b'} \mod n)$. The nonmembership witness is correct because $c^a = g^{ua} = g^{1-bx} = (g^{-b})^x g = (g^{-b'})^x g = d^x g \pmod n$.

Observe that, the second case is slightly more expensive than the first case. The reason is that, in the second case, the group manager needs to find a and b such that $au + bx = 1$ where u could potentially be large, i.e., size linear to m. Yet, in either case, the exponent in the modular exponentiation computed by the group manager is smaller than $\phi(n)$. Thus the nonmembership witness can be calculated efficiently. Also observe that, the number of $x \in \mathcal{X}_k$ such that $\gcd(x, u') \neq 1$ is less than k, as x must be a prime.

Note that computing witness using auxiliary information may not apply to all scenarios. In some applications, it is not allowed to reveal the auxiliary information to the party who computes the accumulator, since the auxiliary data enables her to prove arbitrary statements. In the case when the party who computes the accumulator is trusted, it is acceptable to give her the auxiliary information.

Theorem 1. *Under the strong RSA assumption, the above construction is a secure universal accumulator.*

Proof. We assume all the arithmetic operations in this proof are modulo n unless specified otherwise. The strong RSA assumption says that given a RSA modulus n and a random value $g \leftarrow_R QR_n$, it is computationally infeasible to find x and y such that $x > 1$ and $y^x = g$.

Suppose there exists a polynomial time adversary $\mathcal{A}$, which on input n and $g \in QR_n$, outputs $c_x \in \mathcal{G}_f$, $d \in \mathcal{G}_f$, $x \in \mathcal{X}_\ell$, $a \in \mathbb{Z}_{2^\ell}$, and $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}_\ell$, such that $c = g^{x_1 \cdots x_m}$, $(c_x)^x = c$, and $c^a = d^x g$. We can construct an algorithm $\mathcal{B}$ to break the strong RSA assumption by invoking $\mathcal{A}$.

Let u to denote $\prod_{i=1}^m x_i$. We consider two cases. In the first case, assume $x \in X$, the adversary can compute u, a, d, and x, such that $c = g^u$ and $c^a = d^x g$. That is, the adversary computes u, a, d, and x such that $g^{au-1} = d^x$. Because $x \in X$, $x \mid u$, and $\gcd(au - 1, x) = 1$. By lemma [1], we can efficiently find y such that $y^x = g$.

In the second case, assume $x \notin X$, the adversary can compute u, c_x, and x, such that $c = g^u$ and $(c_x)^x = c$. In other words, the adversary can find u, c_x, and x such that $g^u = (c_x)^x$. As $x_1, \ldots, x_m$ are all prime, and $x \notin X$, clearly $\gcd(x, u) = 1$. By lemma 1, we can efficiently find y such that $y^x = g$.

We now construct an efficient algorithm $\mathcal{B}$ that breaks the strong RSA assumption as follows. Given a RSA modulus n and $g \leftarrow_R QR_n$, $\mathcal{B}$ invokes $\mathcal{A}$ with input n and g, and obtains outputs c_x, d, x, a, X from $\mathcal{A}$. By the preceding arguments, $\mathcal{B}$ can efficiently compute y from c_x, d, x, a, X such that $y^x = d$, which contradicts to the strong RSA assumption. □

Corollary 1. *In the above construction, for any $f \in \mathcal{F}_k$ and any given set $X \subset \mathcal{X}_f$, it is computationally infeasible to find $x \in X$ with a valid nonmembership witness.*

Proof. This follows directly from Theorem 1. □

Note that, in our security definition of the universal accumulator, we limit the adversary to choose x only from $\mathcal{X}_k$. It is acceptable because when a user proves membership or nonmembership to a verifier, the verifier can first check whether $x \in \mathcal{X}_k$, if not, the verifier can reject the proof. If a user can prove that a value x was not accumulated in an accumulator in an anonymous fashion (see Section 5), Corollary 1 guarantees that x is not a member of the accumulated set X.

4 Dynamic Universal Accumulators

Camenisch and Lysyanskaya [10] proposed the concept of dynamic accumulators in which one can dynamically add and delete elements. In this section, we first give the definition of dynamic universal accumulators, then present a dynamic universal accumulator based on our construction in the previous subsection.

4.1 Definition of Dynamic Universal Accumulators

Definition 2. A universal accumulator is *dynamic* if it has the following properties:

- *Efficient Update of Accumulator* There exists an efficient algorithm D such that, suppose $c = f(g, X)$, if $\hat{x} \notin X$, then $D(c, \hat{x}) = \hat{c}$ such that $\hat{c} = f(g, X \cup \{\hat{x}\})$; if $\hat{x} \in X$, then $D(\mathsf{aux}_f, c, \hat{x}) = \hat{c}$ such that $\hat{c} = f(g, X \backslash \{\hat{x}\})$.
- *Efficient Update of Membership Witness* Let c and $\hat{c}$ be the original and updated accumulators respectively and $\hat{x}$ be the new updated element. There exists an efficient algorithm W_1 such that, if $x \neq \hat{x}$, $x \in X$, and $\rho_1(c, x, w) = 1$, then $W_1(w, c, \hat{c}, x, \hat{x}) = \hat{w}$ such that $\rho_1(\hat{c}, x, \hat{w}) = 1$.
- *Efficient Update of Nonmembership Witness* Let c and $\hat{c}$ be the original and updated accumulators respectively and $\hat{x}$ be the new updated element. There exists an efficient algorithm W_2 such that, if $x \neq \hat{x}$, $x \notin X$, and $\rho_2(c, x, w) = 1$, then $W_2(w, c, \hat{c}, x, \hat{x}) = \hat{w}$ such that $\rho_2(\hat{c}, x, \hat{w}) = 1$.

Of course, it is easy to perform updates using computations that are linear in the size of the accumulated set X, i.e., compute the witnesses from the scratch. In the above definition, the term "efficient" means that the time complexity of each update operation is independent of the size of X. Note that, update of a membership witness or a nonmembership witness is achieved without the auxiliary information. That is, given the original and new accumulators, one can update its witness locally. This is a very useful feature: Suppose the group manager updates the set X, computes the new accumulator, and broadcasts its value to all users. Each user can update her witness *locally* without any help from the group manager.

In terms of security requirement, loosely speaking, a dynamic universal accumulator is secure against an adaptive adversary if the adversary cannot win the following game. Suppose a group manager sets up the function f and the value g and hides the auxiliary information aux_f. The adversary adaptively modifies the set X. Whenever a value x is inserted into or deleted from X, the manager calls algorithm D and publishes the updated accumulator. In the end, the adversary outputs $\hat{x} \notin X$ and a valid membership witness for $\hat{x}$ or outputs $\hat{x} \in X$ and a valid nonmembership witness for $\hat{x}$. The formal security definition of dynamic universal accumulator is stated as follows.

Definition 3. Let $\{\mathcal{F}_k\}$ be the family of universal accumulator functions defined in Definition 1. Let M be an interactive Turing machine that receives input (f, aux_f, g), where $f \in \mathcal{F}_k$, aux_f is the auxiliary information about f, and $g \in \mathcal{G}_f$. M maintains a list of values X which is initially empty. The initial accumulator c is set to be g. M responds to two types of messages: for message (add, x), it makes sure that $x \in \mathcal{X}_k$, adds x to the set X, modifies c by running D, and then sends back the updated c; for message (delete, x), it checks that $x \in X$, deletes it from the set X, updates c by running D, and sends back the updated c. In the end, M outputs the current values for X and c. A dynamic universal accumulator scheme is *secure* if, for all probabilistic polynomial-time adversary A_k,

$$\mathtt{Pr}\begin{bmatrix} f \leftarrow G(1^k); g \leftarrow_R \mathcal{G}_f; \\ (x, w_1, w_2, X) \leftarrow A_k(f, \mathcal{G}_f, g) \leftrightarrow M(f, \mathsf{aux}_f, g) \rightarrow (X, c) : \\ x \in \mathcal{X}_k; X \subset \mathcal{X}_k; c = f(g, X); \rho_1(c, x, w_1) = 1; \rho_2(c, x, w_2) = 1 \end{bmatrix} = \mathrm{neg}(k)$$

We now show that if a secure universal accumulator is dynamic under Definition 2, then this dynamic universal accumulator is secure against adaptive adversaries.

Remark 1. A dynamic universal accumulator is secure against an adaptive adversary if the underlying universal accumulator is secure.

The above remark is straight-forward using reduction argument. We can show that if an adversary $\mathcal{A}$ breaks the security property in Definition 3, we could build another adversary $\mathcal{B}$ to break the security property of a universal accumulator in Definition 1 by invoking $\mathcal{A}$. On input $(f, \mathcal{G}_f, g)$, $\mathcal{B}$ passes these values to $\mathcal{A}$. Because $\mathcal{A}$ needs to interacts with the manager M for updating elements, we let $\mathcal{B}$ act as the manager: if $\mathcal{A}$ sends an (add, x) query, $\mathcal{B}$ simply inserts x into X and computes $c = f(g, X)$; if $\mathcal{A}$ sends a (delete, x) query, $\mathcal{B}$ removes x from X and computes $c = f(g, X)$. In the end, if $\mathcal{A}$ outputs an element $x \in \mathcal{X}_k$ with a valid membership witness w_1 and a valid nonmembership witness w_2, $\mathcal{B}$ outputs (x, X, w_1, w_2). Clearly, $\mathcal{B}$ breaks the security property in Definition 1.

4.2 Our Construction

Construction 2 (Dynamic Universal Accumulators). Our construction is built on Construction 1 with the following additional functionalities:

- *Update of Accumulator:* Adding a value $\hat{x}$ to the accumulator c can be computed as $\hat{c} = c^{\hat{x}} \bmod n$. Deleting a value $\hat{x}$ from the accumulator is computed as $\hat{c} = D(\phi(n), c, \hat{x}) = c^{\hat{x}^{-1} \bmod \phi(n)} \bmod n$, where $\phi(n)$ is the auxiliary information.
- *Update of Membership Witness:* Let w be the original membership witness of x. Let c and $\hat{c}$ be the original and new accumulators, respectively. This construction is the same as the one in [10].

  1. *Addition.* Suppose $\hat{x}$ has been added, the new membership witness can be computed as $\hat{w} = f(w, \hat{x}) = w^{\hat{x}} \bmod n$. It is easy to verify that $\rho_1(\hat{c}, x, \hat{w}) = 1$.
  2. *Deletion.* Suppose $\hat{x} \neq x$ has been deleted, the new membership witness $\hat{w}$ can be computed as follows. Algorithm W_1 chooses two integer a and b such that $ax + b\hat{x} = 1$ and then $\hat{w} = w^b \hat{c}^a \bmod n$. We can verify that:

$$\hat{w}^x = (w^b \hat{c}^a)^x = ((w^b \hat{c}^a)^{x\hat{x}})^{1/\hat{x}} = (c^{b\hat{x}} c^{ax})^{1/\hat{x}} = \hat{c} \pmod n$$

- *Update of Nonmembership Witness:* Let (a, d) be the original nonmembership witness of x.

  1. *Addition.* Suppose $\hat{x} \neq x$ has been added, given $c, \hat{c}, x, \hat{x}$ such that $\hat{c} = c^{\hat{x}} \bmod n$, the new nonmembership witness $(\hat{a}, \hat{d})$ can be computed as follows. Algorithm W_2 first finds two integers $\hat{a}_0$ and r_0 such that $\hat{a}_0\hat{x} + r_0 x = 1$. It is easy to find such $\hat{a}_0$ and r_0 because $\hat{x}$ and x are distinct primes. Multiplying by a to both side of the above equation, we have $\hat{a}_0 a\hat{x} + r_0 ax = a$. W_2 then computes $\hat{a} = \hat{a}_0 a \bmod x$, and find $r \in \mathbb{Z}$ such that $\hat{a}\hat{x} = a + rx$. Note that $\hat{a} \in \mathbb{Z}_{2^\ell}$. In the end, W_2 computes $\hat{d} = dc^r \bmod n$. We can verify $\rho_2(\hat{c}, x, \hat{a}, \hat{d}) = 1$, or, $\hat{c}^{\hat{a}} = \hat{d}^x g$ holds:

$$\hat{c}^{\hat{a}} = c^{\hat{a}\hat{x}} = c^{a+rx} = c^{rx} c^a = c^{rx} d^x g = (dc^r)^x g = \hat{d}^x g \pmod n$$

  2. *Deletion.* Suppose $\hat{x}$ has been deleted, given $c, \hat{c}, x, \hat{x}$ such that $c = \hat{c}^{\hat{x}} \bmod n$, the new nonmembership witness $(\hat{a}, \hat{d})$ can be computed as follows. Algorithm W_2 chooses an integer r such that $a\hat{x} - rx \in \mathbb{Z}_{2^\ell}$ (there always exists such r because $x \in \mathbb{Z}^*_{2^\ell}$), then let $\hat{a} = a\hat{x} - rx$ and $\hat{d} = d\hat{c}^{-r} \bmod n$. We can verify $\rho_2(\hat{c}, x, \hat{a}, \hat{d}) = 1$, or, in other words, $\hat{c}^{\hat{a}} = \hat{d}^x g$ holds:

$$\hat{c}^{\hat{a}} = \hat{c}^{a\hat{x}-rx} = c^a \hat{c}^{-rx} = d^x g \hat{c}^{-rx} = (d(\hat{c})^{-r})^x g = \hat{d}^x g \pmod n$$

Note that we can add or delete several values together simply by letting $\hat{x}$ be the product of the added or deleted values. This is also true for updating (non)membership witness.

5 Efficient Proof That a Committed Value Was Not Accumulated

We now present a useful building block for certificate revocation in an anonymous setting – a protocol that proves a committed value was not accumulated in the accumulator. Suppose that a group manager compiles a list of revoked users and publishes the accumulator for the set of revoked serial numbers[2]. If a regular user wants to prove that she is not in the revocation list, she simply shows her serial number and the corresponding nonmembership witness. However, such approach reveals the user's serial number (thus the identity as well). The building block presented in this section enables such nonmembership proof in an anonymous fashion, i.e., without revealing the serial number. The idea here is that the user first commits her serial number in her certificate, then proves that the committed serial number was not accumulated in the revocation list. We shall describe in details how this building block is used for certificate and membership revocation in the anonymous setting in the next section.

The commitment scheme that we use in this section is developed by Fujisaki and Okamoto [18] and improved by Damgård and Fujisaki [17]. The parameters of the this commitment scheme are (n_1, g_1, h_1), where n_1 is a special RSA modulus of length k_1, h_1 is a random value in QR_{n_1}, and g_1 is a random value in the group generated by h_1. To commit a value x, the committer chooses a random $r \leftarrow_R \mathbb{Z}_{n_1}$ and computes the commitment $\mathsf{commit}(x, r) = g_1^x h_1^r \bmod n_1$. The Fujisaki and Okamoto's commitment scheme is statistically hiding and computationally binding if factoring is hard. Note that the protocol described next could also work for other commitment schemes, such as the Pedersen commitment [22], with only minor modifications.

For our protocol, we require an element h in QR_n such that $\log_g h$ is unknown to the prover, where g and n are the parameters of the universal accumulators described in the previous sections. To prove that given a commitment c_1 and an accumulator c, the value committed in c_1 has not been accumulated in c, we build the following zero-knowledge proof protocol. The common inputs to the protocol are $c_1, n_1, g_1, h_1, c, n, g$, and h. The prover has additional inputs: x, r, a, d such that $c_1 = g_1^x h_1^r \bmod n_1$ and $c^a = d^x g \bmod n$, where the first equation shows that x is the committed value of c_1 and the second equation shows that x was not accumulated in c.

Protocol 1. $\mathtt{PK}\{(x, r, a, d) : c_1 = g_1^x h_1^r \ \wedge\ c^a = d^x g \ \wedge\ x \in \mathbb{Z}_{2^\ell} \ \wedge\ a \in \mathbb{Z}_{2^\ell}\}$

1. The prover chooses, uniformly at random, values w, r_x, r_a, r_w, r_z, and r_e of length k. The prover computes the following values (modulo n): $c_x = g^x h^{r_x}$, $c_a = g^a h^{r_a}$, $c_d = d g^w$, $c_w = g^w h^{r_w}$, $z = xw$, $c_z = g^z h^{r_z}$, and $c_e = (c_d)^x h^{r_e}$. The prover sends $(c_x, c_a, c_d, c_w, c_z, c_e)$ to the verifier and carry out the following zero-knowledge proofs of knowledge.
   Note that $c_e = (c_d)^x h^{r_e} = (dg^w)^x h^{r_e} = d^x g^{xw} h^{r_e} = g^{-1} c^a g^z h^{r_e}$.
2. The prover proves to the verifier that the value committed in c_1 in bases (g_1, h_1) is the same as the value committed in c_x in bases (g, h):

$$\mathtt{PK}\{(\varepsilon, \rho, \rho_x) : c_1 = g_1^\varepsilon h_1^\rho \bmod n_1 \ \wedge\ c_x = g^\varepsilon h^{\rho_x} \bmod n\}$$

[2] We assume that each group member in the system has a unique prime serial number.

3. The prover proves to the verifier that the value committed in c_e in bases (c_d, h) is the same as the value committed in c_x:

$$\mathtt{PK}\{(\varepsilon, \rho_e, \rho_x) : c_e = (c_d)^{\varepsilon} h^{\rho_e} \ \wedge\ c_x = g^{\varepsilon} h^{\rho_x}\}$$

4. The prover proves to the verifier that $c_e g$ is also a commitment in bases $((c, g), h)$, and the values committed in $c_e g$ are the same as the values committed in c_a, c_z, and the power of h in $c_e g$ is the same as in c_e in bases (c_d, h):

$$\mathtt{PK}\{(\sigma, \tau, \varepsilon, \rho_a, \rho_z, \rho_e) : \\ c_e g = c^{\sigma} g^{\tau} h^{\rho_e} \ \wedge\ c_a = g^{\sigma} h^{\rho_a} \ \wedge\ c_z = g^{\tau} h^{\rho_z} \ \wedge\ c_e = (c_d)^{\varepsilon} h^{\rho_e}\}$$

5. The prover proves to the verifier that c_z is a commitment to the product of values committed in c_x and c_w:

$$\mathtt{PK}\{(\sigma, \tau, \varepsilon, \rho_z, \rho_w, \rho_x, \rho) : \\ c_z = g^{\tau} h^{\rho_z} \ \wedge\ c_w = g^{\sigma} h^{\rho_w} \ \wedge\ c_x = g^{\varepsilon} h^{\rho_x} \ \wedge\ c_z = (c_w)^{\varepsilon} h^{\rho}\}$$

6. The prover proves to the verifier that c_x is a commitment to an integer of length ℓ, and that c_a is a commitment to an integer of length ℓ:

$$\mathtt{PK}\{(\varepsilon, \sigma, \rho_x, \rho_a) : c_x = g^{\varepsilon} h^{\rho_x} \ \wedge\ c_a = g^{\sigma} h^{\rho_a} \ \wedge\ \varepsilon \in \mathbb{Z}_{2^{\ell}} \ \wedge\ \sigma \in \mathbb{Z}_{2^{\ell}}\}$$

Note that the preceding protocol is similar to the one proposed by Camenisch and Lysyanskaya [11]. The zero-knowledge proof protocol in [11] is used to prove knowledge of a signature, whereas our protocol is to prove knowledge of a nonmembership witness. The details of the zero-knowledge proof protocols in each step are omitted, as these zero-knowledge proof protocols are standard in the literature, e.g., a protocol for proving knowledge of equality of representation modulo two composite modulus in step 2 , 3, and 4 can be found in [13], a protocol for zero-knowledge proof that a committed value is the product of two other committed values in step 5 can be found in [12,17], a protocol for proving that a committed value lies in a given range in step 6 can be found in [6].

Theorem 2. *The preceding protocol is a zero-knowledge proof of knowledge of the values (x, r, a, d) such that $c_1 = g_1^x h_1^r \bmod n_1$ and (a, d) is a valid nonmembership witness of x for accumulator c.*

Proof. The completeness property of Protocol 1 is obvious. The zero-knowledge property of Protocol 1 is also clear. The simulator first computes the commitments c_x, c_a, c_d, c_w, c_z, and c_e at random. Then the simulator invokes the simulator for the zero-knowledge proofs of knowledge of each step. Because the commitments reveal nothing statistically and the proofs of knowledge protocols at each step are zero-knowledge, the preceding protocol is zero-knowledge.

We now show that there exists a knowledge extractor that outputs a valid committed value x and a valid nonmembership witness for x. Our extractor will invoke the extractor for the zero-knowledge proof protocols at each step as a building block. If our extractor fails, then we are able to set up a reduction to break the strong RSA assumption. Suppose the extractor succeeds and computes $(x, a, w, z, r, r_x, r_a, r_w, r_z, r_e)$ such that

$$c_1 = g_1^x h_1^r \tag{1}$$
$$c_x = g^x h^{r_x} \tag{2}$$
$$c_e = (c_d)^x h^{r_e} \tag{3}$$
$$c_a = g^a h^{r_a} \tag{4}$$
$$c_z = g^z h^{r_z} \tag{5}$$
$$c_e g = c^a g^z h^{r_e} \tag{6}$$
$$c_z = g^{xw} h^{r_z} \tag{7}$$

where the equations (1) and (2) come from the extractor in step 2 of the protocol, the equations (2) and (3) come from the extractor in step 3 of the protocol, the equation (4), (5), and (6) come from the extractor in step 4 of the protocol, the equation (5) and (7) come from the extractor in step 5 of the protocol. From equations (5) and (7), we get $z = xw$. Equations (3) and (6) imply that $(c_d)^x g = c^a g^z$. Let $d = c_d/g^w$, we have $d^x = (c_d/g^w)^x = c_d^x/g^z = c^a g^{-1}$, or equivalently, $c^a = d^x g$. Since we also know that x and a are of length ℓ, we can output (x, r, a, d) such that c_1 is a commitment of x, and (a, d) is a valid nonmembership witness for x. □

6 Application to Certificate and Membership Revocation

In this section, we show that the dynamic universal accumulator we constructed can be used for efficient membership revocation for group signatures, anonymous credentials, and direct anonymous attestation schemes; right after a brief review of these schemes.

6.1 Review of Group Signatures, Anonymous Credentials, and Direct Anonymous Attestation

Group signatures, first introduced by Chaum and van Heyst [15], provide anonymity for signers. In a group signature scheme, each group member can sign messages such that the resulting signatures do not reveal the identity of the signer. A number of group signature schemes have been proposed, e.g., [1,2,5,14]. Formally speaking, a group-signature scheme with membership revocation is a digital signature scheme comprised of the following procedures:

- *Setup:* On input a security parameter, this probabilistic algorithm outputs the initial group public key and the secret key for the group manager.
- *Join:* A protocol between the group manager and a user that results in the user becoming a new group member. The user's output is a membership certificate and a membership secret.
- *Sign:* A probabilistic algorithm that on input a group public key, a membership certificate, a membership secret, and a message m outputs group signature of m.

- *Verify:* An algorithm for establishing the validity of a group signature of a message with respect to a group public key.
- *Open:* An algorithm that, given a message, a valid group signature on it, a group public key and a group managers secret key, determines the identity of the signer.
- *Revocation:* An algorithm for the group manager to remove a member from the group. This algorithm results in an updated group's public key and some other information to be stored in a public server.
- *Membership Update:* An algorithm for the users to update their membership certificates and membership secrets using the information available in the public server and the current group public key.

A secure group signature scheme must satisfy the *anonymity* and *unlinkability* property. The anonymity property says that, given a valid signature of some message, identifying the actual signer is computationally hard for everyone but the group manager. The unlinkability property means that deciding whether two different valid signatures were computed by the same group member is computationally hard.

In an anonymous credential system [21,9,11], a user can demonstrate to a verifier that she has a credential, but the verifier cannot infer anything about who the user is other than the fact that the user has the right credential. The Camenisch and Lysyanskaya [9,11] credential system has a similar construction to the group signature schemes. Essentially, their system enables a credential holder to prove to a verifier that the credential holder has a signature signed by the certificate authority.

Direct Anonymous Attestation (DAA) was first proposed by Brickell, Camenisch, and Chen [7]. DAA enables remote authentication of a Trust Platform Module (TPM), while preserving the privacy of the user of the platform that contains the module. The DAA scheme can be seen as a group signature without the feature that a signature can be opened. The DAA scheme presented in [7] is similar to the signature scheme proposed in [11].

6.2 Incorporating Revocation into Group Signature Schemes

In this subsection, we use the group signatures scheme developed by Ateniese et al. [1] as an example, and show that our universal accumulator scheme can be integrated into the group signatures scheme to enable efficient revocation.

- *Setup:* In [1], the group manager chooses two ranges Γ and Λ and chooses $n = pq$ where p and q are safe primes. The group manager also picks $a, a_0, g, h \in QR_n$, and chooses a secret elements x and computes $y = g^x \bmod n$. The group public key is (n, a, a_0, g, h, y) and the secret key is (p, q, x). In addition, the group manager creates n' for the universal accumulators described in section 3, such that $\Gamma \subseteq \mathcal{X}_k$. The group manager also chooses a random $g' \in QR_{n'}$ and publishes (n', g') as the public parameters of the universal accumulator.
- *Join:* In [1], a user interacts with the group manager. In the end, the user obtains $e_i \in \Gamma$, $x_i \in \Lambda$, and A_i such that $a^{x_i} a_0 \equiv A_i^{e_i} \bmod n$. In addition, given the current revocation list $\{e_1, \ldots, e_m\}$, the group manager computes the nonmembership witness for e_i, and sends the witness to the user.

- *Sign and Verify:* In [1], the prover with private key (A_i, e_i, x_i) proves to the verifier that she is a member of the group. More specifically, the prover uses zero-knowledge proof of knowledge to prove the knowledge of (A_i, e_i, x_i) such that $a^{x_i} a_0 \equiv A_i^{e_i} \mod n$. In addition, the prover proves to the verifier that e_i is not in the revocation list. This can be done using the zero-knowledge proof protocol in Section 5. That is, the prover can first commit e_i and then prove that (1) the value committed is the same as the value in her private key, and (2) the value committed has not been accumulated in $\{e_1, \ldots, e_m\}$.
- *Revocation:* To revoke a member with private key (A_i, e_i, x_i) from the group, the group manager inserts e_i into the revocation list. Let c be the current accumulator. The group member updates $c = f(c, e_i) = c^{e_i} \mod n'$.
- *Membership Update:* Let $\hat{c}$ be the current accumulator and c be previous accumulator stored in the public server. Each group member updates her nonmembership witness accordingly using $\hat{c}$ and c using the algorithms presented in Section 4.

Analogously, we could integrate our revocation scheme using universal accumulators with other group signature schemes [2,5,14], anonymous credentials schemes [9,11], compact e-cash scheme [8], and DAA scheme [7] with minor modifications. Observe that the group manager could generate both a valid membership witness and a valid nonmembership witness for a given group member. In our scheme, we assume that the group manager is trusted and it will not generate nonmembership witnesses for the members that are already in the revocation list.

7 Conclusion

We proposed a new cryptographic scheme called universal accumulators which enables one to condense to a set of values using a short accumulator, to efficiently compute a witness of the membership of any value that has been accumulated, and to efficiently compute a witness of the nonmembership of any value that has not been accumulated. We gave a construction for universal accumulators and proved its security based on the strong RSA assumption. We then presented a construction for dynamic universal accumulators in which one can add (or delete) inputs into (or from) the accumulated set with constant cost. Dynamic universal accumulators can be used for efficient membership revocation in the anonymous setting. Universal accumulators may be of independent interest in other applications as well.

Acknowledgement

Most of this work was done while Jiangtao Li and Rui Xue were at Purdue University. This work is supported by NSF IIS-0430274, NSF CCR-0325951, and sponsors of CERIAS. Rui Xue is also supported by the Fund of the China Scholarship Council and the NSFC Grant 60373048. We thank the anonymous reviewers for their helpful comments.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology – CRYPTO '00*, pages 255–270, 2000.
2. G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In *Proceedings of Financial Cryptography*, pages 183–197, 2001.
3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology – EUROCRYPT '97*, pages 480–494, 1997.
4. J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology – EUROCRYPT '93*, pages 274–285, 1994.
5. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS)*, pages 168–177, 2004.
6. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – EUROCRYPT '00*, pages 431–444, May 2000.
7. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS)*, pages 132–145, 2004.
8. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Advances in Cryptology – EUROCRYPT '05*, pages 302–321, 2005.
9. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology – EUROCRYPT '01*, pages 93–118, 2001.
10. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology – CRYPTO '02*, pages 61–76, 2002.
11. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of the 3rd Conference on Security in Communication Networks*, pages 268–289, 2002.
12. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *In Advances in Cryptology – EUROCRYPT '99*, pages 106–121, 1999.
13. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *In Advances in Cryptology – CRYPTO '99*, pages 413–430, 1999.
14. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO '97*, pages 410–424, 1997.
15. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT '91*, pages 257–265, Apr. 1991.
16. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS)*, pages 46–51, Nov. 1999.
17. I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. In *Advances in Cryptology: ASIACRYPT '02*, pages 125–142. Springer, 2002.
18. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – CRYPTO '97*, pages 16–30, Aug. 1997.
19. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT '99*, pages 123–139, 1999.

20. M. T. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Proceedings of the 5th International Conference on Information Security*, pages 372–388, 2002.
21. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography, 6th Annual International Workshop, SAC '99*, pages 184–199, 1999.
22. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*, pages 129–140, 1992.
23. A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems*, 1(1):38, Feb. 1983.

Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes

Stanisław Jarecki and Xiaomin Liu

University of California, Irvine

Abstract. We present the first practical *unlinkable secret handshake* scheme. An unlinkable secret handshake is a two-way authentication protocol in a PKI setting which protects privacy and anonymity of *all* information about the participants to *everyone* except of their intended authentication partners. Namely, if entity A certified by organization CA^A wants to authenticate itself only to other entities certified by CA^A, and, symmetrically, entity B certified by CA^B wants to authenticate itself only to entities also certified by CA^B, then a secret handshake protocol authenticates these parties and establishes a fresh shared key between them if and only if $CA^A = CA^B$ and the two parties entered valid certificates for this CA into the protocol. If, however $CA^A \neq CA^B$, or $CA^A = CA^B$ but either A or B is not certified by this CA, the secret handshake protocol reveals *no information* to the participants except of the bare fact that their inputs do not match. In other words, an Unlinkable Secret Handshake scheme is a perfectly private authentication method in the PKI setting: One can establish authenticated communication with parties that possess the credentials required by one's policy, and at the same time one's affiliation *and* identity remain perfectly secret to everyone except of the parties to whom one wants to authenticate.

Efficient secret handshake schemes, i.e. authentication protocols which protect the privacy of participants' affiliations, were proposed before, but participants in these schemes remained *linkable*. Namely, an attacker could recognize all the instances of the protocol executed by the same entity. Secondly, the previous schemes surrendered user's privacy if the certificates of this user were revoked, and our scheme alleviates this problem as well. Unlinkable schemes were proposed as well, but they either relied on single-use certificates, or did not support revocation, or required instantaneous propagation of revocation information.

Crucial ingredients in our construction of unlinkable secret handshakes are chosen-ciphertext secure key-private encryption and multi-encryption schemes, and the first efficient construction of a key-private group key management scheme, which is a stateful analogue of (key-private) public key broadcast encryption.

1 Introduction

Privacy of Authentication in the PKI Model. "Unlinkable Secret Handshake" is a name we give to an Authenticated Key Exchange [AKE] scheme which, in addition to the standard security properties needed of an AKE scheme provides privacy properties of *affiliation hiding*, *policy hiding*, and *unlinkability*. In order to explain these privacy properties we need to recall how authentication (and authenticated key exchange) works in the public-key infrastructure [PKI] model. In the PKI model each party holds a certificate *cert* on its public key issued by some *Certification Authority* [CA], the fact which we denote as $cert \in CA$. (In the general PKI model, certificates can be put together in arbitrarily long chains, but here we consider only a "flattened PKI model" which does not support certificate chains.) We call the CA an *affiliation* of that party and we use the terms "certified by" and "affiliated with" interchangeably. We refer to the parties affiliated with a given CA as its *group*, and we call the CA a *group manager*.

In addition to a list of certificates, each party also holds an authentication *policy*, represented by a list of CA's, which specifies that this party wants to establish authenticated communication only with entities affiliated with these CA's. In some applications parties affiliated with some CA might have a policy to authenticate only to other entities affiliated with the same CA, e.g. with the employees of the same company, or with the members of the same organization. In a common case a player's policy will *include* all the CA's this player is affiliated with, but in general the two lists might have nothing in common.

Two-sided authentication in the PKI model is successful depending on whether there is a match between the affiliations and the policies of the two interacting parties. For simplicity of discussion, let each player hold only a single certificate, and let its authentication policy consist also of only a single "target" CA. In a secure PKI-based authentication scheme, if party A enters into the computation a certificate $cert^A$ and policy CA^A, and party B enters certificate $cert^B$ and policy CA^B, both players accept (and output a fresh authenticated session key) only if $cert^A \in CA^B$ and $cert^B \in CA^A$.

The standard way in which players A and B discover if their affiliations and policies match is to announce for one party to announce its affiliation and policies to the other. This match-discovery process is then followed by the cryptographic protocol, consisting of the players' verifying each other's certificates and running an AKE protocol on the public keys in these certificates. However, this means that some player's affiliations and/or policies are effectively available to anyone who requests it. This unrestricted leakage of authentication policy and/or affiliation information of a party is a privacy threat, because in many applications both one's affiliation and one's policy is a sensitive information that should be protected from unnecessary exposure. (Note also that most commonly one's policy immediately reveals one's affiliation.) The only entities that have the right to know that player A is affiliated with CA_1 and by policy wants to communicate securely with entities affiliated with CA_2, are entities that are indeed affiliated with CA_2. No one else *needs* to know anything about A's affiliation and policy. In fact, an un-authorized observer should not be able to *link* any two instances of the AKE protocol executed by any party. Note that linkability has been recognized as a privacy threat in the context of many applications, and this motivated research into identity escrow [KP98], electronic-cash, e.g. [CFN88], or unlinkable credentials [CL01], among other applications.

In contrast, an *(Unlinkable) Secret Handshake* [SH] scheme is an AKE protocol with the following privacy property: An adversary playing the role of player B, who does not hold a valid certificate $cert^B \in CA^A$ where CA^A is a policy specified by A in this AKE protocol instance, does not learn *anything* about party A. More specifically, the protocol provides (1) *affiliation/policy hiding*, in the sense that without authorization one cannot tell the affiliation and the authentication policy of any party, and (2) *unlinkability*, in the sense that an un-authorized adversary cannot link any two instances of the SH protocol executed by the same player. (Of course, B's privacy against malicious player A is protected in the analogous way.) Note that it makes no sense to require that affiliation/policy hiding holds against "insiders", i.e. parties which *do* satisfy A's policy. However, in some applications it might make sense to require unlinkability from insiders, and even though we do not model such insider-unlinkability formally, our scheme can be modified to support it, although the on-line computation cost would then grow from $2 \log n$ to $O(\Delta \log n)$ exponentiations.

Prior Work on Secret Handshakes. *Linkable* versions of a Secret Handshake scheme were given before, based on bilinear maps [BDS+03, BHS04], computational Diffie-Hellman [CJT04], or RSA [JKT07a, JKT07b]. (As shown in [JKT07b], the RSA-based secret handshake scheme proposed in [Ver05] is insecure because it fails to protect players' affiliations.) All these solutions are efficient and practical, but all of them display two privacy vulnerabilities: First, even though the affiliations and policies of the participants in these schemes are protected in the sense of affiliation/policy hiding, these schemes do not meet the unlinkability property. In fact, instances of the SH protocol executed by a single party can be efficiently linked by any observer. Secondly, these schemes do not protect affiliation privacy of players whose certificates need to be revoked, e.g. in case of key corruption or loss. There have been several proposed solutions to the unlinkable secret handshake problem, but neither of them solves the problem in a satisfactory way. First, all the above schemes have trivially unlinkable variants if players use single-use certificates, but such single-use certificates require too much storage and make revocation impractical. The scheme of Tsudik and Xu [TX05] relies on a group secret shared by all the group members, and thus it requires perfect synchrony in revocation information between the participating players, or otherwise the players fail to authenticate one another. The scheme of Xu and Yung, [XY04] is not based on any shared secrets, but it only offers a weak version of the privacy property called *k-anonymity.* This notion allows the attacker to learn that the participants' affiliation is contained in the set of k publicly revealed CA's. Moreover, since the real affiliation of player A belongs to an intersection of the k-element sets released each time A runs the protocol, protocol instances can be linked with significant probability. Finally, [AB07] proposed an unlinkable scheme but their scheme does support revocation.

Our Contributions: **(1)** The primary contribution of this paper is the first construction of an efficient *unlinkable* Secret Handshake scheme, with no information leakage due to certificate revocation, with no reliance on single-use certificates, with support of revocation, and without the requirement that the both players assume the same revocation information. The scheme takes only a few communication rounds and its bandwidth is $\Delta \cdot \log n$ standard public key encryption ciphertexts, where n is the upper bound on the number of group members and Δ is a parameter equal to the maximum tolerated *lag* between revocation updates received by the two protocol participants. The computational costs consist of $\Delta \cdot \log n$ off-line exponentiations and $2 \log n$ on-line exponentiations, if the scheme is instantiated with the least expensive CCA-secure variant of ElGamal encryption, e.g. DHAES [ABR01]. For practical values like $\Delta = 10$ and $n \leq 2^{16}$, the bandwidth comes to $20KB$, assuming computational Diffie-Hellman holds on groups of residues of 1024-bit primes, and the on-line computation involves 32 (short) exponentiations with a fixed base, i.e. less than 100 milliseconds.

(2) We provide very strong definitions of both security and privacy for an SH scheme. Security is modeled as in a standard AKE protocol (e.g. [CK02]), and hence our definition implies security against the man-in-the-middle attack, and it offers independence between keys on every session, thus neutralizing session-interference attacks. (This was unknown for the protocols given in [BDS$^+$03, BHS04, CJT04].) Similar AKE-based definition for secret handshakes was given for 2-party secret handshake protocols in [JKT07b], but here we *strengthen* the privacy property so that it includes *unlinkability* of the protocol instances in addition to *affiliation and policy hiding* modeled in [JKT07b].

(3) The main ingredient in our solution is a construction of a *key-private* public-key group key management [PKGKM], which is a stateful version of the public-key broadcast encryption. We show an efficient key-private PKGKM scheme based on the "Logical Key Hierarchy" GKM scheme of Wallner et al. [WHA97], and we show a CCA-secure version of it. Key-privacy for standard encryption has been recognized as an important tool for achieving anonymity and privacy properties in various protocols. A key-private (stateful) *broadcast* encryption might find such applications as well. As a side contribution, we extend results of Bellare et al. [BBs03] on batched encryption, to ElGamal encryption compiled into its CCA-secure version via the Fujisaki-Okamoto construction [FO99].

Note on Intrinsic Limitations of Affiliation/Policy Hiding AKE's. The security notion for AKEs implies complete independence between protocol sessions in the sense that a key agreed on any chosen session remains secure regardless of what happens to keys agreed on all other sessions, including the most extreme case when all these other keys are simply revealed in the clear. In the works on secure AKE's, *e.g.* [CK02], this is modeled by giving the adversary against an AKE scheme an access to a "key revelation" oracle which can reveal keys computed on any protocol session except the session that the adversary is attacking. In particular, this implies that any AKE session remains secure even if the adversary sees whether or not all other protocol sessions were successful or not. (Adversary can do that by revealing the keys computed in these sessions and testing whether they are non-empty and equal.) In contrast to security, the *privacy* property of affiliation/policy hiding *cannot* be achieved if (1) the propagation of a revocation list is not instantaneous, (2) an adversary can engage with any player in a protocol session, and (3) the adversary can observe whether or not any of these sessions were successful. This is because an active adversary who corrupts some player can find out the affiliation/policy (assume these are equal) of any player A who has not yet updated his or her revocation list. Namely, an adversary can engage in a protocol with A assuming the corrupted player's identity, and since A's revocation list is not updated, the protocol succeeds, and the adversary observes that and learns that A is affiliated with the same group as the corrupted player. The adversary then arranges a session between A and any other player B, e.g. by engaging both players and acting as a man in the middle, and if the adversary observes whether the session was successful, he/she can conclude if B's affiliation/policy matches that of A's. Note that this attack can be staged even if player B always has the most recent revocation list, either because the protocol is supposed to tolerate the lag in A's and B's revocation lists *or* because A might have updated his revocation list before running the protocol with B.

Therefore, an AKE scheme *cannot* protect the affiliation/policy hiding of even the players who always immediately update their revocation lists if (1) there are players who do not, (2) the adversary can engage any player in the protocol, and (3) the adversary can observe if a session is successful. Given this intrinsic limitation to privacy of authentication protocols, we believe that the most useful relaxation that would enable privacy protection in practical applications is to remove the third item. In other words, we must require that the adversary cannot tell an execution of a successful protocol that is executed over the secure channel established by an instance of the AKE scheme, from an execution of a "simulation" of such protocol, which a player will perform whenever an AKE instance fails. In other words, the AKE scheme can offer affiliation/policy privacy but only for a special class of protocols which utilize the keys agreed-upon by this AKE, namely for protocols that can be simulated in this fashion. This class includes, for example, protocols which have fixed number of rounds and whose message sizes can be fixed without big efficiency losses. The privacy-preserving execution of such protocol involves padding every message to the fixed upper bound, and its simulation consists of sending random messages of the same size. In both cases the protocol messages need to be encrypted with key-private symmetric encryption (which standard symmetric encryption schemes provide), but in the simulation the key is chosen at random. We defer the full specification of such privacy-preserving protocols to the full version [JL07].

Organization. We define key-privacy for PKGKM's in Section 2, and Unlinkable SH's in Section 3. In Section 4 we introduce multi-encryption, which we use to build key-private GKM in Section 5, and from that we build an SH scheme in Section 6. All proofs have been delegated to the full version of this paper [JL07].

2 Key-Private Group Key Management: Definition

We describe the syntax of a (Public-Key) Group Key Management [PKGKM] scheme, and we define two PKGKM properties: (1) Semantic security under the chosen-ciphertext attack, IND-CCA, which is an adaptation of the standard security notion of IND-CCA for standard encryption [GM84] to GKM schemes; and (2) Key-privacy under CCA, IK-CCA, which is also an adaptation of the IK-CCA notion of key-privacy introduced by Bellare et al. [BBDP01] for standard encryption.

In a Public-Key GKM scheme we consider a group of players administered by a *group manager*, who creates a *public* (encryption) key, issues private (decryption) keys to the group members, and can revoke any member by broadcasting a revocation information, which is used to update both the public and the private keys. A PKGKM scheme is a stateful version of a Public-Key Broadcast Encryption scheme, considered e.g. by Dodis and Fazio [DF02] and by Boneh et al. [BBW06]. In a PK BE scheme, the public and private keys are fixed, and the encryptor can encrypt a message for *any subset* of players. In a PKGKM scheme, messages are always encrypted under the most recent public key, and the revocation information used in computing this key determines the subset of players who can decrypt.

A PKGKM scheme is a tuple of algorithms (Setup, KGen, $\mathsf{Revoke}(\mathsf{PKUpdate}$, $\mathsf{SKUpdate})$, Enc, Dec):

- $\mathsf{Setup}(1^k)$, on input a security parameter k, generates parameters params.
- $\mathsf{KGen}(\mathsf{params})$, executed by the group manager, generates the initial group public key $PK^{(0)}$, the initial master secret $\mathsf{MSK}^{(0)}$, and members' initial private keys $SK_1^{(0)}, ..., SK_n^{(0)}$.
- $\mathsf{Revoke}(\mathsf{MSK}^{(t)}, i, t)$, executed by the group manager in epoch t (initially $t = 0$) revokes key SK_i of member U_i by generating an update message $U^{(t+1)}$, and updating the master secret to $\mathsf{MSK}^{(t+1)}$. Message $U^{(t+1)}$ is used to update the public key as $PK^{(t+1)} = \mathsf{PKUpdate}(PK^{(t)}, U^{(t+1)})$, and each decryption key as ${SK_j}^{(t+1)} = \mathsf{SKUpdate}({SK_j}^{(t)}, U^{(t+1)})$, for each $j \neq i$.
- $\mathsf{Enc}(PK^{(\mathsf{et})}, m)$ encrypts message m on public key $PK^{(\mathsf{et})}$.
- $\mathsf{Dec}({SK_i}^{(\mathsf{dt})}, C)$ is a decryption algorithm which either returns some message m or rejects.

Initialization of the Static Adversary: To express security properties of the PKGKM scheme in the *static* adversary model, it's convenient to denote the initialization pattern for the static adversary who corrupts subset Rev of players, on public parameters params, as $\mathsf{Init}(\mathsf{params}, \mathsf{Rev})$. This initialization involves an execution of $\mathsf{KGen}(\mathsf{params})$ and t executions of Revoke, for t from 1 to $\tau = |\mathsf{Rev}|$, which generates the initial public/private keys and t update messages $U^{(1)}, \ldots, U^{(\tau)}$, which in turn define a set of public/private keys $PK^{(t)}$ and ${SK_i}^{(t)}$, for all $i \notin \mathsf{Rev}$ and $0 \leq t \leq \tau$. The static adversary receives params, the public keys, and the private keys of the players in Rev. In the key-privacy definitions, when we initialize the keys of two groups G_0, G_1, on the same public parameters params, we'll generate the two sets of keys for these two groups as $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_0, 0)$ and $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_1, 1)$.

Δ-Limited Completeness. We stress that we do *not* assume that the key update messages are propagated immediately to all the participants, and hence there

can be a discrepancy between the epoch of the public key used by the encryptor and the epoch of the private key used by the decryptor. The PKGKM scheme we construct in this paper handles such discrepancy between the epochs up to some value Δ, and otherwise it does not guarantee proper decryption. In practice the group members will have to retrieve update messages in an anonymous way, e.g. using onion-routing, often enough to offset this Δ-limitation on the tolerated epoch difference.

Let $\mathsf{params} = \mathsf{Setup}(1^k)$ and all the private/public keys are generated via $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}, \epsilon)$. Let $\mathsf{Rev}(t)$ denotes the first t indices in Rev. We call a PKGKM scheme *Δ-Limited ϵ_{comp}-Complete* if it holds with probability at least $1 - \epsilon_{comp}$ that for any message m, for any $0 \leq \mathsf{et}, \mathsf{dt} \leq \tau$ and $i \notin \mathsf{Rev}(max\{\mathsf{et}, \mathsf{dt}\})$, if $|\mathsf{et} - \mathsf{dt}| \leq \Delta$ then

$$\mathsf{Dec}(SK_i^{(\mathsf{dt})}, C) = m \quad \text{if} \quad C = \mathsf{Enc}(PK^{(\mathsf{et})}, m)$$

IND-CCA Security and IK-CCA Privacy for PKGKM's. We define the IND/IK-CCA security notions for a PKGKM scheme only for *static adversaries*. Both notions are analogous to the IND-CCA security notion and the IK-CCA key-privacy notion for standard public key schemes. The differences in the IND-CCA game for a PKGKM scheme proceeds are as follows: (1) The game proceeds on some set Rev of corrupted players chosen before the game starts; (2) The adversary is given the initial group public key together with $\tau = |\mathsf{Rev}|$ update messages and the private keys of the players in Rev, as generated by $\mathsf{Init}(\mathsf{params}, \mathsf{Rev})$; (3) The queries to the decryption oracle the adversary can make are of the form (C, i, t), where $i \notin \mathsf{Rev}$ and $t \leq \tau$, and the decryption oracle responds with $\mathsf{Dec}(SK_i^{(t)}, C)$; (4) The encryption challenge, on adversarially chosen pair of messages (m_0, m_1), is computed as $C^* = \mathsf{Enc}(PK^{(\tau)}, m_b)$; and (5) After receiving C^* the adversary cannot make decryption queries of the form (C^*, i, t) for any i, t. As in the standard IND-CCA notion, we say that the PKGKM scheme is IND-CCA if the probability that the adversary guesses b is at most negligibly larger than 1/2.

The IK-CCA game for a PKGKM scheme is defined in a similar way, except that the adversary is given two sets of private/public keys, for groups G_0 and G_1, the adversary can access decryption oracles for both groups, the encryption challenge is computed as $C^* = \mathsf{Enc}({PK_b}^{(\tau)}, m)$ on adversarially chosen message m, and after receiving C^* the adversary cannot ask for decryption of C^* to either group. Again, the PKGKM scheme is IK-CCA if the probability that the adversary guesses b is at most negligibly larger than 1/2. We provide both definitions in the full version of this paper on eprint [JL07].

3 Unlinkable Secret Handshake Scheme: Definition

Similarly as in the Group Key Management setting, the model for a Secret Handshake scheme consists of a set of groups $\mathcal{G}$, each managed by its manager, and a set of users $\mathcal{U}$. For notational convenience in describing the security *model* of a Secret Handshakes scheme, we'll assume that every user $U \in \mathcal{U}$ is a member of a unique group $G \in \mathcal{G}$, which we'll denote $G = \mathsf{Membership}(U)$. We say that

$U \in G$ if $G = \mathsf{Membership}(U)$. Each $U \in G$ has its unique index in this group, denoted $i = \mathsf{Index}(U)$. We stress that the restriction that each user is a member of a unique group is taken only for notational convenience in defining the security and privacy properties of a Secret Handshake scheme. It is *not* a restriction on the actual applications of such scheme, where the same user can indeed be a member of many groups.

Syntax of a Secret Handshake Scheme. We define a *Secret Handshake* (SH) scheme as a tuple of algorithms (Setup, KGen, Revoke($\mathsf{PKUpdate}$, $\mathsf{SKUpdate}$), $\mathsf{Handshake}$), where all algorithms except $\mathsf{Handshake}$ have the same syntax as in a GKM scheme. As in the GKM scheme, we'll denote by Init the *static* initialization pattern involving an execution of KGen followed by $|\mathsf{Rev}|$ instances of Revoke, one per each element in the set Rev of revoked players, chosen beforehand.

The new procedure, $\mathsf{Handshake}$, is an interactive protocol executed by any user U, on public inputs params and private inputs (SK, TPK, r), where $SK = SK_{G,i}{}^{(\mathsf{dt})}$ for $G = \mathsf{Membership}(U)$, $i = \mathsf{Index}(U)$, dt is U's current epoch for group G, $TPK = PK_{TG}{}^{(\mathsf{et})}$ is a public key of some "target" group TG, not necessarily equal to G, at some epoch et, and r is the *role* of U in this execution, which is equal either to init for *initializer* or resp for *responder*. Since it's an interactive protocol, it is intended that two users, U and U', execute two *matching* instances of the $\mathsf{Handshake}$ protocol and exchange the messages generated on these protocol instances. Adopting the standard terminology for Key Agreement protocols, we refer to every instance of $\mathsf{Handshake}$ protocol as a *session*, and we'll call the matching instances of this protocol *matching sessions*. We will denote an instance of the protocol executed by θ-th instance of user U as Π_U^θ. Consider the session instance Π_U^θ running on inputs (SK, TPK, r), and instance $\Pi_{U'}^{\theta'}$ running on inputs (SK', TPK', r'). We call these two sessions *matching* if (1) $r \neq r'$, (2) $\mathsf{Membership}(U) = \mathsf{PKGroup}(TPK')$, and (3) $\mathsf{Membership}(U') = \mathsf{PKGroup}(TPK)$, where $\mathsf{PKGroup}(PK)$ identifies the group of the public key. We call two matching instances Π_U^θ and $\Pi_{U'}^{\theta'}$ *partnered* if the protocol transcript on these two sessions are the same, i.e. if the messages sent by Π_U^θ are delivered to $\Pi_{U'}^{\theta'}$ and vice versa. Two partnered sessions should both accept and output the same authenticated and random key K, which can be then used for any subsequent secure communication. If a session instance does not output a key then it rejects.

Δ-Limited Completeness: Informally, we require that if U and U' run matching instances of the $\mathsf{Handshake}$ protocol, i.e. U is a member of group G and wants to authenticate itself to members of group TG, while U' is a member of group $G' = TG$ and wants to authenticate itself to members of group $TG' = G$, then both players accept and output the same key on this session. However, we can only guarantee such completeness property if neither of the two players is revoked from their respective groups and if the epochs of the public and private keys these players use are no farther than Δ apart, for both groups.

Formally, set $\mathsf{params} = \mathsf{Setup}(1^k)$, and initialize two groups $G = 0$ and $G = 1$ on two sets $\mathsf{Rev}_0, \mathsf{Rev}_1 \subset \{1, ..., n\}$ with the static initialization pattern $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_0, 0)$ and $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_1, 1)$. Let $\mathsf{Rev}_G(t)$ denote the first

t indices in Rev_G and let $\tau_G = |\mathsf{Rev}_G(\tau)|$. We call an SH scheme *$\Delta$-Limited ϵ-Complete* if the following holds: For any $0 \leq \mathsf{et}_G, \mathsf{dt}_G \leq \tau_G$ s.t. $|\mathsf{et}_G - \mathsf{dt}_G| \leq \Delta$, and any $i_G \notin \mathsf{Rev}_G(max\{\mathsf{et}_G,\ \mathsf{dt}_G\})$, for both $G = 0$ and $G = 1$, if U and U' run two sessions of the Handshake protocol on inputs

$$U\text{'s inputs: } (SK_{0,\,i_0}^{(\mathsf{dt}_0)}, PK_1^{(\mathsf{et}_1)}, \mathsf{init}) \qquad U'\text{'s inputs: } (SK_{1,\,i_1}^{(\mathsf{dt}_1)}, PK_0^{(\mathsf{et}_0)}, \mathsf{resp})$$

then if all the protocol messages are properly exchanged between these two instances, then with probability $1 - \epsilon$ they both output a common key K.

Security of a Secret Handshake Scheme: Denote a set of groups as $\mathcal{G}$, a set of users $\mathcal{U}$ partitioned between these groups via the Membership function, and for each group $G \in \mathcal{G}$ some set $\mathsf{Rev}_G \subset \{1, ..., n\}$. We define SH security via a game between an adversary $\mathcal{A}$ and the challenger simulating a network of players, on common input $(1^k, \mathcal{U}, \mathcal{G}, \{\mathsf{Rev}_G\}_{G \in \mathcal{G}})$. First, the challenger picks $\mathsf{params} \leftarrow \mathsf{Setup}(1^k)$ and initializes each group $G \in \mathcal{G}$ by the static initialization procedure $\mathsf{Init}(\mathsf{params}, Rev_G, G)$, which produces all the public/private keys and gives $\mathcal{A}$ all the public keys, update messages, and the private keys of the revoked members in each group. After that, $\mathcal{A}$ adaptively issues to the challenger any number of commands of the following type, and outputs a single bit when it's done:

- [Handshake, U, dt, PK_{TG}^{et}, r]: The challenger initializes an instance Π_U^θ where θ is the next index that has not yet been used for player U, and then starts the protocol for user U in epoch dt, i.e. runs Handshake on inputs $(SK_{G,i}^{(\mathsf{dt})}, PK_{TG}^{\mathsf{et}}, r)$ where $G = \mathsf{Membership}(U)$, $i = \mathsf{Index}(U)$, $\mathsf{dt} \leq |\mathsf{Rev}_G|$, and $\mathsf{et} \leq |\mathsf{Rev}_{TG}|$. The challenger keeps the state of the instance, and hands to the adversary any message it generates.
- [Message, U, θ, m]: The challenger wakes up the Π_U^θ instance of the Handshake protocol on message m, if such instance exists, and follows the protocol on behalf of that instance. If the instance outputs another message, the challenger hands it to the adversary. If the instance rejects, the challenger abandons it. If the instance outputs a key, the challenger records it by saving the tuple $(\Pi_U^\theta, PK_{TG}^{\mathsf{et}}, K)$.
- [Reveal, U, θ]: If either the Π_U^θ session or some session partnered with Π_U^θ was *tested* (see below), or if session Π_U^θ did not output a key, the challenger returns $\perp$. Otherwise, the challenger returns the key K output on this session to $\mathcal{A}$.
- [Test, U, θ]: The challenger picks a random bit b. If this is the only Test query $\mathcal{A}$ makes, if session Π_U^θ outputted some key K, if the adversary has not revealed the key on either Π_U^θ *or* some session $\Pi_{U'}^{\theta'}$ partnered with Π_U^θ, and if Π_U^θ executed on the target public key PK_{TG}^{et} s.t. $\mathsf{et} = |\mathsf{Rev}_{TG}|$ (i.e. if the target public key PK on that session is updated so that all adversary's keys in group TG are revoked), then the challenger returns K to $\mathcal{A}$ if $b = 0$, or a random string of length $|K|$ if $b = 1$. If any of these conditions are not met, the challenger returns $\perp$.

Let $\mathsf{AdvSH}_{\mathcal{A},k}$ denote the probability that $\mathcal{A}$ outputs the correct bit b chosen by the challenger. (If $\mathcal{A}$ never makes the Test query we pick b at random.) We say that a SH scheme is $(t, \epsilon, n, n', R, q_S)$-secure if for $|\mathcal{U}| = n$, $|\mathcal{G}| = n'$, for maximum R sessions per user, for all subsets $\{\mathsf{Rev}_G\}_{G \in \mathcal{G}}$, and all algorithms $\mathcal{A}$ running in time t which invoke a total of q_S SH protocol instances, we have $|\mathsf{AdvSH}_{\mathcal{A},k} - \frac{1}{2}| < \epsilon$.

Unlinkability and Affiliation/Policy-Hiding of a Secret Handshake Scheme: We define unlinkability and affiliation /policy-hiding similarly to SH-security, via a game between an adversary $\mathcal{A}$ and a challenger on common input $(1^k, \mathcal{U}, \mathcal{G})$ and the specification of corrupt (and revoked) players $\{\mathsf{Rev}_G\}_{G \in \mathcal{G}}$). As in the SH-security game, the challenger picks $\mathsf{params} \leftarrow \mathsf{Setup}(1^k)$ and initializes each group by $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_G, G)$. After the initialization, the challenger picks a random bit $b \in \{0, 1\}$. Since our AKE protocols protect privacy only for the privacy-preserving protocols (see the note in the introduction), we model the privacy adversary without access to a key-revealing oracle. In the full version of the paper [JL07] we give a full model where the adversary has an access to a protocol-execution oracle instead, and privacy of an AKE scheme holds only if the protocol is privacy-preserving and compiled using a privacy simulator. In these proceedings we simply restrict the privacy adversary $\mathcal{A}$ to issuing only the $\mathsf{Handshake}$ and $\mathsf{Message}$ commands to the challenger.

The challenger in this privacy game services $\mathcal{A}$'s commands depending on bit b the challenger chooses: If $b = 0$ then the challenger executes each command by following the corresponding protocol on behalf of the user entities, as in the SH-security game. However, if $b = 1$, the challenger uses a special interactive machine SIM to serve the commands issued to the instances whose target keys are updated so that all adversary's keys are revoked. For other instances, the challenger follows the protocol on behalf of the user as in the SH-security game. The SIM machine is initialized on string params, it can keep state between invocations, but has no access to group keys created by the challenger in all the Init instances. The challenger executes as follows using SIM:

- [$\mathsf{Handshake}$, U, dt, PK^{et}_{TG}, r]: If the target key PK^{et}_{TG} satisfies $\mathsf{et} = |\mathsf{Rev}_{TG}|$ (i.e. if it is updated so that all adversary's keys in group TG are revoked), then the challenger picks a next index θ that has not been used yet by U and an additional globally unique (random) string $\hat{\theta}$, which we will call an *identifier* for the Π^{θ}_U session. The challenger hands $[\mathsf{Handshake}, \hat{\theta}, \hat{\theta}\text{-}\mathsf{list}]$ to SIM where $\hat{\theta}$-list contains the identifiers of all sessions which match Π^{θ}_U.

  If $\mathsf{et} < |\mathsf{Rev}_{TG}|$, the challenger initializes an instance Π^{θ}_U for θ the next index not used by U, runs the $\mathsf{Handshake}$ protocol on behalf of Π^{θ}_U as in the SH-security game, and hands any message generated by Π^{θ}_U to the adversary.
- [$\mathsf{Message}$, U, θ, m]: If [$\mathsf{Handshake}$, U, dt, PK^{et}_{TG}, r] has been issued and $\mathsf{et} = |\mathsf{Rev}_{TG}|$, the challenger retrieves $\hat{\theta}$ identifier for this session, passes $[\mathsf{Message}, \hat{\theta}, m]$ to SIM, and forwards SIM's answer to $\mathcal{A}$. If $\mathsf{et} < |\mathsf{Rev}_{TG}|$, the challenger follows the real protocol on behalf of Π^{θ}_U as in the SH-security game, and hands any message generated by Π^{θ}_U to the adversary.

Let $\mathsf{AdvSH}_{\mathcal{A},k}$ denote the probability that $\mathcal{A}$ outputs the correct bit b chosen by the challenger in the above game. We say that a SH scheme is $(t, \epsilon, n, n', q_S)$-*unlinkable and affiliation/policy hiding* if there exists a simulator algorithm SIM running in time polynomial in k s.t. for $|\mathcal{U}| = n$, $|\mathcal{G}| = n'$, for R setting the maximum number of sessions per user, for all subsets $\{\mathsf{Rev}_G\}_{G \in \mathcal{G}}$, and all algorithms $\mathcal{A}$ running in time t which invoke a total of q_S SH protocol instances, we have $|\mathsf{AdvSH}_{\mathcal{A},k} - \frac{1}{2}| < \epsilon$.

4 Security and Key-Privacy for Batched Encryption

Our construction of a key-private (Public-Key) Group Key Management scheme (PKGKM), is based on the so-called “multi-encryption” [ME] introduced by Bellare et al. [BBs03]. A multi-encryption is a non-standard method of encrypting a message under many independent public keys, where the encryptor uses the *same randomness* when encrypting the message under each key. The reason we rely on multi-encryption instead of standard encryption in our key-private group key management scheme construction, is that it reduces the cost of decryption procedure from $O(N)$ exponentiations to $O(1)$, where N is the number of component ciphertexts.

Here is why: In our key-private (public-key) group key management scheme [PKGKM], the ciphertext is a vector of ciphertexts. We cannot tag any information that links each component to the public key, as that would leak information about the group. As a consequence, the decrypting party will not know which of these component ciphertexts it should decrypt, and hence it can decrypt the PKGKM ciphertext only in an *oblivious* way, i.e. by attempting to decrypt each of the component (standard) ciphertexts. Now, if the N component ciphertexts are computed in a standard way, such oblivious decryption would take $O(N)$ exponentiations. However, if the component ciphertexts are computed using the same randomness vector, then the decryption procedure requires only $O(1)$ exponentiations and $O(N)$ fast symmetric operations, e.g. xors and tests for equality.

In the rest of this section we (1) define a multi-encryption [ME] version of a public-key encryption scheme, and the IND-CCA and IK-CCA notions for it. Then we (2) define a version of multi-encryption in which decryption is *oblivious*, and we discuss converting an ME to an oblivious multi-encryption [OME]. Finally, (3) we show examples of IND/IK-CCA encryption schemes which yield IND/IK-CCA ME and OME schemes.

Multi-Encryption and its Security and Privacy Properties. We define multi-encryption as a version of a standard public-key encryption, where the same message is encrypted under a set of public keys, and the encryptor uses the same randomness in each encryption. A standard public-key encryption is given by a triple of algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, but for the purpose of multi-encryption we need to split the key-generation procedure into two parts, Setup, which generates some public parameters params, e.g. $\mathsf{params} = (p, q, g)$ where p, q are primes and g is an element of order q in $\mathbb{Z}_p^*$, and $\mathsf{KGen}(\mathsf{params})$ proper, which for example picks private key x at random in $\mathbb{Z}_q$ and sets the public key as $y = g^x \bmod p$.

The multi-encryption Π^{ME} version of the public key encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{ME}, \mathsf{Dec}_{ME})$, (note that the setup does not change), where:

$\mathsf{KGen}_{ME}(\mathsf{params})$ executes $\mathsf{KGen}(\mathsf{params})$ n times to produce a vector of n public keys $\boldsymbol{pk} = \{pk_i\}_{i=1,\dots,n}$ and the corresponding private keys $\boldsymbol{sk}$.
$\mathsf{Enc}_{ME}(\boldsymbol{pk}, m)$ picks a (long-enough) random string r and outputs a vector of ciphertexts $\boldsymbol{c} = \{c_i\}_{i=1,\dots,n}$ where $c_i = \mathsf{Enc}(pk_i, m; r)$ [i.e. the encryption of m under key pk_i using randomness r].
$\mathsf{Dec}_{ME}(sk_i, \boldsymbol{c})$ outputs $\mathsf{Dec}(sk_i, c_i)$ where $\boldsymbol{c} = (c_1, \dots, c_n)$.

The IND-CCA notion of security and the IK-CCA notion of key-privacy for a multi-encryption scheme Π^{ME} is defined analogously to the IND-CCA security and IK-CCA key-privacy definitions for the underlying standard encryption scheme Π. In the security notion (IND-CCA), the difference between security of multi-encryption and standard encryption is that (1) in the multi-encryption case the adversary receives a vector of public keys $\boldsymbol{pk}$ instead of a single key pk, (2) the challenge ciphertext is a multi-encryption vector $\boldsymbol{c}^* = \mathsf{Enc}_{ME}(\boldsymbol{pk}, m_b)$ for the randomly chosen message m_b instead of a single ciphertext $c^* = \mathsf{Enc}(pk, m_b)$, and (3) the adversary has an adaptive access to a "flexible" decryption oracle, which takes as input the index i of the decryptor and the ciphertext vector $\boldsymbol{c} = \{c_i\}$, and outputs $\mathsf{Dec}(sk_i, c_i)$. Also, after the adversary sees the encryption challenge ciphertext $\boldsymbol{c}^* = (c_1^*, ..., c_n^*)$ the adversary's queries $(i, \boldsymbol{c})$ must satisfy $c_i \neq c_i^*$ where $\boldsymbol{c} = (c_1, ..., c_n)$.

The changes between IK-CCA notion for multi-encryption and the IK-CCA notion for standard encryption are analogous. We note that the above IND-CCA and IK-CCA notions for multi-encryption scheme are for a *static adversary*, which cannot corrupt parties after the protocol starts.

Oblivious Multi-Encryption. An *oblivious* multi-encryption scheme is a multi-encryption scheme as described above, except that the decryptor is not told which ciphertext is directed to him. In other words, an oblivious multi-encryption scheme Π^{OME} is a version of the public key encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{ME}, \mathsf{Dec}_{OME})$, where Setup is as in the underlying standard encryption Π, algorithms KGen_{ME}, Enc_{ME} are as in the multi-encryption scheme Π^{ME} formed from Π as defined above, and the decryption procedure $\mathsf{Dec}_{OME}(sk_i, \boldsymbol{c})$ proceeds differently than $\mathsf{Dec}_{ME}(sk_i, \boldsymbol{c})$: Namely, if $\boldsymbol{c} = (c_1, \dots, c_n)$, the decryption procedure computes $m_j \leftarrow \mathsf{Dec}(sk_i, c_j)$, for each j ranging from 1 to n. If all m_j's are equal to the rejection symbol $\perp$, then $\mathsf{Dec}_{OME}(sk_i, \boldsymbol{c})$ outputs $\perp$ as well. Otherwise, it outputs the first m_j s.t. $m_j \neq \perp$.

The IND-CCA and IK-CCA notions for oblivious multi-encryption scheme Π^{OME} are very similar to those for the (non-oblivious) multi-encryption Π^{ME} (see above), and the only difference is in the adversary's interaction with the (flexible) decryption oracle: First, the decryption oracle implements the Dec_{OME} procedure instead of Dec_{ME}. Second, after getting the encryption challenge $\boldsymbol{c}^* = (c_1^*, \dots, c_n^*)$, the decryption query $(i, \boldsymbol{c})$ made by the adversary must satisfy $\boldsymbol{c} \neq \boldsymbol{c}^*$. (Note that $\boldsymbol{c}$ can contain one or more component ciphertexts of c^*, as long as $\boldsymbol{c} \neq \boldsymbol{c}^*$.) Analogous changes are made for the IK-CCA notion of an oblivious multi-encryption scheme.

Constructing IND+IK-CCA Secure and Complete Oblivious Multi-Encryption Schemes. Consider the following conversion from an OME scheme $\Pi^{OME} = (\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{ME}, \mathsf{Dec}_{OME})$ to another OME scheme $\Pi'^{OME} = (\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}'_{ME}, \mathsf{Dec}'_{OME})$, where $\mathsf{Enc}'_{ME}(\boldsymbol{pk}, m)$ picks random $r \leftarrow \{0,1\}^k$ for the security parameter k, computes $\boldsymbol{c} \leftarrow \mathsf{Enc}_{ME}(\boldsymbol{pk}, (m, r))$ and outputs ciphertext $C = (\boldsymbol{c}, H(\boldsymbol{c}, m, r))$, where $H(\cdot) \rightarrow \{0,1\}^{2k}$ is a hash function (modeled as a random oracle in the security analysis); and $\mathsf{Dec}'_{OME}(sk, C)$ parses ciphertext C as $(\boldsymbol{c}, h)$, and outputs $\hat{m}$ if $(m, r) \leftarrow \mathsf{Dec}_{OME}(sk, \boldsymbol{c})$ s.t. $h = H(\boldsymbol{c}, \hat{m}, \hat{r})$.

Theorem 1. *If Π^{ME} is an IND+IK-CCA ME scheme, then Π'^{OME} is an $(n2^{-k})$-complete IND+IK-CCA OME scheme in ROM.*

Examples of IND+IK-CCA Oblivious Multi-Encryption Schemes. Bellare et al. showed a generic method for converting IND-CCA standard encryption into an IND-CCA ME. It required a technical property of "reproducibility" of the underlying encryption.[1] It's easy to extend their results to IK privacy. I.e., the same reproducibility implies that IK-CCA encryption yields IK-CCA ME. By combining the results of [BBDP01] and [BBs03] with the discussion above, this yields IND/IK-CCA and complete OME from DHAES and Cramer-Shoup. We extend these results in the following sense: Fujisaki and Okamoto showed a way to convert one-way encryption schemes, with additional technical property of γ-uniformity, into IND-CCA encryption schemes, in the ROM model, via a hybrid with symmetric encryption [FO99]. Their main theorem can be easily extended to cover also key privacy. We refer to the full paper [JL07] for details.

5 Key-Private PKGKM from Oblivious Multi-encryption

Our key-private public-key group key management scheme is based on the so-called *Wallner Tree* key distribution scheme proposed by Wallner et al [WHA97] [WGL98], which uses a binary tree to assign subsets of keys to group members. Our key of a node in the tree is a pair of public/private keys $({pk_z}^{(t)}, {sk_z}^{(t)})$ in epoch t. We could encrypt a message under the top key ${pk_\varepsilon}^{(t)}$, as it is known to all members of the group. However, such scheme would work only if the public key used by the encryptor has the same epoch as the private key used by the decryptor, e.g. if both parties have the most recent key-update message.

Such synchrony assumption is not realistic in practice. On the other hand, we can relax this assumption and construct a practical group key management scheme with Δ-limited completeness, i.e. a scheme which works assuming a limit Δ on the discrepancy between the key epochs assumed by the encryptors and decryptors. Our way to use the set of keys is similar to the extension of the

[1] We note that the notion of multi-encryption introduced in [BBs03] is stronger than here. Namely, the messages encrypted for each public key need not be the same.

Wallner Tree construction in which any subset of Δ players can be revoked in a batch. Let $\mathsf{Rev}^{(t)}$ be a set of Δ indices corresponding to Δ most recently revoked users. Let $\mathsf{co\text{-}path}(u)$ be the co-path of user u and $\widetilde{\mathcal{R}}^{(t)}$ be a $\Delta \times \log n$ table of indices whose i-th column is made of indices in set $\mathsf{co\text{-}path}(r_i)$ for i-th element r_i in $\mathsf{Rev}^{(t)}$. In any column i, the element in row j is the j-bit long element in $\mathsf{co\text{-}path}(r_i)$. Let $\mathcal{R}^{(t)}$ be a transformation of the $\widetilde{\mathcal{R}}^{(t)}$ matrix, where every element of $\mathcal{R}^{(t)}$ which is a prefix of some index i in the revoked set $\mathsf{Rev}^{(t)}$ is replaced by a special symbol $\star$. For example, if $n = 16$, $\Delta = 3$, and $Rev^{(t)} = \{000, 011, 101\}$ then

$$\widetilde{\mathcal{R}}^{(t)} = \begin{pmatrix} 1 & 1 & 0 \\ 01 & 00 & 11 \\ 001 & 010 & 100 \\ 0000 & 0111 & 1010 \end{pmatrix} \longrightarrow \quad \mathcal{R}^{(t)} = \begin{pmatrix} \star & \star & \star \\ \star & \star & 11 \\ 001 & 010 & 100 \\ 0000 & 0111 & 1010 \end{pmatrix} \tag{1}$$

In this way, matrix $\mathcal{R}^{(t)}$ consists of nodes which cover all the leaves except those in set $\mathsf{Rev}^{(t)}$. Therefore, if the encryptor encrypted a message m under public keys ${pk_z}^{(t-\Delta)}$ for all indices $z \in \mathcal{R}^{(t)}$, then every node except those in $\mathsf{Rev}^{(t)}$ would be able to get m using its key from epoch $t - \Delta$. Let $PK^{(t)}$ be a $\log n \times \Delta$ table of public keys from epoch $t - \Delta$ corresponding to the indices in $\mathcal{R}^{(t)}$. If some entry in $\mathcal{R}^{(t)}$ is a symbol "$\star$", then the entry in the same position of $PK^{(t)}$ is also a "$\star$". Continuing the above example we have

$$PK^{(t)} = \begin{pmatrix} \star & \star & \star \\ \star & \star & pk_{11}^{(t-3)} \\ pk_{001}^{(t-3)} & pk_{010}^{(t-3)} & pk_{100}^{(t-3)} \\ pk_{0000}^{(t-3)} & pk_{0111}^{(t-3)} & pk_{1010}^{(t-3)} \end{pmatrix} \tag{2}$$

We can then encrypt a message m under $PK^{(t)}$ using an OME scheme, as $\mathcal{C}^{(t)}$ given below. For the entries in $PK^{(t)}$ that are marked "$\star$", the corresponding ciphertexts are filled with randomness, in the format of real ciphertexts.

$$\mathcal{C}^{(t)} = \begin{pmatrix} \$ & \$ & \$ \\ \$ & \$ & Enc_{pk_{11}^{(t-3)}}(m) \\ Enc_{pk_{001}^{(t-3)}}(m) & Enc_{pk_{010}^{(t-3)}}(m) & Enc_{pk_{100}^{(t-3)}}(m) \\ Enc_{pk_{0000}^{(t-3)}}(m) & Enc_{pk_{0111}^{(t-3)}}(m) & Enc_{pk_{1010}^{(t-3)}}(m) \end{pmatrix} \tag{3}$$

Then each user U_i for $i \notin \mathsf{Rev}^{(t)}$ could use its key set of epoch $t - \Delta$, *i.e.* $\{{sk_z}^{(t-\Delta)} : z \in \mathsf{path}(i)\}$, to decrypt the message m. (Actually, $\{{sk_z}^{(t-\Delta)} : z \in \mathsf{path}(i)\}$ remain the same until epoch t.)

However, instead of requiring the decryptor to be Δ epochs behind the encryptor, we want to tolerate any lag between the encryptor and decryptor epochs, et and dt, as long as $|\mathsf{et} - \mathsf{dt}| \leq \Delta$. To do this, we will make each member of the group store its key set for $(\Delta{+}1)$ consecutive epochs. Each player's secret key $SK_i^{(t)}$ is a $\log n \times (\Delta{+}1)$ key table whose j-th column is a key set of user U_i in epoch $\tau = t - (\Delta{+}1){+}j$. The keys in each column are arranged so that ${sk_z}^{(\tau)}$, for $\tau \in \{t-\Delta, t\}$, is in row number $|z|$, *i.e.* the bit-length of index z.[2] For the same example above, the node 0111 has the secret key table:

$$SK_{0111}^{(t)} = \begin{pmatrix} sk_0^{(t-3)} \neq sk_0^{(t-2)} \neq sk_0^{(t-1)} = sk_{01}^{(t)} \\ sk_{01}^{(t-3)} = sk_{01}^{(t-2)} \neq sk_{01}^{(t-1)} = sk_{01}^{(t)} \\ sk_{011}^{(t-3)} = sk_{011}^{(t-2)} \neq sk_{011}^{(t-1)} = sk_{011}^{(t)} \\ sk_{0111}^{(t-3)} = sk_{0111}^{(t-2)} = sk_{0111}^{(t-1)} = sk_{0111}^{(t)} \end{pmatrix} \quad (4)$$

where $\neq$ means the key changed from one epoch to the next one.

Let $(\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{OME}, \mathsf{Dec}_{OME})$ be an IND/IK-CCA secure and complete Oblivious Multi-Encryption scheme. The IND/IK-CCA and Δ-complete PKGKM scheme can be constructed as follows:

– $\mathsf{Setup}(1^k)$: Output $(\mathsf{params}, n, \Delta)$, where $\mathsf{params} \leftarrow \mathsf{Setup}$, n is the maximum number of members in each group (assumed to be a power of 2), and Δ is the maximum difference between the encryptor's epoch and the decryptor's epoch for which we guarantee correctness of decryption.
– $\mathsf{KGen}(\mathsf{params})$: For Wallner Tree of depth $\log n$. Use $\mathsf{KGen}_{ME}(\mathsf{params})$ to generate a set of public/private key pairs $({pk_z}^{(0)}, {sk_z}^{(0)})$ for all tree nodes z. Store all these key pairs as $\mathsf{MSK}^{(0)}$. To simplify the description of the key generation process, we let the group manager revoke Δ dummy members, so that $PK^{(0)}$ has the format as eq.(2), and user U_i's key ${SK_i}^{(0)}$ is a $\log n \times (\Delta{+}1)$ key-table, whose j-th column is filled with keys $\{{sk_z}^{(j-\Delta-1)} \mid z \in \mathsf{path}(i), z \neq \varepsilon\}$, as in eq.(4).
– $\mathsf{Revoke}(\mathsf{MSK}^{(t-1)}, r^{(t)})$: The update message $U^{(t)}$ consists of the standard Wallner Tree update message which revokes user $r^{(t)}$ (and updates both the standard Wallner Tree keys and the keys $pk_z^{(t-1)}, sk_z^{(t-1)}$ for tree-nodes z on the path from the root to the leaf corresponding to player $r^{(t)}$. Additionally, the update message contains also the $PK^{(t)}$ table as in eq.(2). The table contains a set of keys from epoch $t - \Delta$ which is determined by the tree-leaves assigned to the last Δ revoked members.
– $\mathsf{PKUpdate}(U^{(t)})$: Extract $PK^{(t)}$ from $U^{(t)}$.
– $\mathsf{SKUpdate}({SK_i}^{(t-1)}, U^{(t)})$: User U_i extracts the key update part from $U^{(t)}$, perform the Wallner Tree user key update to get the new set of keys of epoch t, i.e. $\{{sk_z}^{(t)}\}$, for z in the key path of U_i. Denote ${\mathbf{sk}_i}^{(t)}$ as the resulting Wallner-tree secret key set for user U_i in epoch t. Then U_i discards ${\mathbf{sk}_i}^{(t-\Delta-1)}$ from the first column of his key table ${SK_i}^{(t-1)}$, and append ${\mathbf{sk}_i}^{(t)}$ to the last column, so that ${SK_i}^{(t)}$ in the format of eq.(4).
– $\mathsf{Enc}(PK^{(t)}, m)$: Compute $C \leftarrow \mathsf{Enc}_{OME}(PK^{(t)}, m)$ and format it as an $\log n \times \Delta$ table, as in eq.(3).

[2] In this way all the keys in U_i's key set are present in column τ except of the group secret ${sk_\varepsilon}^{(\tau)}$. The users could use these group secrets too, but this makes the description of the scheme slightly more complicated, and it does not improve the performance by much.

– $\mathsf{Dec}({SK_i}^{(t)}, C)$: For each key sk in the left-most column ${\mathbf{sk}_i}^{(t-\Delta)}$ in U_i's key table ${SK_i}^{(t)}$, from top level down, compute $m \leftarrow \mathsf{Dec}_{OME}(sk, C)$. If $m \neq \perp$, output m. If the trials fail for all $sk \in {\mathbf{sk}_i}^{(t-\Delta)}$, then repeat the above procedure for the keys in the right-most column ${\mathbf{sk}_i}^{(t)}$ in ${SK_i}^{(t)}$. If all trials fail, output $\perp$.

Note on Efficiency. The GKM encryption cost is a multi-encryption with $\Delta \log n$ keys, i.e. $\Delta \log n$ standard encryptions. The GKM decryption cost is $O(\log n)$ *multi-encryption* decryptions. We can reduce both costs by modifying the OME scheme so that the OME scheme is used as Key Encapsulation, to encrypt a random key k for an IND-CCA *symmetric* encryption scheme, padded with a tag of s zero bits. The key k is then used to encrypt the message using the symmetric encryption. The resulting OME scheme remains IND/IK-CCA and complete, but (1) All the public-key encryption costs can be done off-line, before the encryptor knows the plaintext, which in particular pushes all the encryption cost in the secret handshake scheme of Section 6 off-line; and (2) Heuristically, the oblivious decryptor rejects an attempt to decrypt a ciphertext using a wrong key with $1 - 2^{-s}$ probability. For all the ElGamal-based OME schemes we provide (using DHAES, Cramer-Shoup, or DDH-based ElGamal with the Fujisaki-Okamoto transformation), this means that the decryption cost in the above GKM scheme takes only $2 \log n$ exponentiations and $O(\Delta(\log n)^2)$ xor's.

Theorem 2. *If the underlying oblivious multi-encryption scheme is ϵ-complete and IND+IK-CCA secure then the above PKGKM scheme is ϵ'-complete and IND+IK-CCA, where $\epsilon' = 2\Delta \log n\epsilon$.*

6 Unlinkable Handshakes from Key-Private PKGKM

Let (Setup, KGen, Revoke, Enc, Dec) be a Δ-complete IND/IK-CCA GKM scheme and $H : \{0,1\}^* \rightarrow \{0,1\}^k$ is a hash function modeled as random oracle in the security analysis. The SH scheme uses the same algorithm Setup, KGen, and Revoke, and the $\mathsf{Handshake}$ protocol is shown in Figure 1. Each player's inputs in the protocol is a triple (SK, TPK, $\mathsf{resp/init}$) where SK is that player's GKM key for his/her group, TPK is the public key of this player's *target* group, and $\mathsf{resp/init}$ is the player's role in the protocol. Let final be a special symbol, different from resp and init. Parameter $\hat{k}$ can be set as $\hat{k} = 2k$. In particular, it must be large enough so that the probability that two sessions choose the same nonce is negligible in k. Hash function H has a $\hat{k}$-bit range.

We refer to the full paper [JL07] for proofs of the following claims:

Theorem 3 (Δ-limited completeness). *Suppose the underlying Group Key Management scheme is Δ-Limited ϵ-complete. Then our Secret Handshake scheme is Δ-Limited ϵ'-complete, where $\epsilon' = 2\epsilon$.*

Theorem 4 (Security of the SH Scheme). *If the underlying Group Key Management scheme is (t, ϵ, n, q_D)-IND-CCA secure and Δ-limited ϵ_c-complete,*

$\mathbf{U}_i(SK_i, TPK_i, \mathsf{init})$		$\mathbf{U}_j(SK_j, TPK_j, \mathsf{resp})$
$G_i = \mathsf{SKGroup}(SK_i)$		$G_j = \mathsf{SKGroup}(SK_j)$
$TG_i = \mathsf{PKGroup}(TPK_i)$		$TG_j = \mathsf{PKGroup}(TPK_j)$
$s_i \overset{R}{\leftarrow} \{0,1\}^{\hat{k}}$	$\xrightarrow{s_i}$	$s_j \overset{R}{\leftarrow} \{0,1\}^{\hat{k}}$
	$\xleftarrow{s_j, \phi_j}$	$\phi_j \leftarrow \mathsf{Enc}(TPK_j, (G_j, s, \mathsf{resp}, K_j))$ for $s = s_i \| s_j$; $K_j \overset{R}{\leftarrow} \{0,1\}^k$
$\phi_i \leftarrow \mathsf{Enc}(TPK_i, (G_i, s, \mathsf{init}, K_i))$ for $s = s_i \| s_j$; $K_i \overset{R}{\leftarrow} \{0,1\}^k$		
$(G'_j, s'_j, r'_j, K'_j) \leftarrow \mathsf{Dec}(SK_i, \phi_j)$;		
if $r'_j = \mathsf{resp}$ and $(G'_j, s'_j) = (TG_i, s)$,		
$h_i \leftarrow H(\mathsf{init}, K_i, K'_j)$	$\xrightarrow{\phi_i, h_i}$	$(G'_i, s'_i, r'_i, K'_i) \leftarrow \mathsf{Dec}(SK_j, \phi_i)$;
		if $r'_i = \mathsf{init}$ and $(G'_i, s'_i) = (TG_j, s)$,
		$h_j \leftarrow H(\mathsf{resp}, K'_i, K_j)$
	$\xleftarrow{h_j}$	
if $h_j = H(\mathsf{resp}, K_i, K'_j)$		if $h_i = H(\mathsf{init}, K'_i, K_j)$
output $\hat{K} = H(\mathsf{final}, K_i, K'_j)$.		output $\hat{K} = H(\mathsf{final}, K'_i, K_j)$.

Fig. 1. Privacy-protecting AKE protocol $\mathsf{Handshake}(U_i, U_j)$

then the SH scheme constructed in Section 6 is $(t', \epsilon', n, n', R, q_S, q_H)$*-secure, for at most* q_S *commands,* q_H *hash queries,* n' *groups,* n *members per group, and maximum* R *sessions for each member, for* $R = q_D/n$, $q_S = q_D$, $t' = t - (R+2)nn'\Delta \log n \cdot t_{exp}$, *where* t_{exp} *is the cost of a single exponentiation, and* $\epsilon' = nn'R \cdot (\epsilon + (q_H + q_S) \cdot 2^{-\hat{k}} + \epsilon_c)$.

Theorem 5 (Anonymity and Affiliation/Policy Hiding Property of the SH Scheme). *If the underlying group key management scheme is* $(t_1, \epsilon_1, n, q_D)$*-IND-CCA and* $(t_2, \epsilon_2, n, q_D)$*-IK-CCA secure, then the SH scheme constructed in Section 6 is* $(t', \epsilon', n, n', R, q_S, q_H)$*-private (unlinkable and affiliation/policy hiding), for at most* q_S *commands,* q_H *hash queries,* n' *groups,* n *members per group, and maximum* R *sessions for each member, for* $R = q_D/n$, $q_S = q_D$, $t = \min\{t_1, t_2\} - (R+2)nn'\Delta \log n \cdot t_{exp}$, *where* t_{exp} *is the cost of a single exponentiation, and* $\epsilon = nn'R(\epsilon_1 + \epsilon_2 + (q_H + q_S) \cdot 2^{-\hat{k}} + \epsilon_c)$.

References

[AB07] G. Ateniese and M. Blanton. Secret handshakes with dynamic and fuzzy matching. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium, NDSS*, 2007.

[ABR01] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHAES. In *CT-RSA*, pages 143–158, 2001.

[BBDP01] M. Bellare, A. Boldyrevay, A. Desaiz, and D. Pointchevalx. Key-privacy in public-key encryption. In *Aisacrypt '01, Proceedings*, 2001.

[BBs03] M. Bellare, A. Boldyreva, and J. staddon. Randomness re-use in multi-recipient encryption schemes. In *Public-Key Cryptography '03, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[BBW06] A Barth, D Boneh, and B. Waters. Private encrypted content distribution using private broadcast encryption. In *Proceedings of Financial Crypto*, 2006.

[BDS$^+$03] D. Balfanz, G. Durfee, N. Shankar, D.K. Smetters, J. Staddon, and H.C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, 2003.

[BHS04] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies in hidden credentials. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, 2004.

[CFN88] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proc. Crypto'88*, volume 403 of *LNCS*, pages 319–327. Springer-Verlag, 1988.

[CJT04] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption. In *proceedings of Asiacrypt*, 2004.

[CK02] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology - EUROCRYPT 2002*, 2002.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. Eurocrypt '01*, pages 93–118, 2001.

[DF02] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop*, pages 61–80, 2002.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology-CRYPTO'99*, pages 537–554, August 1999.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[JKT07a] S. Jarecki, J. Kim, and G. Tsudik. Authenticated group key agreement protocols with the privacy property of affiliation-hiding. In *RSA Conference – Cryptography Track*, 2007.

[JKT07b] S. Jarecki, J. Kim, and G. Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange protocols with perfect forward privacy. manuscript, 2007.

[JL07] S. Jarecki and X. Liu. Unlinkable secret handshakes and key-privacy in group key management scheme. `http://eprint.iacr.org/2007/`, 2007.

[KP98] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptography - CRYPT0 1998*, Santa Barbara, CA, August 1998.

[TX05] Gene Tsudik and Shouhuai Xu. Brief announcement: a flexible framework for secret handshakes. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 39–39, New York, NY, USA, 2005. ACM Press.

[Ver05] D. Vergnaud. RSA-based secret handshakes. In *In International Workshop on Coding and Cryptography, Bergen, Norway*, 2005.

[WGL98] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *SIGCOMM '98*, 1998.

[WHA97] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. *IETF draft wallner-key*, 1997.

[XY04] Shouhuai Xu and Moti Yung. k-anonymous secret handshakes with reusable credentials. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 158–167, New York, NY, USA, 2004. ACM Press.

Identity-Based Proxy Re-encryption

Matthew Green and Giuseppe Ateniese

The Johns Hopkins University
Department of Computer Science
3400 N. Charles Street; Baltimore, MD 21218, USA
{mgreen,ateniese}@cs.jhu.edu

Abstract. In a proxy re-encryption scheme a semi-trusted proxy converts a ciphertext for Alice into a ciphertext for Bob without seeing the underlying plaintext. A number of solutions have been proposed in the public-key setting. In this paper, we address the problem of Identity-Based proxy re-encryption, where ciphertexts are transformed from one *identity* to another. Our schemes are compatible with current IBE deployments and do not require any extra work from the IBE trusted-party key generator. In addition, they are non-interactive and one of them permits multiple re-encryptions. Their security is based on a standard assumption (DBDH) in the random oracle model.

Keywords: proxy re-encryption, identity-based encryption, bilinear maps.

1 Introduction

In a proxy re-encryption scheme, a proxy can convert an encryption computed under Alice's public-key into an encryption intended for Bob. Such a scheme can be used by Alice to temporarily forward encrypted messages to Bob without giving him her secret key. The fundamental property of proxy re-encryption schemes is that the proxy is not fully trusted, i.e., it does not know the secret keys of Alice or Bob and does not learn the plaintext during the conversion. The proxy and Bob, however, are not allowed to collude, thus it is usually assumed that at least one of the two is honest or that their collusion is preventable or detectable via other means.

A number of proxy re-encryption protocols have been proposed in the context of public-key encryption [1,2,3,4,5]. In this work we extend the notion of proxy re-encryption to the area of Identity-Based Encryption (IBE), in which senders encrypt messages using the recipient's identity (a string) as the public key. For example, Charles could encrypt a message for Alice by just using her email address. First introduced by Shamir in 1984 and then realized by Boneh-Franklin [6] and by Cocks [7] several years later, identity-based encryption has proven useful in solving many key-distribution issues, and has facilitated the development of a variety of novel cryptographic protocols, *e.g.,* secret handshakes [8], public-key searchable encryption [9,10], CCA2-secure public-key encryption [11], and digital signatures [12]. The Boneh-Franklin scheme is particularly efficient, and has been commercially deployed [13].

Our identity-based proxy re-encryption (IB-PRE) schemes allow a proxy to translate a ciphertext encrypted under Alice's identity into one computed under Bob's identity. To permit this translation, Alice generates and provisions the proxy with a *delegation key* (or "*re-encryption key*"), that the proxy uses to perform the re-encryption. No information about the secret keys of Alice or Bob can be deduced from this value, nor does the proxy learn anything about the underlying plaintext of the messages it processes. Our constructions are compatible with existing Boneh-Franklin IBE deployments, and can be implemented using existing secrets and parameters.

Users in an Identity-Based Encryption scheme request keys from a trusted party known as a Private Key Generator (PKG). Thus, in principle, it is possible that delegation keys could be generated by the PKG directly, rather than by individual scheme users. However, we categorically exclude this possibility and we focus only on *non-interactive* schemes where individual users delegate their own decryption rights without the involvement of the Private Key Generator. This is for theoretical and practical reasons: (1) From a theoretical point of view, having the PKG, or any other trusted party, generating the proxy keys makes the problem of finding IB-PRE schemes quite unchallenging given prior art, (2) from a practical point of view, it is clearly undesirable to have the PKG involved in the generation of proxy keys. It would constitute a considerable bottleneck in many applications, it would force the PKG to be *on-line* and available even during the generation of proxy keys (other than IBE keys), and, in certain applications, it would make the PKG liable for creating (potentially unwanted) decryption rights.

Previous Work. Mambo and Okamoto proposed a technique for delegating decryption rights in [1]. Blaze, Bleumer and Strauss [2] later presented the first secure "atomic" re-encryption primitive: an Elgamal-based scheme in which the proxy could not learn the message being processed. Unfortunately, the BBS approach was inherently bidirectional: a corrupted proxy could re-encrypt ciphertexts not only from Alice to Bob, but also from Bob to Alice. Jakobsson [4], and Zhou, *et. al.* [14] addressed this collusion problem via quorum-based protocols which divided the proxy into many distinct components.

More recent works have focused on *unidirectional* schemes, where collusion between a delegator and the proxy does not compromise the delegatee. Dodis and Ivan [5] realized a form of unidirectional proxy *encryption* by doubly-encrypting messages under two separate keys (or by splitting a single decryption key into two parts). Their approach permits a form of proxy re-encryption when parties pre-distribute shared secrets. Ateniese, Fu, Green and Hohenberger [3] proposed several non-interactive unidirectional proxy re-encryption schemes that eliminated the need for pre-shared keys and permitted arbitrary delegations. That work left an interesting open problem, which we address in this paper: namely, to construct chosen-ciphertext secure (CCA) proxy re-encryption schemes. Canetti and Hohenberger [15] also addressed this problem in the public key setting, proposing a CCA-secure bidirectional proxy re-encryption scheme. Though the constructions differ from ours, their security definition is compatible and we adopt some aspects of their presentation for consistency.

Finally, Boneh, Goh and Matsuo [16] presented a "hybrid" form of proxy re-encryption based on IBE. In such schemes, the PKG performs all delegations; thus users are unable to perform offline ("non-interactive") delegations and each delegation requires an online request to the PKG. Furthermore, the Boneh-Goh-Matsuo approach specifies a new private-key generation algorithm and it seems therefore incompatible with existing IBE deployments.

Paper Outline. The outline of the rest of this paper is as follows. In section 3 we present definitions for Identity-Based Proxy Re-encryption and for the hardness assumptions used in our proofs. In section 4 we introduce our constructions. In section 5 we discuss several applications for the new primitives. Finally, section 6 lists open research problems and provides our conclusions.

2 Properties of Our Schemes

Ateniese *et. al.* [3] proposed a series of properties by which to evaluate proxy re-encryption schemes. We briefly reiterate some of these properties, in particular those that our scheme provides and that, we believe, are relevant for practical instantiations of Identity-Based Proxy Re-encryption.

- *Unidirectionality.* A unidirectional scheme permits user A to delegate to user B, without permitting A to decrypt user B's ciphertexts.
- *Non-Interactivity.* Non-interactive schemes permit user A to construct a re-encryption key $rk_{id_A \rightarrow id_B}$ while offline, (*i.e.,*without the participation of B or the Private Key Generator).
- *Multiple-use capability.* A multi-use scheme permits the proxy (or proxies) to perform multiple consecutive re-encryptions on a ciphertext, *e.g.,* re-encrypt from id_A to id_B, then re-encrypt the result from id_B to id_C and so on.
- *Space-optimality.* Many existing schemes (*e.g.,* [5,16,3]) incur additional communication costs in order to support re-encryption. This inefficiency takes several common forms, including: (a) ciphertext expansion upon re-encryption (see the practical implementations of [3]), (b) a required pre-distribution stage in which secrets are shared with delegatees (as in [5]), or (c) the inclusion of ciphertext material that is discarded during re-encryption (see [16]).

In this paper we focus on unidirectional schemes only. Notice that a bidirectional scheme can always be achieved by running a unidirectional one in both directions, i.e., from Alice to Bob and vice versa. Thus, a unidirectional IB-PRE is clearly a more powerful primitive than a bidirectional one but also harder to devise.

In addition, we believe that non-interactivity is a fundamental property and our schemes provide it. In a non-interactive scheme, Alice can generate the re-encryption key from Bob's identity, without ever involving Bob. In the identity-based setting, this property provides an interesting twist: Alice can delegate decryption rights to delegatees that do not exist yet or will join the system later. Moreover, as noted by Boneh and Franklin [6], identities can be seen as credentials and express conditions. For instance, an encryption under *"Alice || security-clearance || time period"* can be opened by Alice only if she has security clearance and within the time period specified in the string. Analogously, in our schemes, Alice can specify the conditions under which the delegation of decryption rights has to happen. We will explore applications of this feature in section 5.

In section 4.2 we discuss an optimization that provides for space-optimal proxy re-encryption in some circumstances. Finally, one of our schemes is *multi-use* in the sense that once a re-encryption from Alice to Bob is computed, the resulting ciphertext can be re-encrypted again from Bob to Charles, etc., multiple times. Finding a unidirectional and multi-use scheme was left as an open problem in prior art for the public-key case. We show how to achieve this property for our IB-PRE but at the cost of allowing the ciphertext to expand linearly with respect to the number of re-encryptions (however, this appears to be inevitable for a non-interactive scheme).

3 Definitions

We begin by describing the setting and computational problems used within this work. We then formally define an Identity-Based Proxy Re-encryption scheme and propose a new, generalized security definition.

Definition 1 (Bilinear Map). We say a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a *bilinear map* if:

1. $\mathbb{G}, \mathbb{G}_T$ are groups of the same prime order q.
2. For all $a, b \in \mathbb{Z}_q^*$, $g \in \mathbb{G}$, $e(g^a, g^b) = e(g, g)^{ab}$.
3. The map is non-degenerate (i.e., if $\mathbb{G} = \langle g \rangle$, then $\mathbb{G}_T = \langle e(g, g) \rangle$).
4. e is efficiently computable.

For simplicity our constructions are defined in the *symmetric* setting as above. However they also work in the *asymmetric* setting with a bilinear map of the form: $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

Definition 2 (Decisional Bilinear Diffie Hellman Assumption (DBDH)). Our schemes are based on the assumed intractability of the Decisional Bilinear Diffie-Hellman problem (DBDH) in $\mathbb{G}, \mathbb{G}_T$. This assumption is believed to hold in certain groups, and used as the basis of several Identity-Based Encryption schemes, *e.g.,* [17,18].

We define the DBDH problem as follows: Let $(\mathbb{G}, \mathbb{G}_T)$ be a pair of bilinear groups with an efficiently computable pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and let g be a random generator of $\mathbb{G}$. The DBDH problem is to decide, given a tuple of values $(g, g^a, g^b, g^c, T) \in \mathbb{G}^4 \times \mathbb{G}_T$ (where $a, b, c \in_R \mathbb{Z}_q^*$), whether $T = e(g, g)^{abc}$ or if T is a random element of $\mathbb{G}_T$.

Let k be a security parameter of sufficient size. Formally, we say that the DBDH assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ if for all probabilistic polynomial time algorithms $\mathcal{A}$, the following condition is true:

$$\left| \begin{array}{l} \Pr\left[a, b, c \xleftarrow{\$} \mathbb{Z}_q^*;\ 1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}).\right] - \\ \Pr\left[a, b, c \xleftarrow{\$} \mathbb{Z}_q^*;\ T \xleftarrow{\$} \mathbb{G}_T;\ 1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, T).\right] \end{array} \right| \leq \nu(k)$$

Where $\nu(\cdot)$ is defined as a *negligible* function, *i.e.,* for all polynomial functions $p(\cdot)$, $\nu(k) < 1/p(k)$.

3.1 Identity-Based Proxy Re-encryption

An Identity-Based Proxy Re-encryption (IB-PRE) scheme is an extended Identity Based Encryption scheme. The first extension is an algorithm that generates *re-encryption keys* that can be given to the proxy. The proxy uses the second algorithm to apply these re-encryption keys to ciphertexts and "atomically" re-encrypt them from one identity to another. In a *non-interactive* scheme, re-encryption keys may be generated by the delegator using only her IBE secret key— the IBE master secret is not required.

Encryption Levels. Our definitions refer to the notion of an "encryption level" as an implicit property of a ciphertext. A ciphertext generated directly using the Encrypt algorithm is termed a "level-1" ciphertext. Applying the re-encryption algorithm to a level-ℓ ciphertext results in a level-$(\ell + 1)$ ciphertext. Specific constructions may optionally place bounds on the number of consecutive re-encryptions; for instance, non-"multi-use" schemes such as [5,16,3] are limited to a single re-encryption. In our definitions below, we define MaxLevels as the highest-possible encryption level (for a single-use scheme, this value is 2).

Definition 3 (Non-interactive Identity-Based Proxy Re-encryption (IB-PRE)). A *non-interactive identity-based proxy re-encryption scheme* is tuple of algorithms (Setup, KeyGen, Encrypt, Decrypt, RKGen, Reencrypt):

- Setup$(1^k$, MaxLevels$)$ accepts a security parameter and optionally a value indicating the maximum number of consecutive re-encryptions permitted by the scheme. The algorithm outputs both the master public parameters (params) which are distributed to users, and the master secret key (msk) which is kept private.
- KeyGen(params, msk, id) on input an identity $id \in \{0,1\}^*$ and the master secret key, outputs a decryption key sk_{id} corresponding to that identity.
- Encrypt(params, id, m) on input a set of public parameters, an identity $id \in \{0,1\}^*$, and a plaintext $m \in \mathcal{M}$, output c_{id}, the encryption of m under the specified identity.
- RKGen(params, sk_{id_1}, id_1, id_2) on input a secret key sk_{id_1} (derived via the KeyGen algorithm) and identities $(id_1, id_2) \in \{0,1\}^*$, outputs a *re-encryption key* $rk_{id_1 \rightarrow id_2}$.

- $\mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, c_{id_1})$ on input a ciphertext c_{id_1} under identity id_1, and a re-encryption key $rk_{id_1 \to id_2}$ (generated by the RKGen routine), outputs a "re-encrypted" ciphertext c_{id_2}.
- $\mathsf{Decrypt}(\mathsf{params}, sk_{id}, c_{id})$ decrypts the ciphertext c_{id} using the secret key sk_{id}, and outputs a plaintext or the distinguished symbol $\perp$.

Correctness. Intuitively, an IB-PRE scheme is *correct* if the Decrypt algorithm always outputs the expected decryption of a properly-generated ciphertext (when supplied with the appropriate decryption key). We define "proper generation" as the process of (1) encrypting a plaintext using Encrypt, and subsequently (2) iteratively applying the Reencrypt algorithm up to $\mathsf{MaxLevels} - 1$ times using valid re-encryption keys.

Slightly more formally, let $c_{id_1} \leftarrow \mathsf{Reencrypt}^n(\cdots, \mathsf{Encrypt}(\mathsf{params}, \cdot, m))$ be a properly-generated ciphertext. Then $\forall m \in \mathcal{M}, \forall id_1, id_2 \in \{0,1\}^*, \forall n < \mathsf{MaxLevels} - 1$, where $sk_{id_1} = \mathsf{KeyGen}(\mathsf{msk}, id_1)$, $sk_{id_2} = \mathsf{KeyGen}(\mathsf{msk}, id_2)$, $rk_{id_1 \to id_2} \leftarrow \mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_1, id_2)$, the following propositions hold:

- $\mathsf{Decrypt}(\mathsf{params}, sk_{id_1}, c_{id_1}) = m$
- $\mathsf{Decrypt}(\mathsf{params}, sk_{id_2}, \mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, c_{id_1})) = m$

Security. Security definitions for Identity-Based Encryption (see [6]) address the case where keyholders collude by combining secrets. Identity-Based Proxy re-encryption schemes require a further extension of this collusion guarantee, to model the presence of colluding *proxies* provisioned with re-encryption keys. Many existing security definitions (*e.g.,* [2,5,3]) address the proxy via separate definitional games. We choose instead to incorporate all of these properties into a single game, by providing re-encryption keys to the adversary via an oracle.

When the adversary possesses re-encryption keys, we must naturally restrict it in some ways to avoid a trivial condition, *e.g.,* to prevent it from obtaining a set of re-encryption keys leading from the challenge identity id^* to some identity for which the adversary holds a decryption key. In the CCA case, these restrictions are more complex. To simplify the presentation, we adopt the notion of *derivative* ciphertexts introduced in [15].

Definition 4 (Security of Non-Interactive Identity Based Proxy Re-Encryption (IND-prID-CPA, IND-prID-CCA)). Let $\mathcal{S}$ be an IB-PRE scheme defined as a tuple of algorithms (Setup, KeyGen, Encrypt, Decrypt, RKGen, Reencrypt). Security is defined according to the following game $Exp^{\mathcal{A},\,\mathsf{IND\text{-}prID\text{-}ATK}}$, where $\mathsf{ATK} \in (\mathsf{CPA}, \mathsf{CCA})$.

1. Setup. Run $\mathsf{Setup}(1^k)$ to get $(\mathsf{params}, \mathsf{msk})$, and give params to $\mathcal{A}$.
2. Find phase. $\mathcal{A}$ makes the queries (`extract`, `rkextract`, `decrypt`, `reencrypt`). If $\mathsf{ATK} = \mathsf{CPA}$, the queries (`decrypt`, `reencrypt`) are answered with $\perp$.
   - On (`extract`, id), return $\mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$.
   - On (`rkextract`, id_1, id_2), extract the key $sk_{id_1} = \mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id_1)$ and return $\mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_1, id_2)$.

- On $(\texttt{decrypt}, id, c)$: Extract $sk_{id} = \mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$ and return $\mathsf{Decrypt}(\mathsf{params}, sk_{id}, c)$.
- On $(\texttt{reencrypt}, id_1, id_2, c)$: Extract $sk_{id} = \mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$, and derive a re-encryption key $rk_{id_1 \rightarrow id_2} = \mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_1, id_2)$. Return $\mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \rightarrow id_2}, id_1, id_2, c)$.

At the conclusion of this phase $\mathcal{A}$ selects $id^* \in \{0,1\}^*$ and $(m_0, m_1) \in \mathcal{M}^2$. $\mathcal{A}$ is restricted to choices of id^* such that "trivial" decryption is not possible using keys extracted during this phase (*e.g.,* by using re-encryption keys to translate from id^* to identity id' for which $\mathcal{A}$ holds a decryption key).

3. Challenge. When $\mathcal{A}$ presents $(\texttt{choice}, id^*, m_0, m_1)$, select $i \xleftarrow{\$} \{0,1\}$ and compute $c^* = \mathsf{Encrypt}(\mathsf{params}, id^*, m_i)$. Return c^* to $\mathcal{A}$.
4. Guess stage. $\mathcal{A}$ makes queries as in the Find stage, with the following restrictions.
   (a) $\mathcal{A}$ is restricted from querying on $(\texttt{decrypt}, id, c)$ if $\langle id, c\rangle$ is a *challenge derivative*. This notion is defined inductively (as in [15]):
      i. $\langle id^*, c^*\rangle$ is a challenge derivative.
      ii. If$\langle id, c\rangle$ is a challenge derivative, and$\mathcal{A}$ has issued the query $(\texttt{reencrypt}, id, id', c)$ to receive a value c', then $\langle id', c'\rangle$ is a challenge derivative.
      iii. If $\langle id, c\rangle$ is a challenge derivative, $\mathcal{A}$ has issued query $(\texttt{rkgen},\ id, id')$ to receive $rk_{id \rightarrow id'}$, and $c' = \mathsf{Reencrypt}(rk_{id_1 \rightarrow id_2}, id, id', c)$, then $\langle id', c'\rangle$ is a challenge derivative.

   (b) $\mathcal{A}$ is restricted from querying on $(\texttt{extract}, id)$ if there exists a challenge derivative $\langle id, c\rangle$.
   (c) $\mathcal{A}$ is restricted from querying on $(\texttt{rkextract}, id, id')$ if $\mathcal{A}$ has previously issued the query $(\texttt{extract}, id')$ and there exists a challenge derivative $\langle id, c\rangle$.
   (d) $\mathcal{A}$ is restricted from querying on $(\texttt{reencrypt}, id, id', c)$ if the query would produce (perhaps implicitly) a challenge derivative $\langle id'', c''\rangle$ and $\mathcal{A}$ has previously issued the query $(\texttt{extract}, id'')$.

   At the conclusion of this stage, $\mathcal{A}$ outputs i', where $i' \in \{0,1\}$.

The outcome of the game is determined as follows: If $i' = i$ then $\mathcal{A}$ wins the game. $\mathcal{A}$'s advantage in the above game, $Adv_{\mathcal{A}}^{\mathsf{IND\text{-}prID\text{-}ATK}}$ is defined as $|Pr\,[i' = i] - 1/2|$. For $\mathsf{ATK} \in (\mathsf{CPA}, \mathsf{CCA})$ we say that the Identity-Based Proxy Re-encryption scheme $\mathcal{S}$ is IND-prID-ATK-secure if for all probabilistic polynomial time algorithms $\mathcal{A}$, $Adv_{\mathcal{A}}^{\mathsf{IND\text{-}prID\text{-}ATK}} \leq \nu(k)$.

Bidirectional and PKG-based IBE Proxy Re-encryption. To underscore the importance of non-interactive unidirectional proxy re-encryption, we note that it is possible to construct a bidirectional or PKG-based scheme from any unidirectional scheme. In the bidirectional case, RKGen_{bi} is implemented via two separate calls to RKGen: derive $rk_{id_1 \rightarrow id_2}$ and $rk_{id_2 \rightarrow id_1}$, which are together functionally equivalent to $rk_{id_1 \leftrightarrow id_2}$. Similarly, RKGen_{pkg} can be implemented by deriving secret keys (sk_{id_1}, sk_{id_2}) at the PKG using the KeyGen routine and subsequently calling RKGen as in the bidiretional case. We leave further discussion and non-generic constructions of these alternative forms of proxy re-encryption for future work.

4 Non-interactive Unidirectional Proxy Re-encryption Schemes

The first schemes we present are based on Boneh and Franklin's IBE scheme [6], and are secure under the Decisional Bilinear Diffie-Hellman Assumption (DBDH) in the random oracle model. While ciphertexts in the proposed schemes have a different form from those in the standard Boneh-Franklin scheme, the master parameters and secret keys remain unchanged. As a result, it is possible to implement proxy re-encryption within an existing Boneh-Franklin deployment (*i.e.,* using pre-existing parameters and keys).

4.1 A First Attempt (IBP1)

Consider a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G} = \langle g \rangle$. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two independent hash functions[1] such that: $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $\mathcal{H}_2 : \mathbb{G}_T \rightarrow \mathbb{G}$. Finally, let s and g^s be the master secret and public key of the PKG, respectively. For some $r \in_R \mathbb{Z}_q^*$, an encryption of $m \in \mathbb{G}_T$ under Alice's identity can be computed as:

$$IBE_{Alice}(m) = (g^r, m \cdot e(g^s, \mathcal{H}_1(Alice))^r)$$

Suppose Alice wants to delegate her decryption rights to Bob. She must generate a re-encryption key to give to the proxy. Let $IBE_{Bob}(\cdot)$ be a standard identity-based encryption under Bob's identity. Alice selects a random $X \overset{\$}{\leftarrow} \mathbb{G}_T$ and generates the re-encryption key as:

$$rk_{Alice \rightarrow Bob} = \mathcal{H}_1(Alice)^{-s} \cdot \ fulldomainhash_2(X),\ IBE_{Bob}(X),$$

Given an encryption for Alice, $IBE_{Alice}(m) = (c_1, c_2)$ the proxy can transform it into an encryption for Bob by releasing: $(c_1' = c_1, c_2' = c_2 \cdot e(g^r, rk_{Alice \rightarrow Bob}), c_3' = IBE_{Bob}(X))$. Indeed, notice that:

$$\begin{aligned} c_1' &= g^r \\ c_2' &= m \cdot e(g^r, \mathcal{H}_2(X)), \\ c_3' &= IBE_{Bob}(X). \end{aligned}$$

Bob can recover X from c_3' and then m by computing $c_2'/e(c_1', \mathcal{H}_2(X))$.

In practice, the scheme presented above can be seen as a variant of the efficient Dodis/Ivan [5] key-splitting approach applied to settings where the decryption process makes use of a bilinear map. Note that (1) The scheme is *unidirectional* since the key $rk_{Alice \rightarrow Bob}$ can be used to convert ciphertexts from Alice to Bob but not vice versa. (2) It is *non-interactive* since Bob is not involved during

[1] Both $\mathcal{H}_1(\cdot)$ and $\mathcal{H}_2(\cdot)$ are more properly "hash-and-encode" functions (see Boneh-Franklin [6] for a detailed definition). Each function consist of a standard hash function which maps inputs to elements of the finite field of order q and then uses an admissible encoding function, MapToPoint, to map those elements into points in $\mathbb{G}$.

the generation of the re-encryption key. (3) It provides *non-transitivity* since the proxy is not allowed to create new re-encryption keys from the existing ones. (4) Finally, we observe that the scheme is *multi-use* since the proxy can re-encrypt the result of a re-encryption and do it multiple times. To see this, consider the re-encryption ciphertext above: (c_1', c_2', c_3'). Notice that c_3' is just a standard IBE encryption for Bob! A proxy equipped with a re-encryption key $rk_{Bob \rightarrow Charles}$ could just apply the re-encryption algorithm *recursively* to c_3' and allow Charles to recover X which in turn allows him to recover the original message m.

Scheme Description. We now provide a formal description of the scheme (IBP1).

- Setup. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map, where $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ have order q. Let $\mathcal{H}_1, \mathcal{H}_2$ be independent full-domain hash functions $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $\mathcal{H}_2 : \mathbb{G}_T \rightarrow \mathbb{G}$. To generate the scheme parameters, select $s \overset{\$}{\leftarrow} \mathbb{Z}_q^*$, and output $\mathsf{params} = (\mathbb{G}, \mathcal{H}_1, \mathcal{H}_2, g, g^s)$, $\mathsf{msk} = s$.
- $\mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$. To extract a decryption key for identity $id \in \{0,1\}^*$, return $sk_{id} = \mathcal{H}_1(id)^s$.
- $\mathsf{Encrypt}(\mathsf{params}, id, m)$. To encrypt m under identity id, select $r \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and output $c_{id} = (g^r, m \cdot e(g^s, \mathcal{H}_1(id))^r)$.
- $\mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_2)$. Select $X \overset{\$}{\leftarrow} \mathbb{G}_T$ and compute $\langle R_1, R_2 \rangle = \mathsf{Encrypt}$ $(\mathsf{params},\ id_2,\ X)$. Return $rk_{id_1 \rightarrow id_2} = \langle R_1, R_2, sk_{id_1}^{-1} \cdot \mathcal{H}_2(X) \rangle$.
- $\mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \rightarrow id_2}, c_{id_1})$. To re-encrypt a level-$\ell$ ciphertext from id_1 to id_2, first parse c_{id_1} as $(C_1, \ldots, C_{2\ell})$ and $rk_{id_1 \rightarrow id_2}$ as (R_1, R_2, R_3). Next:
  1. If $\ell = 1$, output $c_{id_2} = \langle C_1, C_2 \cdot e(C_1, R_3)), R_1, R_2 \rangle$.
  2. If $\ell > 1$, treat the elements $\langle C_{2\ell-1}, C_{2\ell} \rangle$ as a first-level ciphertext δ. Compute $\langle C_1', C_2', C_3', C_4' \rangle = \mathsf{Reencrypt}(rk_{id_1 \rightarrow id_2}, \delta)$. Output the ciphertext $c_{id_2} = \langle C_1, \cdots, C_{2\ell-2}, C_1', C_2', C_3', C_4' \rangle$.
- $\mathsf{Decrypt}(\mathsf{params}, sk_{id}, c_{id})$. Parse the level-$\ell$ ciphertext c_{id} as $(C_1, \ldots, C_{2\ell})$. Next:
  1. If $\ell = 1$ output $m = C_2 / e(C_1, sk_{id})$.
  2. If $\ell > 1$, treat the pair $\langle C_{2\ell-1}, C_{2\ell} \rangle$ as a first-level ciphertext c_{id}', and compute $X_\ell = \mathsf{Decrypt}(sk_{id}, c_{id}')$. For $i = (\ell - 1)$ descending to 1, compute $X_i = C_{2i} / e(C_{2i-1}, \mathcal{H}_2(X_{i+1}))$. Finally, output X_1 as the plaintext.

Each level-ℓ ciphertext in the above scheme contain 2ℓ elements. In principle, the scheme permits an arbitrary number of re-encryptions on a ciphertext, with a two-element ciphertext expansion on each re-encryption.

Correctness. We first show correctness for first-level ciphertexts (*i.e.,* those produced by $\mathsf{Encrypt}$). Let $c_{id_1} = (g^r, m \cdot e(g^s, \mathcal{H}_1(id_1))^r)$ be the first-level encryption of m under id_1, and $sk_1 = \mathcal{H}_1(id_1)^s$ be the corresponding decryption key. The decryption process produces the following result:

$$(m \cdot e(g^s, \mathcal{H}_1(id_1))^r) / e(g^r, \mathcal{H}_1(id_1)^s) = m$$

The correctness under re-encryption is shown as follows. Given a first-level ciphertext $c_{id_1} = (g^r, C_2)$ and a correctly-formed re-encryption key $rk_{id_1 \to id_2} = (\langle R_1, R_2 \rangle = \mathsf{Encrypt}(\mathsf{params}, id_2, X), R_3)$, we obtain the "second-level" ciphertext $c_{id_2} = (g^r, C_2' = C_2 \cdot e(g^r, R_3), R_1, R_2)$ where C_2' is:

$$\begin{aligned} C_2' &= C_2 \cdot e(g^r, R_3) \\ &= m \cdot e(g^s, \mathcal{H}_1(id_1))^r) \cdot e(g^r, \mathcal{H}_1(id_1)^{-s} \cdot \mathcal{H}_2(X)) \\ &= m \cdot e(g, \mathcal{H}_2(X))^r \end{aligned}$$

Given $sk_{id_2} = \mathcal{H}_1(id_2)^s$ we decrypt $c_{id_2} = (g^r, C_2', R_1, R_2)$ as follows. Begin by decrypting the first-level ciphertext $\hat{c}_{id_2} = \langle R_1, R_2 \rangle$ under sk_{id_2}: $X = \mathsf{Decrypt}(\mathsf{params}, sk_{id_2}, \hat{c}_{id_2})$. Then compute $C_2'/e(g^r, \mathcal{H}_2(X))$ to obtain m. Having shown correctness for a single re-encryption, the correctness for multiple re-encryptions follows. Given level-ℓ ciphertext c_{id_i} and sk_{id_i}, strip the the final two elements and treat them as a first-level ciphertext under id_i, decrypting to reveal X_ℓ. Use the value X_ℓ as a decryption secret for the previous two elements, and repeat until the final two elements remain. The final value in this chain contains the original message m.

Security. We next show that IBP1 scheme defined above meets the IND-prID-CPA definition if the Decisional Bilinear Diffie-Hellman assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. Our proof is in the random oracle model, and is an extension of the original proof of Boneh/Franklin [6].

Theorem 1. *If there exists a p.p.t. adversary $\mathcal{A}$ that wins the* IND-prID-CPA *game on $IBP1$ with non-negligible advantage, then there exists an adversary $\mathcal{B}$ that solves the DBDH problem over $\mathbb{G}, \mathbb{G}_T$ with non-negligible advantage.*

A proof sketch of Theorem 1 is presented in Appendix A.

4.2 An Optimization

Ciphertexts in scheme section 4.1 expand upon re-encryption. This is caused by the inclusion within the re-encrypted ciphertext of a portion of the re-encryption key. There are scenarios where Bob knows that the original ciphertext was intended for Alice (this information can even be appended to the ciphertext) and there is no need for multiple re-encryptions. In such cases we can simplify our construction by using a result of Sakai *et. al.* [19]. Specifically, in the Boneh-Franklin IBE symmetric setting, Alice and Bob inherently share a secret key $K_{AB} = e(\mathcal{H}_1(Alice), \mathcal{H}_1(Bob))^s$. Alice can use this value to compute the re-encryption key as follows:

$$rk_{Alice \to Bob} = \mathcal{H}_1(Alice)^{-s} \cdot \mathcal{H}_3(\{K_{AB}\}||Alice \to Bob).$$

Where $\{K_{AB}\}$ denotes binary representation, and $\mathcal{H}_3 : \{0,1\}^* \to \mathbb{G}$ is an independent full domain hash function. The string "$Alice \to Bob$" is added to ensure that a re-encryption key from Bob to Alice is computed under a distinct secret (bidirectional re-encryption). Note that the resulting scheme permits only a single re-encryption for each ciphertext. A primary advantage of this construction is the absence of ciphertext expansion during re-encryption.

4.3 A Chosen Ciphertext Secure Scheme (IBP2)

The scheme presented above is secure under chosen plaintext attack. While this is the level of security provided by many IBE and proxy re-encryption schemes (*e.g.,* [3,18] and the practical proxy encryption constructions of Dodis/Ivan [5]), it is important to consider stronger definitions such as security under adaptive chosen ciphertext attack.

Background. A common approach to building CCA-secure Identity-Based *Encryption* schemes in the random oracle model is to begin with a CPA-secure construction, and then apply the generic Fujisaki-Okamoto conversion [20] (see *e.g.,* [6,21]). It is tempting to believe that this approach is by itself sufficient to construct CCA-secure IB-PRE schemes. Unfortunately, this does not appear to be the case. Notice that a re-encryption proxy grants adversaries an alternative means by which adversaries may decrypt ciphertexts: a malicious delegatee B may decrypt A's ciphertexts by first using the proxy to re-encrypt from $id_A \rightarrow id_B$, and then decrypting the result under his own secret key. When a malicious delegatee uses the proxy to "alternatively decrypt" in this manner, he need not follow the specified F-O decryption algorithm, and can ignore the critical ciphertext validity checks. Unfortunately, the validity checks of the F-O approach cannot be moved into the re-encryption process, as they fundamentally require access to the decryption secret.

Intuition. In order to surmount the issues raised above, we propose an approach that provides the proxy with the means to verify ciphertext validity and reject improperly-formed ciphertexts. As a result of this check, a malicious delegatee no longer gains any advantage by using the re-encryption proxy as an oracle. The building block of our construction is a Hierarchical Identity-Based Proxy Re-encryption scheme, which we implement using a modified form of the Gentry-Silverberg HIBE [22] (this scheme is in turn based on the Boneh/Franklin scheme). To achieve IND-prID-CCA-secure IB-PRE, we make use of the Canetti, Halevi and Katz (CHK) [11] technique, which allows us to transform a HIBE into a CCA-secure IBE scheme with a type of *publicly-verifiable* ciphertext validity check. In order to present a more efficient construction, we re-use randomness and implement the CHK transform using a Boneh/Lynn/Shacham short signature [23].

The Construction. We now present a *single-use*, non-interactive CCA-secure IB-PRE construction (IBP2).

- Setup. Let $n()$ be a polynomial function of the security parameter k. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map, where $\mathbb{G}, \mathbb{G}_T$ have order q and $\mathbb{G} = \langle g \rangle$. To generate the scheme parameters, select $s \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and output $\mathsf{params} = (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5, g, g^s)$, $\mathsf{msk} = s$, with independent hash functions $\mathcal{H}_{1-6}$ defined as below:

$$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}, \mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{G}$$
$$\mathcal{H}_3 : \{0,1\}^* \rightarrow \mathbb{G}, \mathcal{H}_4 : \mathbb{G}_T \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$$
$$\mathcal{H}_5 : \mathbb{G}_T \rightarrow \{0,1\}^n$$

- $\mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$. To extract a decryption key for identity $id \in \{0,1\}^*$, return $sk_{id} = \mathcal{H}_1(id)^s$.
- $\mathsf{Encrypt}(\mathsf{params}, id, m \in \{0,1\}^n)$. To encrypt m under identity $id \in \{0,1\}^*$, first:
  1. Select $\sigma \xleftarrow{\$} \mathbb{G}_T$, and set $r = \mathcal{H}_4(\sigma, m)$.
  2. Compute $\langle A, B, C\rangle = (g^r, \sigma \cdot e(g^s, \mathcal{H}_1(id)^r), m \oplus \mathcal{H}_5(\sigma))$.
  3. Compute $S = \mathcal{H}_3(id || \langle A, B, C\rangle)^r$.
  4. Output the ciphertext $c = \langle S, A, B, C\rangle$.
- $\mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_1, id_2)$. To compute a re-encryption key from $id_1 \to id_2$:
  1. Select $N \xleftarrow{\$} \{0,1\}^{n(k)}$, and compute $K = e(sk_{id_1}, \mathcal{H}_1(id_2))$.
  2. Output $rk_{id_1 \to id_2} = \langle N, \mathcal{H}_2(K || id_1 || id_2 || N) \cdot sk_{id_1}\rangle$.
- $\mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, c_{id_1})$. To re-encrypt a first-level ciphertext, first parse c_{id_1} as (S, A, B, C), and parse $rk_{id_1 \to id_2}$ as $\langle N, R\rangle$. Next:
  1. Let $h = \mathcal{H}_3(id_1 || \langle A, B, C\rangle)$.
  2. Check if $e(g, S) = e(h, A)$. If not, return $\perp$.
  3. Otherwise, select $t \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $B' = B / \frac{e(A, R \cdot h^t)}{e(g^t, S)}$.
  4. Output the re-encrypted ciphertext $c_{id_2} = (A, B', C, id_1, N)$.
- $\mathsf{Decrypt}(\mathsf{params}, sk_{id}, c_{id})$. To decrypt a first-level (non re-encrypted) ciphertext, first parse c_{id} as (S, A, B, C). Next:
  1. Let $h = \mathcal{H}_3(id, \langle A, B, C\rangle)$.
  2. Select $t \xleftarrow{\$} \mathbb{Z}_q^*$, and compute $\sigma' = B / \frac{e(A, sk_{id} \cdot h^t)}{e(g^t, S)}$.
  3. Compute $m' = C \oplus \mathcal{H}_5(\sigma')$, and $r' = \mathcal{H}_4(\sigma', m')$.
  4. Verify that $S = h^{r'}$ and $A = g^{r'}$. If either check fails, return $\perp$, otherwise output m'.

  To decrypt a second-level (re-encrypted) ciphertext, first parse c_{id} as (A, B, C, id_{src}, N). Next:
  1. Compute $K = e(\mathcal{H}_1(id_{src}), sk_{id})$.
  2. Compute $\sigma' = B \cdot e(A, \mathcal{H}_2(K || id_{src} || id || N))$.
  3. Compute $m' = C \oplus \mathcal{H}_5(\sigma')$, and $r' = \mathcal{H}_4(\sigma', m')$.
  4. Verify that $A = g^{r'}$. If this check fails, return $\perp$, otherwise output m'.

Correctness. We begin by showing correctness for first-level ciphertexts (*i.e.*, those produced by $\mathsf{Encrypt}$). Let $c_{id_1} = \langle S, A, B, C\rangle$ be the first-level encryption of m under id_1, with $h = \mathcal{H}_3(id_1 || \langle A, B, C\rangle)$.

$$c_{id_1} = \langle h^r, g^r, \sigma \cdot e(g^s, \mathcal{H}_1(id)^r), m \oplus \mathcal{H}_5(\sigma)\rangle$$

Let $sk_1 = \mathcal{H}_1(id_1)^s$ be the corresponding decryption key. For a random $t \in \mathbb{Z}_q^*$, the decryption process proceeds as follows:

$$\begin{aligned}(\sigma \cdot e(g^s, \mathcal{H}_1(id_1))^r) / \frac{e(g^r, \mathcal{H}_1(id_1)^s \cdot h^t)}{e(g^t, h^r)} &= \sigma \\ \mathcal{H}_5(\sigma) \oplus (m \oplus \mathcal{H}_5(\sigma)) &= m \\ g^{\mathcal{H}_4(\sigma, m)} &\overset{?}{=} g^r \\ h^{\mathcal{H}_4(\sigma, m)} &\overset{?}{=} h^r\end{aligned}$$

The correctness of re-encryption is shown as follows. Given the first-level ciphertext c_{id_1} presented above, and a correctly-formed re-encryption key $rk_{id_1 \to id_2} = \langle N, R \rangle$, the re-encryption process begins with a ciphertext validity check:

$$e(g, h^r) \overset{?}{=} e(h, g^r)$$

Recall that $R = sk_{id_1} \cdot W$ where $W = \mathcal{H}_2(e(\mathcal{H}_1(id_1)^s, \mathcal{H}_1(id_2))||id_1||id_2||N)$. To generate the "second-level" ciphertext $c_{id_2} = (A, B', C, id_1, N)$, we choose $t \in_R \mathbb{Z}_q^*$ and obtain:

$$B' = (\sigma \cdot e(g^s, \mathcal{H}_1(id_1))^r))/\frac{e(g^r, R \cdot h^t)}{e(g^t, h^r)} = \sigma/e(g^r, W)$$

Finally, we decrypt the re-encrypted ciphertext $c_{id_2} = (A, B', C, id_1, N) = (g^r, \sigma/e(g^r, W), m \oplus \mathcal{H}_5(\sigma), id_1, N)$ as follows. Given $sk_{id_2} = \mathcal{H}_1(id_2)^s$:

$$\begin{aligned} \mathcal{H}_2(e(\mathcal{H}_1(id_1), \mathcal{H}_1(id_2)^s)||id_1||id_2||N) &= W \\ (\sigma/e(g^r, W)) \cdot e(g^r, W) &= \sigma \\ (m \oplus \mathcal{H}_5(\sigma)) \oplus \mathcal{H}_5(\sigma) &= m \\ g^{\mathcal{H}_4(\sigma, m)} &\overset{?}{=} g^r \end{aligned}$$

Security. We claim that IBP2 meets the IND-prID-CCA definition if the Decisional Bilinear Diffie-Hellman assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. Our proof is in the random oracle model. Due to space limitations we are unable to include the proof here. However, it can be found in the full version of this paper [24].

5 Applications of Identity-Based Proxy Re-encryption

Proxy Re-encryption has a number of practical applications, which have been detailed in previous works. All of these applications translate directly to the Identity-Based setting but with some additional features.

Secure Email with IBE. The most natural application of proxy re-encryption is to allow Bob to read Alice's encrypted emails while she is on vacation. Messages are encrypted under the email address *"alice@company.com"* and are translated by the proxy into encryptions under *"bob@company.com"*. The proxy does not learn the content of the messages being translated.

Attribute-based Delegations. As noted by Boneh and Franklin [6], identities can be created to include attributes or to express conditions. For instance, a message encrypted under *"alice || lawyer || from 01/01/2008"* can be read by Alice only if she is a lawyer and not before the beginning of year 2008. This idea applies directly to our IBE-PRE scheme and it allows Alice to specify under which conditions the proxy is allowed to translate her ciphertexts into Bob's. For instance, consider the case of *temporary* delegations [3] where the time is divided in time intervals t_1, t_2, …, t_k and Alice can specify that the proxy can translate her ciphertexts for Bob only during t_i. With our scheme, Alice could just create the proxy key:

$$rk_{Alice\|t_i \to Bob},$$

so that any encryption under $Alice\ \|t_i$ can be converted into an encryption for Bob but not during other time periods. This eliminates the need for designing a separate and specialized scheme as it was done in [3].

Even more interestingly, Alice could specify the conditions under which Bob can read her messages. For instance, a re-encryption key of this form:

$$rk_{Alice \to Bob\|after\ Nov\ 2007\|security-clearance},$$

would specify that encryptions under Alice's identity can be converted into encryptions for Bob but that Bob can read the messages only in the future, after Nov 2007, and under the condition that he is able to obtain a security clearance.

Bridging IBE and PKE. *Hybrid* proxy re-encryption is a concept put forward by Boneh, Goh and Matsuo [16] to create a bridge between IBE and public-key based encryption (PKE). Our scheme can also be used to translate from IBE to PKE. Indeed, consider the ciphertext after the re-encryption, which has the form:

$$c_1' = g^r,\ \ c_2' = m \cdot e(g^r, \mathcal{H}_2(K)),\ \ c_3' = IBE_{Bob}(K).$$

Notice that c_3' is a standard (semantically-secure) id-based encryption of a key K. This encryption can be substituted with a public-key based one (or even a semantically-secure symmetric one). In this way, an encryption under Alice's identity is converted into an encryption under Bob's public-key. Our approach provides some advantages over the one in [16]. Indeed, no TTP is involved in creating re-encryption keys and parameters in our scheme are compatible with those of the standard Boneh-Franklin IBE.

Travel Key. Boneh and Franklin [6] suggested to use an IBE system to store temporary keys into the laptop during travel so that, if the laptop is lost or stolen, only those keys get exposed. The idea is to let Bob act as a PKG that generates his own master secret and public keys. Alice could use Bob's master public-key to encrypt messages for Bob under identities *day1*, *day2*, ... etc., for all days in which Bob is traveling. Bob can store into his laptop just the keys corresponding to those days while leaving his master secret key safely stored elsewhere.

This solution, however, requires Bob to inform of his travels any of his potential correspondents and have them act according to the encryption scheme (that is, they have to encrypt under *day1*, *day2*, etc.). An alternative solution is to set up a proxy (Bob's mail server, for instance) with a re-encryption key of the form:

$$rk_{Bob \to Bob's-Travel-Key}.$$

Every encryption intended for Bob will be encrypted under Bob's travel key, which is the only secret key stored into his laptop. Notice that the proxy does not have to be trusted and can be set-up by a system administrator who won't be able to read Bob's messages.

Access Control in Networked File Storage. In [3], the authors describe an application of proxy re-encryption to the distribution of key material within a cryptographic filesystem. Each file stored on an untrusted file server is encrypted using a symmetric key; these keys are encrypted under a public *master key* which is stored alongside the encrypted material. When a user wishes to decrypt a file, the semi-trusted *keyserver* re-encrypts these encapsulated symmetric keys from the master key to the keys of individual users who can then decrypt. The key server provides access control for the encrypted material, but does not itself possess the ability to decrypt files.

This application translates naturally to the Identity Based setting with the additional benefit of allowing the holder of the master key to specify access control policies directly within the identity strings of the users. A re-encryption key can even be generated before an individual has joined the system.

6 Conclusions and Future Work

In this work we introduced new constructions enabling non-interactive, unidirectional proxy re-encryption in the IBE setting. Our schemes are very efficient and can be deployed within standard IBE frameworks. New compelling applications can be realized thanks to our schemes, most notably attribute-based delegation and access control.

An interesting open problem is to find efficient constructions for *multi-use* CCA-secure IBE-PRE schemes. Another important open problem is to find efficient IBE-PRE secure in the standard model (rather than in the RO model).

Acknowledgments. We would like to thank Susan Hohenberger for helpful discussion and suggestions.

References

1. Mambo, M., Okamoto, E.: Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. IEICE Trans. Fund. Electronics Communications and Computer Science **E80-A/1** (1997) 54–63
2. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Proceedings of Eurocrypt '98. Volume 1403. (1998) 127–144
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In: the 12th Annual Network and Distributed System Security Symposium. (2005) 29–43 Full version available at http://eprint.iacr.org/2005/028.
4. Jakobsson, M.: On quorum controlled asymmetric proxy re-encryption. In: Proceedings of Public Key Cryptography. (1999) 112–121
5. Dodis, Y., Ivan, A.: Proxy cryptography revisited. In: Proceedings of the Tenth Network and Distributed System Security Symposium. (2003)

6. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil Pairing. In: Advances in Cryptology (CRYPTO 2001). Volume 2139 of Lecture Notes in Computer Science., Springer (2001) 213–229
7. Cocks, C.: An identity based encryption scheme based on Quadratic Residues. In: Cryptography and Coding, 8th IMA International Conference. Volume 2260 of Lecture Notes in Computer Science., Springer (2001) 360–363
8. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.C.: Secret handshakes from pairing-based key agreements. In: Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP '03), Washington, DC, USA, IEEE Computer Society (2003) 180
9. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology - EUROCRYPT 2004. Volume 3027 of Lecture Notes in Computer Science. (2004) 506–522
10. Waters, B.R., Balfanz, D., Durfee, G., Smetters, D.K.: Building an encrypted and searchable audit log. In: Proceedings of Network and Distributed System Security Symposium 2004 (NDSS'04), San Diego, CA (2004)
11. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from Identity Based Encryption. In: Proceedings of Eurocrypt '04. Volume 3027 of Lecture Notes in Computer Science., Springer-Verlag (2004) 207–222
12. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil Pairing. SIAM Journal of Computing **32**(3) (2003) 586–615
13. Voltage Security, Inc. (http://www.voltage.com)
14. Zhou, L., Marsh, M.A., Schneider, F.B., Redz, A.: Distributed blinding for ElGamal re-encryption. Technical Report 2004–1924, Cornell Computer Science Department (2004)
15. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. (*Unpublished manuscript*)
16. Boneh, D., Goh, E.J., Matsuo, T.: Proposal for P1363.3 Proxy Re-encryption. (`http://grouper.ieee.org/groups/1363/IBC/submissions/NTTDataProposal-for-P1363.3-2006-09-01.pdf`)
17. Boneh, D., Boyen, X.: Efficient selective-id secure Identity-Based Encryption without random oracles. In: Proceedings of Eurocrypt '04. Volume 3027 of Lecture Notes in Computer Science., Springer (2004) 223–238
18. Waters, B.: Efficient Identity-Based Encryption without random oracles. In: Proceedings of Eurocrypt '05. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 114–127
19. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: Symposium on Cryptography and Information Security (SCIS 2000). (2000)
20. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Proceedings of Crypto '99. Volume 1666 of Lecture Notes in Computer Science., Springer (1999) 537–554
21. Yang, P., Kitagawa, T., Hanaoka, G., Zhang, R., Matsuura, K., Imai, H.: Applying Fujisaki-Okamoto to Identity-Based Encryption. In: Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 16th International Symposium (AAECC-16). Volume 3857 of Lecture Notes in Computer Science., Springer (2006) 183–192
22. Gentry, C., Silverberg, A.: Hierarchical ID-Based cryptography. In: Proceedings of Asiacrypt '02. Volume 2501 of Lecture Notes in Computer Science., Springer (2002) 548–566

23. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil Pairing. In: ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security. Volume 2248 of Lecture Notes in Computer Science., London, UK, Springer-Verlag (2001) 514–532
24. Green, M., Ateniese, G.: Identity-based proxy re-encryption. Cryptology ePrint Archive, Report 2006/473 (2006) http://eprint.iacr.org/2006/473.pdf.

A Security Proof of IBP1

Proof sketch. Let $\mathcal{A}$ be a p.p.t. algorithm that has non-negligible advantage ϵ in $Exp^{\mathcal{A},\,\mathsf{IND\text{-}prID\text{-}CPA}}$ against IBP1. We use $\mathcal{A}$ in order to construct a second algorithm $\mathcal{B}$ which has non-negligible advantage at solving the DBDH problem in $\mathbb{G}, \mathbb{G}_T$. Algorithm $\mathcal{B}$ accepts as input an appropriately-distributed tuple $\langle \mathbb{G} = \langle g \rangle, g^a, g^b, g^c, T \rangle \in \mathbb{G}^4 \times \mathbb{G}_T$ and outputs 1 if $T = e(g,g)^{abc}$. We now describe the algorithm $\mathcal{B}$, which interacts with algorithm $\mathcal{A}$ via the IND-prID-CPA interface.

Oracle Queries. $\mathcal{B}$ simulates the random oracle $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}$ as follows: On receipt of a query for id (on which it has not previously been queried), select $z \xleftarrow{\$} \mathbb{Z}_q^*$ and randomly flip a weighted coin to set $\alpha \leftarrow 1$ with probability γ (see below), and $\alpha \leftarrow 0$ otherwise. If $\alpha = 0$ then compute $h \leftarrow (g^c)^z$, else compute $h \leftarrow g^z$. Record the tuple (id, h, z, α). Finally, return h as the result of the query (if id has previously been queried, simply locate the existing tuple and return the previously-computed h). Note that the distribution of the values h returned by the simulated oracle is random, regardless of the choice of α. $\mathcal{B}$ simulates (initial) queries to the random oracle $\mathcal{H}_2 : \mathbb{G}_T \to \mathbb{G}$ by simply returning elements $\in_R \mathbb{G}$.

Our simulation proceeds as follows:

1. Setup. $\mathcal{B}$ generates the scheme's master parameters params$=(\mathbb{G}, \mathcal{H}_1, \mathcal{H}_2, g, g^a)$ and gives this tuple to $\mathcal{A}$.
2. Find. When $\mathcal{A}$ submits (extract, id), $\mathcal{B}$ evaluates $\mathcal{H}(id)$ as described above, to obtain (id, h, z, α). $\mathcal{B}$ outputs $sk_{id} = (g^a)^z$ to $\mathcal{A}$.

   When $\mathcal{A}$ submits (rkextract, id_1, id_2), $\mathcal{B}$ selects $r \xleftarrow{\$} \mathbb{Z}_q^*$, $x \xleftarrow{\$} \mathbb{G}$ and $X \xleftarrow{\$} \mathbb{G}_T$, then evaluates $\mathcal{H}_1(id_1)$ and $\mathcal{H}_1(id_2)$ to obtain the values $(\alpha_1, z_1), (\alpha_2, z_2)$ (for id_1, id_2 respectively). Now:
   (a) If $\alpha_1 = 0$ then $\mathcal{B}$ returns $rk_{id_1 \to id_2} = \left((g^b)^r, T^{rz_2} \cdot X, x\right)$ to $\mathcal{A}$ (note that this key is incorrectly formed, see section below).
   (b) If $\alpha_1 = 1$ then $\mathcal{B}$ returns the correctly-formed tuple $rk_{id_1 \to id_2} = (g^r, e(g^a, \mathcal{H}_1(id_2)^r) \cdot X, (g^a)^{-z_1} \cdot \mathcal{H}_2(X))$.
3. Challenge. At the conclusion of the Find phase, $\mathcal{A}$ outputs (id^*, m_0, m_1) with the condition that $\mathcal{A}$'s choice of id^* is not trivial.[2] $\mathcal{B}$ selects $i \xleftarrow{\$} \{0,1\}$,

[2] We reject a choice of id^* when $\mathcal{A}$ has previously extracted a series of re-encryption keys, and a decryption key $sk_{id'}$ such that $\mathcal{A}$ can consecutively re-encrypt ciphertexts from id^* to id'.

then evaluates $\mathcal{H}_1(id^*)$ to recover (id^*, h, z, α) from the $\mathcal{H}_1$ table. $\mathcal{B}$ returns $c^* = \langle g^b, T^z \cdot m_i \rangle$ to $\mathcal{A}$.

4. Guess. $\mathcal{A}$ makes queries $(\mathsf{extract}, \ldots)$ and $(\mathsf{rkextract}, \ldots)$ as in the Find stage, except that $\mathcal{A}$ is restricted from making any query that would result in a trivial situation (a valid decryption path from id^* to an identity for which the adversary possesses a secret key). At the conclusion of this phase, $\mathcal{A}$ outputs its guess $i' \in \{0, 1\}$.

   Conditions for Abort. Let α_i represent the value α generated by $\mathcal{H}_1(id_i)$. Prior to outputting a value, $\mathcal{B}$ verifies several conditions:

   (a) The value α corresponding to id^* is 0.
   (b) For each of $\mathcal{A}$'s queries $(\mathsf{extract}, id_i)$, $\alpha_i = 1$.
   (c) For each of $\mathcal{A}$'s queries $(\mathsf{rkextract}, id_i, id_j)$, where $id_i \rightarrow id_j$ lies along a path leading from id^*, $\alpha_j = 0$.
   (d) For each of $\mathcal{A}$'s queries $(\mathsf{rkextract}, id_i, id_j)$, where $id_i \rightarrow id_j$ does not lie along a path leading from id^*, $\alpha_i = 1$.

   If any of the above conditions are false, $\mathcal{B}$ aborts the simulation. Otherwise, if $i' = i$, $\mathcal{B}$ outputs 1, or 0 otherwise.

Claim. If $\langle g, g^a, g^b, g^c, T \rangle$ is a DBDH tuple and $\mathcal{B}$ does not abort, then $\mathcal{A}$'s view is identical to the real attack— with the significant exception of re-encryption keys having the form $rk_{id^* \rightarrow \cdot}$. We argue below that $\mathcal{A}$ cannot detect these improperly-formed re-encryption keys, and thus cannot distinguish the simulation. Hence, when the input to $\mathcal{B}$ is a DBDH tuple, then the challenge ciphertext c^* is a correct encryption of m_i under id^* and thus from definition of $\mathcal{A}$ and the argument above it holds that $\left|\Pr\left[\, i = i' \,\right] - \frac{1}{2}\right| = \epsilon$ and thus $\mathcal{B}$ outputs 1 with probability $\frac{1}{2}$ plus a non-negligible quantity. When the input to $\mathcal{B}$ is random, c^* represents the encryption of a random element in $\mathbb{G}_T$ and is independent of $\mathcal{B}$'s choice of i (and therefore $\Pr\left[\, i = i' \,\right] = \frac{1}{2}$). Thus, $\mathcal{B}$ succeeds in distinguishing DBDH tuples with non-negligible advantage.

Invalid Re-encryption keys. In the simulation above, every re-encryption key that lies along a path from id^* is incorrectly formed. (At the same time, it is easy to see that all other re-encryption keys *are* correctly formed.) Unfortunately, this condition is unavoidable, as the simulator does not possess the knowledge required in order generate a valid re-encryption key from the challenge identity id^*. To complete our proof, therefore, we make a separate argument that no adversary $\mathcal{A}$ can distinguish our simulation from a "real-world" interaction in which all values have the correct form. The heuristic argument for security is simple: each correctly-formed re-encryption key $rk_{id_1 \rightarrow id_2}$ consists of a semantically secure encryption (R_1, R_2) of some element $X \in_R \mathbb{G}_T$, along with the value $R_3 = (\mathcal{H}_1(id_2)^{-s} \cdot \mathcal{H}_2(X)) \in \mathbb{G}$. An incorrectly-formed re-encryption key replaces R_3 with some value $x \in_R \mathbb{G}$; this x can naturally be expressed as $(\mathcal{H}_1(id_2)^{-s} \cdot y)$ for some unknown $y \in \mathbb{G}$. Intuitively, an adversary who can distinguish malformed re-encryption keys in our simulation must therefore be able to determine that (R_1, R_2) do *not* encrypt some value $Y \in \mathbb{G}_T$ *s.t.* $\mathcal{H}_2(Y) = y$. We formalize the statement via the following Lemma.

Lemma 1 (Indistinguishability of simulations). If there exists a *p.p.t.* algorithm $\mathcal{A}'$ with non-negligible advantage ϵ' at distinguishing the simulation above from a "correct" simulation (in which all values are correctly-formed), then we can construct an algorithm $\mathcal{B}'$ that solves the DBDH problem in $(\mathbb{G}, \mathbb{G}_T)$ with non-negligible advantage.

Probability of abort. A variety of conditions in the above simulation can lead the simulator to abort. Boneh and Franklin [6] provide a technique for computing the value γ used in simulating the random oracle $\mathcal{H}_1$, and for placing bounds on the abort probability. We refer the reader to this discussion, and provide a detailed argument in the full version. $\square$

A More Natural Way to Construct Identity-Based Identification Schemes

Guomin Yang[1], Jing Chen[2], Duncan S. Wong[1], Xiaotie Deng[1], and Dongsheng Wang[2]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
{csyanggm,duncan,deng}@cs.cityu.edu.hk
[2] Department of Computer Science
Tsinghua University
Beijing, China
jingchen@cityu.edu.hk, wds@tsinghua.edu.cn

Abstract. Constructing identification schemes is one of the fundamental problems in cryptography, and is very useful in practice. An identity-based identification (IBI) scheme allows a prover to identify itself to a public verifier who knows only the claimed identity of the prover and some common information. In this paper, we propose a simple and efficient framework for constructing IBI schemes. Unlike some related framework which constructs IBI schemes from some standard identification schemes, our framework is based on some more fundamental assumptions on intractable problems. Depending on the features of the underlying intractable problems presumed in our framework, we can derive IBI schemes secure against passive, active and concurrent adversaries. We show that the framework can capture a large class of schemes currently proposed, and also has the potential to cover many newly constructed schemes. As an example, based on the Katz-Wang standard signature scheme, we propose a new IBI scheme that is secure against active adversaries in a concurrent manner. It can be seen that our framework also help simplify the security proofs for new IBI schemes. Finally, and of independent interest, we define a new notion for proof systems called Witness Dualism. This notion is weaker than that of witness indistinguishable and we show that it is enough for constructing an IBI scheme secure against the most powerful type of adversaries defined.

Keywords: Identity-based cryptography, Identification schemes, Concurrent attacks.

1 Introduction

In an identity-based cryptosystem, there is an authority having a master public/secret key pair. This authority can provide a user with a user secret key which is derived from the user's identity and the master secret key. In an identity-based identification (IBI) scheme, a user, playing the role of a prover, identifies itself to a verifier, who knows only the prover's identity and the master public key.

There are three notions for the security of IBI schemes: security against impersonation under passive attacks (id-imp-pa), active attacks (id-imp-aa), and concurrent attacks (id-imp-ca). In a passive attack, an adversary can obtain communication transcripts between the real prover and a verifier. In an active or concurrent attack, the adversary can directly communicate with the prover by playing the role of a cheating verifier. The difference between id-imp-aa and id-imp-ca is that in the former case, the adversary can have only one active session at a time, but in a concurrent attack, the adversary can have concurrent (or parallel) active sessions.

In this paper, we propose a simple and efficient method to construct IBI schemes. Our method is based on two notions, namely *trapdoor weak-one-more relation* and *trapdoor strong-one-more relation*. We show that the former one can be constructed from intractable problems such as trapdoor one-way permutations and the Computational Diffie-Hellman (CDH) problem; and the latter one can be constructed from the factoring problem, the RSA problem and any strongly unforgeable [1] (referred to as *non-malleability in* [17]) signature schemes. By applying a trapdoor weak-one-more relation with an honest verifier zero knowledge proof of knowledge, we get an IBI scheme secure against passive attacks. While if we apply a trapdoor strong-one-more relation with a *witness dualism* proof of knowledge, we obtain an IBI scheme secure against active and concurrent attacks. Since the notion of witness dualism is weaker than that of witness indistinguishability [9], any proof system which is witness indistinguishable can readily be used in our framework as a witness dualism proof system. Besides proposing the generic framework for constructing IBI schemes with various levels of security, we also propose a concrete scheme. The scheme is based on the Katz-Wang strongly unforgeable signature scheme. The concrete IBI scheme falls in our framework and can be shown easily to be id-imp-ca secure.

1.1 Related Work

Since Shamir introduced the identity-based cryptosystems [16], a lot of IBI schemes have been proposed. A survey can be found in [2]. In [2], the authors proposed a method to construct IBI schemes by using digital certificates: the master key generation center (or called authority) picks a public/secret key pair (pk, sk) for a standard identification (SI) scheme, and provides these to prover I along with a certificate *cert* consisting of the authority's signature on (I, pk). The prover sends pk, and *cert* to a verifier and identifies itself using the SI scheme. The verifier needs to know only I and the public key of the authority. Although simple, this method (named *certificate-based* IBI) is inefficient, and its significance is to answer a fundamental question: secure IBI schemes (in the standard model) exists if and only if one-way function exists. In [2], another framework is proposed that transforms any standard identification scheme which satisfies certain conditions (referred to as convertible SI schemes) to IBI schemes in the random oracle model [4].

Independently in [14], a transformation is proposed that converts some digital signature scheme to an IBI scheme. The authors showed that the resulting IBI scheme is id-imp-pa secure if the underlying signature scheme is existentially unforgeable against adaptive chosen message attack [10]. One aspect of this transformation is that it is not necessarily to be in the random oracle model, however, the signature scheme (the BLS short signature scheme [7]) they used to construct a concrete IBI scheme is only proven secure in the random oracle model.

In this paper, we propose a more "natural" and efficient method to construct IBI schemes. Comparing with the approach of[2] which requires a underlying provably secure convertible standard indentification (SI) scheme, our method starts directly from the definitions of some intractable problems. And more importantly, our construction explicitly explains the features a hard problem should have in order to achieve passive and active/concurrent security.

Our method is also more generic than that of [14], in the sense that constructing IBI schemes from standard signature schemes is just one of the many possible instantiations in our framework. Additionally, we can construct IBI schemes secure against active and concurrent attacks from strongly unforgeable signature schemes. In Sec. 5, we also construct a concrete IBI scheme that is secure against concurrent attacks from the Katz-Wang signature scheme [13].

2 Identity-Based Identification Schemes

An interactive proof system $(\mathbf{P}, \mathbf{V})$ is said to be *canonical* if it follows a three-move structure where prover $\mathbf{P}$ initiates a communication with verifier $\mathbf{V}$ by sending a *commitment* Cmt, distributed uniformly over a set CmtSet, to $\mathbf{V}$; $\mathbf{V}$ then replies with a *challenge* Ch chosen uniformly from a set ChSet; and $\mathbf{P}$ finishes the communication by sending a *response* Rsp to $\mathbf{V}$. $\mathbf{V}$ accepts or rejects according to the output of a deterministic function $1/0 \leftarrow \mathbf{Dec}(St_V, \mathrm{Cmt}\|\mathrm{Ch}\|\mathrm{Rsp})$ where St_V is the initial state of $\mathbf{V}$. The bitstring $\mathrm{Cmt}\|\mathrm{Ch}\|\mathrm{Rsp}$ is called a *conversation* between $\mathbf{P}$ and $\mathbf{V}$.

Let $k \in \mathbb{N}$ be a security parameter. A canonical interactive proof system $(\mathbf{P}, \mathbf{V})$ has commitment length $\beta(\cdot)$ if $|\mathrm{CmtSet}| \geq 2^{\beta(k)}$, has challenge length $\ell(\cdot)$ if $|\mathrm{ChSet}| \geq 2^{\ell(k)}$, and is *non-trivial* if the function $2^{-\beta(k)}$ is negligible in k.

Definition 1 (Identity-Based Identification (IBI)). *An identity-based identification (IBI) scheme consists of four probabilistic polynomial-time (PPT) algorithms* $(\mathbf{MKGen}, \mathbf{UKGen}, \mathbf{P}, \mathbf{V})$.

1. **MKGen:** *On input* 1^k*, it generates a master public/secret key pair* (mpk, msk).
2. **UKGen:** *On input msk and some identity I of a user, it outputs a user secret key* $usk[I]$.
3. $(\mathbf{P}, \mathbf{V})$ – ***User Identification Protocol:*** *The prover with identity I runs interactive algorithm* $\mathbf{P}$ *with initial state* $usk[I]$*, and the verifier runs* $\mathbf{V}$ *with initial state* (mpk, I)*. The first and last messages of the protocol belong to the prover. The protocol ends when* $\mathbf{V}$ *outputs either 'accept' or 'reject'.*

We require that for all $k \in \mathbb{N}$, $I \in \{0,1\}^*$, $(mpk, msk) \leftarrow \mathbf{MKGen}(1^k)$, and $usk[I] \leftarrow \mathbf{UKGen}(msk, I)$, $\mathbf{V}$ (initialized with mpk, I) always outputs '*accept*' after interacting with $\mathbf{P}$ (initialized with $usk[I]$).

The security of an IBI scheme is commonly considered against three types of attacks: impersonation under passive attacks (id-imp-pa), active attacks (id-imp-aa) and concurrent attacks (id-imp-ca). The following definitions are due to [2].

Definition 2 (id-imp-pa). *For an IBI scheme* $(\mathbf{MKGen}, \mathbf{UKGen}, \mathbf{P}, \mathbf{V})$*, the id-imp-pa security is defined by the following game, which is carried out by a simulator against an adversary* $\mathcal{A}$*.*

1. $(mpk, msk) \leftarrow \mathbf{MKGen}$ *is executed and* mpk *is given to* $\mathcal{A}$*. Two sets are maintained:* HU *and* CU*. Initially, both* HU *and* CU *are empty.*
2. $\mathcal{A}$ *can make queries to the following oracles:*
   (a) $\mathrm{INIT}(I)$ *– create a user with identity* I*: If* $I \in HU \cup CU$*,* $\perp$ *is returned indicating that* I *has already been created. Otherwise,* $usk[I] \leftarrow \mathbf{UKGen}(msk, I)$ *is executed and* I *is added into* HU*. A symbol '1' is returned indicating that the creation is successful.*
   (b) $\mathrm{CORR}(I)$ *– corrupt a user with identity* I*: If* $I \notin HU$*,* $\perp$ *is returned, otherwise,* I *is deleted from* HU *and added into* CU*, and* $usk[I]$ *is returned.*
   (c) $\mathrm{CONV}(I)$ *– get a conversation between a user (as the prover) and a verifier: If* $I \notin HU$*,* $\perp$ *is returned, otherwise, a conversation between a prover with initial state* $usk[I]$ *and a verifier with initial state* (mpk, I) *is returned.*
3. $\mathcal{A}$ *can adaptively query* INIT, CORR *and* CONV*, and then output an identity* $\mathrm{I_b} \in HU$*, which corresponds to the user that* $\mathcal{A}$ *wants to impersonate. After receiving* $\mathrm{I_b}$*, the simulator removes* $\mathrm{I_b}$ *from* HU *and adds it into* CU*.*
4. $\mathcal{A}$ *begins a run of the user identification protocol with a verifier* $\mathbf{V}$ *(initialized with* $(mpk, \mathrm{I_b})$*) which is simulated by the simulator.* $\mathcal{A}$ *can continue querying* INIT*,* CORR *and* CONV*. The simulate halts when* $\mathbf{V}$ *outputs 'accept' or 'reject'.*

The id-imp-pa advantage of $\mathcal{A}$ *on security parameter* k *is defined as the probability that* $\mathbf{V}$ *outputs 'accept'. The IBI scheme* $(\mathbf{MKGen}, \mathbf{UKGen}, \mathbf{P}, \mathbf{V})$ *is said to be id-imp-pa secure if the id-imp-pa advantage is negligible for any PPT adversary* $\mathcal{A}$*.*

id-imp-aa and id-imp-ca security. The id-imp-aa security is defined by a similar game, but the conversation oracle, CONV, is replaced by a proving oracle, PROV. $\mathcal{A}$ can select any identity $I \in$ HU and start a conversation with PROV which is the simulation of $\mathbf{P}(usk[I])$. The difference between id-imp-aa and id-imp-ca is that in the former case, $\mathcal{A}$ can have only one active session with PROV at a time, but in the latter case, $\mathcal{A}$ can have concurrent (or parallel) active sessions.

3 A Generic IBI Scheme Secure Against Passive Attacks

In this section, we propose a generic construction of IBI schemes that can be proven secure against passive attacks (namely, id-imp-pa secure in the sense of Def. 2). In the following, we define a relation called *trapdoor weak-one-more relation*, which enables our generic construction to capture many concrete IBI schemes which include GQ-IBI [11], Sh-IBI [16] (under the RSA assumption) and Hs-IBI [12], ChCh-IBI [8] (under the CDH assumption)[1].

A binary relation $\mathbf{R}$ on $W \times \Delta$ is a finite set of ordered pairs (x, y) such that $x \in W$ and $y \in \Delta$. x is called a witness of y. We denote the set of witnesses of y by $W(y)$.

Definition 3 (Trapdoor Weak-One-More Relation Family). *A family of trapdoor weak-one-more relations $\mathcal{R}$ is a triple of PPT algorithms* $(\mathbf{Gen}, \mathbf{Ver}, \mathbf{Inv})$*:*

1. **Gen:** *On input 1^k, where $k \in \mathbb{N}$ is the security parameter,* **Gen** *generates $(\langle \mathbf{R} \rangle, t)$ where $\langle \mathbf{R} \rangle$ denotes the description of relation $\mathbf{R}$ on $W \times \Delta$ and t a trapdoor information.*
2. **Ver:** *For any $k \in \mathbb{N}$, $(\langle \mathbf{R} \rangle, t) \leftarrow \mathbf{Gen}(1^k)$, $\mathbf{Ver}(1^k, \langle \mathbf{R} \rangle, x, y) = 1$ if and only if $(x, y) \in \mathbf{R}$, otherwise, it outputs 0.*
3. **Inv:** *On input $(1^k, \langle \mathbf{R} \rangle, y, t)$, it outputs x such that $(x, y) \in \mathbf{R}$ for any $y \in \Delta$.*
4. ***Weak-one-more resistance*:** *Consider the following game against an adversary $\mathcal{A}$ which is given $\langle \mathbf{R} \rangle$ but not t, and has access to two oracles:*
   (a) *A challenge oracle* RAM *that on any input returns a new random target point $y \in \Delta$.*
   (b) *An inversion oracle* INV *that on any input y,*
      i. *if y is an output of* RAM*, a witness of y is returned, and the same witness is returned if the same value of y is queried again;*
      ii. *if y is not an output of* RAM*, $\perp$ is returned indicating that the input is invalid.*

   $\mathcal{A}$ wins if $\mathcal{A}$ finds witnesses for all the target points output by RAM and makes strictly fewer queries to INV. We say that $(\langle \mathbf{R} \rangle, t)$ is a trapdoor weak-one-more relation if the probability to win the game is negligible in k for any PPT $\mathcal{A}$.

The trapdoor weak-one-more relation family can be instantiated easily and in many different ways. In the following, we describe several methods and show that they satisfy the definition of trapdoor weak-one-more relation family.

3.1 Instantiations of Trapdoor Weak-One-More Relations

Trapdoor One-way Permutation Based. Let $f : \Delta \rightarrow \Delta$ be a trapdoor one-way permutation. The following theorem describes a method to construct a trapdoor weak-one-more relation from any trapdoor one-way permutation.

[1] The abbreviations of these IBI schemes were first used by Bellare, Namprempre and Neven in [2].

Theorem 1. *The binary relation* $\mathbf{R}^{TOP} = \{(x, y) : x, y \in \Delta; f(x) = y\}$ *is a trapdoor weak-one-more relation.*

Proof. It is obvious that $\mathbf{R}^{TOP}$ is efficient to generate, verify, and find witness with trapdoor. Now we show that it also satisfies the weak-one-more resistance. Suppose there exists an adversary $\mathcal{A}$ which breaks the weak-one-more resistance. We build an adversary $\mathcal{B}$ to break the one-wayness of f. $\mathcal{B}$ is given a random instance $y^* \in \Delta$, and $\mathcal{B}$ is to find the inverse $x^* \in \Delta$ such that $f(x^*) = y^*$. Suppose $\mathcal{A}$ makes at most $Q(k)$ queries to RAM. Initially, $\mathcal{B}$ randomly selects a number $1 \leq i \leq Q(k)$ and simulates the weak-one-more resistance game as follows:

To answer j-th query to RAM, if $j \neq i$, $\mathcal{B}$ randomly selects $x_j \in \Delta$ and returns $y_j = f(x_j)$ to $\mathcal{A}$; if $j = i$, y^* is returned. When $\mathcal{A}$ makes a query to INV on y_j, if $y_j \neq y^*$, x_j is returned; otherwise, $\mathcal{B}$ aborts. If $\mathcal{A}$ finds a witness $\tilde{x}$ such that $f(\tilde{x}) = y^*$, $\mathcal{B}$ outputs $\tilde{x}$ and halts. If $\mathcal{A}$ halts, $\mathcal{B}$ halts.

It is easy to see that if $\mathcal{A}$ wins with probability at least ϵ, $\mathcal{B}$ breaks the one-wayness of f with probability at least $\epsilon/Q(k)$. □

Computational Diffie-Hellman (CDH) Assumption Based. To be more concrete, and also make our weak-one-more relation family more explicitly linked to the techniques of some actual IBI schemes (e.g. Hs-IBI [12] and ChCh-IBI [8]), we describe another instantiation of the weak-one-more relation defined above in Def. 3.

For a security parameter $k \in \mathbb{N}$, let q be a k-bit prime. Let $\mathbb{G}_1$ be an additive cyclic group of order q and $\mathbb{G}_2$ be a multiplicative cyclic group of the same order. Let P be a generator of $\mathbb{G}_1$. A bilinear map is defined as $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties: *bilinear*: For any $U, V \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_q$, $e(aU, bV) = e(U, V)^{ab}$; *non-degenerate*: $e(P, P) \neq 1$; and *computable*: there exists an efficient algorithm to compute $e(U, V)$ for any $U, V \in \mathbb{G}_1$.

The Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ is to compute abP from $\langle P, aP, bP\rangle$ where a, b are randomly selected from $\mathbb{Z}_q$. Based on the CDH problem, we can construct a trapdoor weak-one-more relation as follows: on input 1^k, **Gen** outputs $(\mathbb{G}_1, \mathbb{G}_2, q, P, e, \hat{S} = sP)$ where s is randomly selected from $\mathbb{Z}_q$, the relation is defined as $\mathbf{R}^{CDH} = \{(x, y) : x, y \in \mathbb{G}_1; e(P, x) = e(\hat{S}, y)\}$ and s is the trapdoor information.

Theorem 2. *If the CDH problem is hard,* $\mathbf{R}^{CDH}$ *is a trapdoor weak-one-more relation.*

The proof is similar to that for Theorem 1 and is omitted here.

Digital Signature Schemes Secure under Known Message Attacks. Besides trapdoor one-way permutation based and some concrete CDH assumption based instantiations, we now show that the trapdoor weak-one-more relation can also be constructed from a signature scheme which is only existentially unforgeable against known message attack (euf-kma) in the sense of [10]. This also demonstrates that the trapdoor weak-one-more relation defined above can be very useful for capturing some potentially new concrete construction methods of IBI schemes. This is also a new application for signature schemes which are proven secure under the weak notion of existential unforgeability, namely, against only known message attacks.

Let $\mathcal{SIG} = (\mathcal{KG}, \mathcal{S}, \mathcal{V})$ be a signature scheme defined on some message space $\mathcal{MS}$. Here, we assume that $|\mathcal{MS}| \geq 2^{\ell(k)}$ where $\ell(k)$ is super logarithmic in k. The security of euf-kma [10] is defined as follows: an adversary has signatures for a set (denoted by $\mathcal{M}_{known}$) of messages which are uniformly selected from $\mathcal{MS}$, the adversary's goal is to produce a signature for a message in $\mathcal{MS} \setminus \mathcal{M}_{known}$.

We can construct a trapdoor weak-one-more relation from $\mathcal{SIG}$ as follows: on input 1^k, **Gen** runs the key generation algorithm $\mathcal{KG}$ to generate a public/private key pair (pk, sk), the relation is defined as $\mathbf{R}^{SIG} = \{(x, y) : y \in \mathcal{MS}; \mathcal{V}(pk, y, x) = 1\}$ and sk is the trapdoor information.

Theorem 3. *If $\mathcal{SIG}$ is euf-kma, $\mathbf{R}^{SIG}$ is a trapdoor weak-one-more relation.*

Proof. Suppose there exists an adversary $\mathcal{A}$ which breaks the weak-one-more resistance. We build another adversary $\mathcal{F}$ which breaks $\mathcal{SIG}$ under the known message attacks. Suppose $\mathcal{A}$ makes at most $Q(k)$ queries to RAM. Initially, $\mathcal{F}$ obtains $Q(k) - 1$ message-signature pairs $\{(m_1, \sigma_1), \cdots, (m_{Q(k)-1}, \sigma_{Q(k)-1})\}$ where m_j $(1 \leq j \leq Q(k) - 1)$ is uniformly selected from $\mathcal{MS}$. Thus, $\mathcal{M}_{known} = \{m_1, \cdots, m_{Q(k)-1}\}$). $\mathcal{F}$ then uniformly selects m^* from $\mathcal{MS}$, and randomly inserts m^* into the message sequence. For simplicity, we assume any two messages in $\mathcal{M}_{known} \cup \{m^*\}$ are different. The proof then proceeds as in the proof of Theorem 1, $\mathcal{F}$ answers $\mathcal{A}$'s queries to RAM and INV by simply sending back the corresponding message/signature, $\mathcal{F}$ fails if $\mathcal{A}$ makes a query to INV on message m^*.

If $\mathcal{A}$ wins the weak-one-more resistance game with probability at least ϵ, $\mathcal{F}$ breaks the signature scheme with probability at least $\epsilon/Q(k)$. □

3.2 Our Generic Construction of IBI Schemes

We now start describing our method of constructing an IBI scheme. The method is based on the trapdoor weak-one-more-relation family (Def. 3) and the Honest Verifier Zero-Knowledge (HVZK) proof with special soundness defined as follows.

Definition 4. *A trapdoor weak-one-more relation* $\mathbf{R}$ *on* $W \times \Delta$ *has an HVZK proof with special soundness if there exists a non-trivial canonical proof system* $(\tilde{\mathbf{P}}, \tilde{\mathbf{V}})$ *such that for any* $y \in \Delta$,

1. **Completeness.** *If* $\tilde{\mathbf{P}}$ *knows* x *such that* $(x, y) \in \mathbf{R}$*, then* $\Pr(\tilde{\mathbf{V}}\,accepts) = 1$.
2. **Special Soundness.** *A witness of* y *can be computed from any two acceptable transcripts* $(\mathrm{Cmt}, \mathrm{Ch}_1, \mathrm{Rsp}_1)$ *and* $(\mathrm{Cmt}, \mathrm{Ch}_2, \mathrm{Rsp}_2)$ *such that* $\mathrm{Ch}_1 \neq \mathrm{Ch}_2$.
3. **Honest Verifier Zero Knowledge.** *There exists a polynomial time algorithm* $\mathcal{SIM}$ *such that on input* $(\langle \mathbf{R} \rangle, y)$ *its output distribution is computationally indistinguishable from the distribution of a real conversation between* $\tilde{\mathbf{P}}$*(initialized with a witness of* y*) and* $\tilde{\mathbf{V}}$*(initialized with* $\langle \mathbf{R} \rangle, y$*).*

Let $\mathrm{H} : \{0,1\}^* \rightarrow \Delta$ be a hash function that is considered to be a random oracle [4] for security analysis. We construct an IBI scheme as follows.

1. **MKGen:** $(\langle \mathbf{R} \rangle, t) \leftarrow \mathbf{Gen}(1^k)$. Set $mpk = \langle \mathbf{R} \rangle$ and $msk = t$.
2. **UKGen:** on input $I \in \{0,1\}^*$, run $x \leftarrow \mathbf{Inv}(1^k, \langle \mathbf{R} \rangle, \mathrm{H}(I), t)$ and set $usk[I] = x$.
3. $(\mathbf{P}, \mathbf{V})$**:** set $\mathbf{P}$ to be the prover algorithm $\tilde{\mathbf{P}}$ of the HVZK proof with initial state x, and $\mathbf{V}$ the verifier algorithm $\tilde{\mathbf{V}}$ of the HVZK proof with initial state $(\langle \mathbf{R} \rangle, \mathrm{H}(I))$.

The following theorem states that an IBI scheme constructed as above is id-imp-pa secure (Def. 2).

Theorem 4. *Let* $\mathbf{R}$ *be a trapdoor weak-one-more relation which has an HVZK interactive proof with special soundness. If the challenge length* $\ell(k)$ *of the HVZK proof is super logarithmic in* k*, the IBI scheme constructed above is* id-imp-pa *secure in the random oracle model.*

Proof. Given an adversary $\mathcal{A}$ that can break the IBI scheme with advantage ϵ, we construct an adversary $\mathcal{B}$ which breaks the weak-one-more resistance of the underlying trapdoor weak-one-more relation with advantage $\epsilon' \geq (\epsilon - 2^{-\ell(k)})^2$.

$\mathcal{B}$ simulates the id-imp-pa game by setting the $mpk = \langle \mathbf{R} \rangle$. $\mathcal{B}$ maintains two user lists HU and CU, which are empty at the beginning. $\mathcal{B}$ also maintains a table T, each row of T contains an identity I and the value of $\mathrm{H}(I)$. T is also empty at the beginning. $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows:

1. H-query: On input $I \in \{0,1\}^*$, $\mathcal{B}$ checks if I is in table T. If I is not in T, $\mathcal{B}$ asks its challenge oracle RAM to get a random point $y \in \Delta$, and sets $\mathrm{H}(I) = y$ by putting (I, y) in table T. If I is already in table T, the existing value is returned.
2. INIT(I): If $I \in \mathrm{HU} \cup \mathrm{CU}$, $\perp$ is returned. Otherwise, $\mathcal{B}$ checks whether I is in table T. If I is in T, I is added into HU and a symbol '1' is returned. Otherwise, $\mathcal{B}$ asks RAM to get a random point $y \in \Delta$, and sets $\mathrm{H}(I) = y$ by putting (I, y) in table T, I is then added into HU and a symbol '1' is returned.
3. CORR(I): If $I \notin \mathrm{HU}$, $\perp$ is returned. Otherwise, $\mathcal{B}$ asks INV to generate a witness w for H(I) and returns w to $\mathcal{A}$. I is then deleted from HU and added into CU.
4. CONV(I): If $I \notin \mathrm{HU}$, $\perp$ is returned. Otherwise, $\mathcal{B}$ runs the simulation algorithm $\mathcal{SIM}$ in Def. 4 to generate a simulated transcript and returns it to $\mathcal{A}$.

If $\mathcal{A}$ successfully impersonates a user $\mathrm{I_b}$ that is created but not corrupted (i.e. $\mathrm{H(I_b)}$ is returned by RAM, but the witness of $\mathrm{H(I_b)}$ is still not known to $\mathcal{B}$) with probability ϵ, by the Reset Lemma (Appendix A) and the special soundness, $\mathcal{B}$ can extract a witness of $\mathrm{H(I_b)}$ with probability at least $(\epsilon - 2^{-\ell(k)})^2$. Thus $\mathcal{B}$ breaks the weak-one-more resistance of $\mathbf{R}$ with a non-negligible probability. □

By applying the generic construction above, we can derive the id-imp-pa securiy of GQ-IBI [11], Sh-IBI [16] under the RSA assumption, and Hs-IBI [12], ChCh-IBI [8] under the CDH assumption.

4 Transforming to a Generic IBI Scheme Secure Against Active and Concurrent Attacks

To construct an IBI scheme secure against active and concurrent attacks (namely, id-imp-aa secure and id-imp-ca secure), we do not need to do so from scratch. Interestingly, as described in this section, we only need to replace the trapdoor weak-one-more relation of our generic construction described in Sec. 3 with a *trapdoor strong-one-more relation* and the HVZK proof with a *witness indistinguishable* proof, for transforming our generic construction secure against passive attacks (i.e. id-imp-pa) to a generic IBI scheme secure against active and concurrent attacks.

4.1 Trapdoor Strong-One-More Relations

Definition 5 (Trapdoor Strong-One-More Relation). *A family of trapdoor strong-one-more relations $\mathcal{R}$ is a triple of PPT algorithms* ($\mathbf{Gen}'$, $\mathbf{Ver}'$, $\mathbf{Inv}'$) *such that the following properties hold:*

1. $\mathbf{Gen}'$**:** *On input 1^k, where $k \in \mathbb{N}$ is the security parameter, the probabilistic polynomial-time algorithm $\mathbf{Gen}'$ outputs $(\langle \mathbf{R} \rangle, t)$ where $\langle \mathbf{R} \rangle$ denotes the description of a binary relation $\mathbf{R}$ on $W \times \Delta$ and t is the trapdoor information of $\mathbf{R}$.*
2. $\mathbf{Ver}'$**:** *For every $k \in \mathbb{N}$, $(\langle \mathbf{R} \rangle, t) \leftarrow \mathbf{Gen}'(1^k)$, $\mathbf{Ver}'(1^k, \langle \mathbf{R} \rangle, x, y) = 1$ if and only if $(x, y) \in \mathbf{R}$.*
3. $\mathbf{Inv}'$**:** *It is a (probabilistic or deterministic) polynomial-time algorithm such that on input $(1^k, \langle \mathbf{R} \rangle, y, t)$, it outputs an x such that $(x, y) \in \mathbf{R}$ for any $y \in \Delta$.*
4. ***Non triviality:*** *$|\Delta|$ is greater than $p(k)$ where $p(\cdot)$ is any positive polynomial.*
5. ***Strong-one-more resistance:*** *It is defined by a game. The adversary $\mathcal{A}$ is given $1^k, \langle \mathbf{R} \rangle$ as input where $(\langle \mathbf{R} \rangle, t) \leftarrow \mathbf{Gen}'(1^k)$ and access to two oracles:*
   (a) A challenge oracle RAM *that on any input returns a new random target point $y \in \Delta$.*
   (b) An inversion oracle INV *that on any input y:*
      i. If y is from the output of RAM, INV *returns a witness of y, and the same witness is returned if y is queried again later.*
      ii. If y is not from the output of RAM, *a symbol $\perp$ is returned indicating that the input is invalid.*

   The adversary wins if he can find a pair $(x', y') \in \mathbf{R}$ such that y' is one output of RAM *but (x', y') does not appear in the input/output pairs of the inversion oracle (i.e. the adversary can find one more distinct pair than the pairs given by the inversion oracle)*[2]*. A relation is a trapdoor strong-one-more relation if the probability to win the game is negligible in k for any polynomial-time adversary.*

[2] There are two cases: in the first case, y' has never been queried to the inversion oracle; in the second case, the inversion oracle has returned a witness x of y' before, but $x' \neq x$.

In the following, we describe some primitives that can be used to construct trapdoor strong-one-more relations.

Factoring Assumption Based. A Blum-Williams generator is a modulus generator that returns Blum-Williams (BW) moduli N [18,5], meaning that $N = pq$ with $p \equiv q \equiv 3 \bmod 4$. Let $QR_N = \{x^2 \bmod N | x \in \mathbb{Z}_N^*\}$ be the set of all quadratic residues modulo N. It is known that if N is a BW modulus, then squaring is a permutation on QR_N. Let $\mathbb{Z}_N^*[+1] = \{x \in \mathbb{Z}_N^* | Jac_N(x) = +1\}$ where $Jac_N(x)$ is the Jacobi symbol of x with respect to N. We also know that if N is a BW modulus, -1 is a non-square modulo N with Jacobi symbol $+1$, and for every element $x \in \mathbb{Z}_N^*[+1]$, either x or $-x$ is a square modulo N.

We construct a trapdoor strong-one-more relation as follows: on input 1^k, $\mathbf{Gen'}$ runs the Blum-Williams generator to generate (N, p, q). (p, q) is the trapdoor for relation $\mathbf{R}^{SQ} = \{(X, Y) \in \mathbb{Z}_N^* \times \mathbb{Z}_N^*[+1] : X > (N-1)/2; Y \equiv \pm X^2 \bmod N\}$. On input $Y \in \mathbb{Z}_N^*[+1]$, $\mathbf{Inv'}$ uniformly chooses an $X \in \mathbb{Z}_N^*$ over the two square roots (greater than $(N-1)/2$) of $\pm Y$ (remember either Y or $-Y$ is a square).

Theorem 5. *Assume the factoring problem is hard,* $\mathbf{R}^{SQ}$ *is a trapdoor strong-one-more relation.*

Proof. The proof is by contradiction. Assume there exists an adversary $\mathcal{A}$ which can break the strong-one-more resistance, then we can build an adversary $\mathcal{B}$ to factor N. Here is the simulation.

When $\mathcal{A}$ asks a challenge query, $\mathcal{B}$ uniformly selects an $x \in \mathbb{Z}_N^*$ at random such that $x > (N-1)/2$, and returns $y \overset{R}{\leftarrow} \pm x^2 \bmod N$ to $\mathcal{A}$. When $\mathcal{A}$ asks the inversion query on y, $\mathcal{B}$ returns x to $\mathcal{A}$. If $\mathcal{A}$ aborts, $\mathcal{B}$ also aborts.

Suppose $\mathcal{A}$ wins the strong-one-more resistance game, then one of the following two events must occur[3]. $\mathbf{E}_1$: $\mathcal{A}$ outputs a witness x' for a challenge y that has appeared in an inversion query. Denote the witness selected by $\mathcal{B}$ in the challenge query by x, then $x' \neq \pm x$, and $\mathcal{B}$ is able to factor N. $\mathbf{E}_2$: $\mathcal{A}$ outputs a witness x' for a challenge y that has not appeared in an inversion query. Denote the witness selected by $\mathcal{B}$ in the challenge query by x, if $x' = x$, $\mathcal{B}$ aborts with failure. Otherwise, $x' \neq \pm x$, and $\mathcal{B}$ is able to factor N. Since x is uniformly selected at random, $\Pr[x' \neq \pm x] = 1/2$.

Thus, if $\mathcal{A}$ can break the strong-one-more resistance with probability ϵ, $\mathcal{B}$ can factor N with probability at least $\epsilon/2$. □

RSA Assumption Based. On input 1^k, the RSA key generator outputs a modulus N that is the product of two distinct odd primes p, q where $|p| = |q| = k/2$, and exponents e, d such that $ed \equiv 1 \bmod \varphi(N)$ where $\varphi(N) = (p-1)(q-1)$ is the Euler's totient function. A prime-exponent RSA key generator only outputs keys with e prime. The RSA problem is hard if

$$\mathbf{Adv}_A^{rsa}(k) = \Pr[(N, e, d) \overset{R}{\leftarrow} K_{rsa}(1^k); y \overset{R}{\leftarrow} Z_N^*; x \leftarrow A(1^k, N, e, y) : x^e \equiv y \bmod N]$$

is negligible in k for all polynomial-time algorithm A.

[3] There is a chance that y and -y are returned in two challenge queries, but this only happens with a negligible probability.

We construct a trapdoor strong-one-more relation as follows: on input 1^k, $\mathbf{Gen}'$ first runs the prime-exponent RSA key generator to generate (N, e, d) such that $e > 2^{\ell(k)}$ where $\ell(k)$ is super-logarithmic in k, and then randomly picks $g \overset{R}{\leftarrow} \mathbb{Z}_N^*$. (N, d) is the trapdoor for relation $\mathbf{R}^{RSA} = \{((x_1, x_2), Y) \in (\mathbb{Z}_e \times \mathbb{Z}_N^*) \times \mathbb{Z}_N^* : g^{-x_1} x_2^{-e} \equiv Y \bmod N$. On input $Y \in \mathbb{Z}_N^*$, $\mathbf{Inv}'$ randomly chooses $x_1 \overset{R}{\leftarrow} \mathbb{Z}_e$, and then calculates $x_2 = (g^{x_1} Y)^{-d} \bmod N$.

Theorem 6. *Assume the RSA problem is hard,* $\mathbf{R}^{RSA}$ *is a trapdoor strong-one-more relation.*

Proof (Sketch). Assume there exists an adversary $\mathcal{A}$ which can break the strong-one-more resistance with probability ϵ, then we can build another adversary $\mathcal{B}$ which solves the RSA problem with probability at least $(1 - 1/e)\epsilon$.

Given the RSA challenge y, adversary $\mathcal{B}$ sets $g = y$ and simulates the strong-one-more resistance game as follows:

When $\mathcal{A}$ asks a challenge query, $\mathcal{B}$ randomly selects $x_1 \overset{R}{\leftarrow} \mathbb{Z}_e, x_2 \overset{R}{\leftarrow} \mathbb{Z}_N^*$, and returns $Y = g^{-x_1} x_2^{-e} \bmod N$ to $\mathcal{A}$. When $\mathcal{A}$ asks the inversion query on Y, $\mathcal{B}$ returns (x_1, x_2) to $\mathcal{A}$. If $\mathcal{A}$ aborts, $\mathcal{B}$ also aborts.

If $\mathcal{B}$ can obtain two different witnesses (x_1, x_2) and $(\hat{x}_1, \hat{x}_2)$ for the same challenge Y, since e is prime and $0 < |x_1 - \hat{x}_1| < e$, two integers a, b can be found such that $a(x_1 - \hat{x}_1) + be = 1$, then $\mathcal{B}$ outputs $g^b (x_2 \hat{x}_2^{-1})^a \bmod N$. By analyzing the probability of two similar events $\mathbf{E}_1$ and $\mathbf{E}_2$ in the proof of Theorem 5, we can see that $\mathcal{B}$ breaks the RSA problem with probability at least $(1 - 1/e)\epsilon$. □

Strongly Unforgeable Signature Based. Let $\mathcal{SIG}$ be defined as in Sec. 3.1, $\mathcal{SIG}$ is strongly unforgeable [1] under known message attack [10] (seuf-kma) if no polynomial-time adversary is feasible to produce a message-signature pair (m, σ) such that (m, σ) is not in his known list of message-signature pairs.

By using the same construction as in Sec. 3.1, we can get the following theorem.

Theorem 7. *If* $\mathcal{SIG}$ *is strong unforgeable under known message attack, and for any message* $m \in \mathcal{MS}$[4] *there are more than one valid signatures,* $\mathbf{R}^{SIG}$ *is a trapdoor strong-one-more relation.*

Proof (Sketch). Assume there exists an adversary $\mathcal{A}$ which can break the strong-one-more resistance, we build a forger $\mathcal{F}$ as follows.

Suppose $\mathcal{A}$ asks at most $Q(k)$ challenge queries. $\mathcal{F}$ first gets $Q(k)$ message-signature pairs (the messages are not chosen by him). Then $\mathcal{F}$ answers $\mathcal{A}$'s challenge/inversion queries by simply sending back the corresponding message/ signature.

By analyzing the probability of two similar events $\mathbf{E}_1$ and $\mathbf{E}_2$ in the proof of Theorem 5, we can see that $\mathcal{F}$ has a non-negligible probability to win the strong unforgeability game. □

[4] Again, we assume $|\mathcal{MS}| \geq 2^{\ell(k)}$ where $\ell(k)$ is super logarithmic in k.

4.2 Transformation to a Generic IBI Scheme Secure Against Active and Concurrent Attacks

With reference to the generic construction of IBI schemes with id-imp-pa security described in Sec. 3.2, we use it to construct a generic IBI scheme with id-imp-aa security and id-imp-ca security, by replacing the original $\mathbf{R}$ with a trapdoor strong-one-more relation (Def. 5) and $(\tilde{\mathbf{P}}, \tilde{\mathbf{V}})$ with a non-trivial interactive proof with *witness dualism* defined below.

Definition 6 (Witness Dualism). *Let* $\mathbf{R}$ *be a trapdoor strong-one-more relation. We say that* $\mathbf{R}$ *has* Witness Dualism *if there exists a non-trivial interactive proof system* (P, V) *with special soundness such that for every* $y \in \Delta$*, and for every* $x \in W(y)$*, there exists at least one* $x' \in W(y)$ *such that* $x' \neq x$ *and for any verifier* V' *and any auxiliary input* z *for* V'*, the ensembles,* $V'_{P(y,x)}(y, z)$ *and* $V'_{P(y,x')}(y, z)$*, generated as* V'*s view of the interactive proof, are indistinguishable.*

The notion of Witness Dualism is related to *Witness Indistinguishability* [9]. For witness dualism, given a witness x of y, the notion only requires it to be indistinguishable with another witness x', rather than with all other witnesses in $W(y)$. Hence it is a weaker notion when compared with witness indistinguishability.

Theorem 8. *Let* $\mathbf{R}$ *be a trapdoor strong-one-more relation which has Witness Dualism, if the challenge length* $\ell(k)$ *of the interactive proof system is super logarithmic in* k*, then the generic IBI scheme in Sec. 3.2 (replace the HVZK proof by (P, V)) is* id-imp-aa *and* id-imp-ca *secure in the random oracle model.*

Proof. Given an adversary $\mathcal{A}$ that can break the IBI scheme, we construct an adversary $\mathcal{B}$ which breaks the strong-one-more resistance of the underlying trapdoor strong-one-more relation.

$\mathcal{B}$ simulates the id-imp-aa (id-imp-ca) game by setting the $mpk = \langle \mathbf{R} \rangle$. $\mathcal{B}$ maintains two user lists HU and CU, which are empty at the beginning. $\mathcal{B}$ also maintains a table T, each row of T contains an identity I and the value of $\mathrm{H}(I)$ and a witness of $\mathrm{H}(I)$. T is also empty at the beginning. $\mathcal{B}$ answers $\mathcal{A}$'s oracle queries as follows:

1. H-query: On input $I \in \{0, 1\}^*$, $\mathcal{B}$ checks if I is in table T. If I is not in T, $\mathcal{B}$ asks RAM to get a random point $y \in \Delta$, then $\mathcal{B}$ sets $\mathrm{H(I)} = y$ by putting $(I, y, \perp)$ in table T[5]. If I is already in table T, the existing value is returned.
2. INIT(I): If $I \in \mathrm{HU} \cup \mathrm{CU}$, $\perp$ is returned. Otherwise, $\mathcal{B}$ checks if I is in table T. If I is in table T, I is added into HU and a symbol '1' is returned. Otherwise, $\mathcal{B}$ asks RAM to get a random point $y \in \Delta$, then $\mathcal{B}$ sets $\mathrm{H}(I) = y$ by putting $(I, y, \perp)$ in T, I is then added into HU and a symbol '1' is returned.
3. CORR(I): If $I \notin \mathrm{HU}$, $\perp$ is returned. Otherwise, $\mathcal{B}$ finds the row corresponding to I in table T. If the witness is unknown, $\mathcal{B}$ asks the inversion oracle for a witness x of $\mathrm{H}(I)$, and replaces the $\perp$ symbol in that row by x. $\mathcal{B}$ returns x to $\mathcal{A}$. I is then deleted from HU and added into CU.

[5] The symbol "$\perp$" denotes the value is unknown yet.

4. PROV(I): If $I \notin \mathrm{HU}$, $\perp$ is returned. Otherwise, $\mathcal{B}$ finds the row corresponding to I in table T and retrieves x. If the witness is unknown, $\mathcal{B}$ asks the inversion oracle for a witness x of H(I), and replaces the $\perp$ symbol in that row by x. then $\mathcal{B}$ runs a copy of P with initial state x.

Finally, if $\mathcal{A}$ can successfully impersonate a user $\mathrm{I_b}$ that is created but not corrupted (i.e. $\mathrm{H(I_b)}$ is returned by RAM, but $\mathcal{A}$ does not ask for its witness), by the Reset Lemma (Appendix A) and the special soundness, $\mathcal{B}$ can extract a witness x_b of $\mathrm{H(I_b)}$ with probability at least $(\epsilon - 2^{-\ell(k)})^2$.

If $\mathcal{B}$ has never asked the inversion oracle for a witness of $\mathrm{H(I_b)}$, $\mathcal{B}$ successfully breaks the strong-one-more resistance. Otherwise, because of the Witness Dualism, with probability at least 1/2, the witness extracted (with the help of $\mathcal{A}$) is different from the one in table T (by following the same proof of [9], witness dualism is also preserved under concurrent composition).

Thus $\mathcal{B}$ breaks the strong-one-more resistance of the underlying trapdoor strong-one-more relation with probability at least $1/2(\epsilon - 2^{-\ell(k)})^2$. □

By applying the RSA-based trapdoor strong-one-more relation together with a witness indistinguishable interactive proof with special soundness [15], we can derive the Okamoto-RSA-IBI scheme [15,2] that is imp-ca secure.

In the next section, we construct a concrete IBI scheme that is imp-ca secure from strong unforgeable signature schemes.

5 A Concrete IBI Scheme Secure Against Concurrent Attacks

In this section, we construct an IBI scheme from the Katz-Wang signature scheme [13] which is shown to be strongly unforgeable under the DDH assumption [6] for any message space $\mathcal{MS} \subset \{0,1\}^*$. Let $\mathbb{G}$ be a cyclic group of prime order q with generator g, $H : \{0,1\}^* \to \{0,1\}^k$ and $H' : \{0,1\}^* \to \{0,1\}^k$ be hash functions which are assumed to behave as independent random oracles for security analysis. Let $k \in \mathbb{N}$ be the security parameter and $k < |q|$. We first review the signature scheme due to Katz and Wang.

The Katz-Wang Signature Scheme: To generate a public/secret key pair, $h \in \mathbb{G}$ and $x \leftarrow \mathbb{Z}_q^*$ are first chosen randomly. $y_1 = g^x$ and $y_2 = h^x$ are then computed and the public key is set to $PK = (h, y_1, y_2)$ and the secret key to x. To sign a message m, the following steps are carried out.

1. Choose random $r \leftarrow \mathbb{Z}_q$.
2. Compute $A = g^r$, $B = h^r$, and $c = H'(A, B, m)$.
3. Compute $s = cx + r \bmod q$ and set signature $\sigma = (c, s)$.

To verify the signature, $A = g^s y_1^{-c}$ and $B = h^s y_2^{-c}$ are computed and if $c = H'(A, B, m)$, the signature is valid.

The IBI Scheme: Based on the Katz-Wang signature scheme, we build an IBI scheme with id-imp-ca security. The scheme is described in Fig. 1. Next, we prove

MKGen: Choose a cyclic group $\mathbb{G}$ of prime order q with generator g such that $|q| > k$. Choose hash functions $H : \{0,1\}^* \rightarrow \{0,1\}^k$ and $H' : \{0,1\}^* \rightarrow \{0,1\}^k$. Randomly choose $h \in \mathbb{G}^*$ and $x \leftarrow \mathbb{Z}_q^*$. Compute $y_1 = g^x$ and $y_2 = h^x$. Set master public key to $MPK = (\mathbb{G}, q, g, h, y_1, y_2, H, H')$ and master secret key to x.

UKGen: Randomly choose $r \leftarrow \mathbb{Z}_q$, compute $A = g^r$, $B = h^r$, $c = H'(A, B, H(I))$, and $s = cx + r \bmod q$. The user secret key is $\sigma = (c, s)$.

User Identification Protocol: randomly choose $r' \in \mathbb{Z}_q$, and compute $A \leftarrow g^s y_1^{-c}$, $B \leftarrow h^s y_2^{-c}$, $A' \leftarrow g^{r'}$, $B' \leftarrow h^{r'}$, $c' \leftarrow H'(A', B', H(I))$.

Prover P (c, s) **Verifier V** (MPK, I)

$\lambda \overset{R}{\leftarrow} \mathbb{Z}_q$, $T_1 \leftarrow g^\lambda$, $T_2 \leftarrow h^\lambda$
$z' \overset{R}{\leftarrow} \mathbb{Z}_q, \alpha' \overset{R}{\leftarrow} \mathbb{Z}_q$
$T_1' \leftarrow g^{z'}(A' y_1^{c'})^{-\alpha'}$
$T_2' \leftarrow h^{z'}(B' y_2^{c'})^{-\alpha'}$

$$\xrightarrow{A, B, A', B', T_1, T_2, T_1', T_2'}$$

$c \leftarrow H'(A, B, H(I))$
$c' \leftarrow H'(A', B', H(I))$
$U \leftarrow A y_1^c, R \leftarrow B y_2^c$
$U' \leftarrow A' y_1^{c'}, R' \leftarrow B' y_2^{c'}$
$\alpha_0 \overset{R}{\leftarrow} \mathbb{Z}_q$

$$\xleftarrow{\alpha_0}$$

$\alpha \leftarrow \alpha_0 - \alpha' \bmod q$
$z \leftarrow \lambda + \alpha s \bmod q$

$$\xrightarrow{z, z', \alpha, \alpha'}$$

$\alpha + \alpha' \overset{?}{=} \alpha_0$
$T_1 \overset{?}{=} g^z U^{-\alpha}$
$T_2 \overset{?}{=} h^z R^{-\alpha}$
$T_1' \overset{?}{=} g^{z'} U'^{-\alpha'}$
$T_2' \overset{?}{=} h^{z'} R'^{-\alpha'}$

Fig. 1. The IBI scheme based on Katz-Wang signature scheme

its security by following our framework described in the previous section. It can be seen that the user identification protocol in Fig. 1 is actually a proof that the prover knows at least one of two valid signatures.

Lemma 1. *The user identification protocol in Fig. 1 has special soundness.*

Proof. Given two successful conversations where **V** outputs '*accept*':

$$(A, B, A', B', T_1, T_2, T_1', T_2', \alpha_0, z, z', \alpha, \alpha')$$
$$(A, B, A', B', T_1, T_2, T_1', T_2', \hat{\alpha}_0, \hat{z}, \hat{z}', \hat{\alpha}, \hat{\alpha}')$$

such that $\alpha_0 \neq \hat{\alpha}_0$, it must be the case that at least one of the inequalities $\alpha \neq \hat{\alpha}$ and $\alpha' \neq \hat{\alpha}'$ take place. For example, if $\alpha \neq \hat{\alpha}$, (s, c) can be obtained from $s = (z - \hat{z})(\alpha - \hat{\alpha})^{-1}$ and $c = H'(A, B, H(I))$. Hence we can see that at least one of (s, c) and (s', c') can be extracted. □

Lemma 2. *The user identification protocol in Fig. 1 is witness dualism (Def. 6).*

Proof. For the two valid signatures $\sigma = (c, s)$ (with respect to r) and $\sigma' = (c', s')$ (with respect to r') of the message $H(I)$, the ensembles, $V'_{P(y,\sigma)}(y, z)$ (with illusive witness σ') and $V'_{P(y,\sigma')}(y, z)$ (with illusive witness σ), generated as V's view of the protocol, are identically distributed, where y refers to (MPK, I) and z is any auxiliary input for V'. □

Remark: In this IBI scheme, the user is required to use the same illusive witness (c', s') (with respect to r') in all the conversations, and the 'Dual Witness' of (c, s) is exactly (c', s').

By combining these two lemmas and Theorem 8, we obtain the following theorem.

Theorem 9. *The IBI scheme in Fig. 1 is secure against impersonation under concurrent attacks (id-imp-ca).*

References

1. Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.
2. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer, 2004.
3. Mihir Bellare and Adriana Palacio. GQ and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002.
4. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security (CCS)*, pages 62–73, 1993.
5. Manuel Blum. Coin flipping by telephone. In *CRYPTO 1981*, pages 11–15, 1981.
6. D. Boneh. The decision Diffie-Hellman problem. In *Proc. of the Third Algorithmic Number Theory Symposium*, pages 48–63. Springer-Verlag, 1998. LNCS 1423.
7. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.
8. Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In *Public Key Cryptography 2003*, pages 18–30, 2003.
9. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing (STOC)*, 1990.
10. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, April 1988.
11. Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 1990.
12. Florian Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography, SAC 2002*, pages 310–324, 2002.

13. Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *ACM CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 155–164, 2003.
14. Kaoru Kurosawa and Swee-Huay Heng. From digital signature to id-based identification/signature. In *Public Key Cryptography 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 2004.
15. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1993.
16. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1985.
17. Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2002.
18. Hugh C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Trans. Inf. Theory*, 26(6), 1980.

A Reset Lemma

Lemma 3 (Reset Lemma [3]). *Let CP be a prover in a canonical IBI scheme with challenge set* ChSet *and challenge length $\ell(\cdot)$. St_V and St_{CP} are the initial states of the verifier and CP, respectively. Let $acc(St_{CP}, St_V)$ be the probability that the verifier accepts, and $res(St_{CP}, St_V)$ the probability that the following reset experiment returns 1:*

```
Choose random tape $\rho$ for CP; $(\mathrm{Cmt}, St'_{CP}) \leftarrow CP(St_{CP}, \rho)$
$\mathrm{Ch}_1 \xleftarrow{R} \mathrm{ChSet}(St_V); (\mathrm{Rsp}_1, St''_{CP}) \leftarrow CP(\mathrm{Ch}_1, St'_{CP})$;
$d_1 \leftarrow \mathbf{Dec}(St_V, \mathrm{Cmt}\|\mathrm{Ch}_1\|\mathrm{Rsp}_1)$
$\mathrm{Ch}_2 \xleftarrow{R} \mathrm{ChSet}(St_V); (\mathrm{Rsp}_2, St'''_{CP}) \leftarrow CP(\mathrm{Ch}_2, St'_{CP})$;
$d_2 \leftarrow \mathbf{Dec}(St_V, \mathrm{Cmt}\|\mathrm{Ch}_2\|\mathrm{Rsp}_2)$
If ($d_1 = 1$ and $d_2 = 1$ and $\mathrm{Ch}_1 \neq \mathrm{Ch}_2$) then return 1 else return 0
```

Then,

$$res(St_{CP}, St_V) \geq (acc(St_{CP}, St_V) - 2^{-\ell(k)})^2.$$

Tweaking TBE/IBE to PKE Transforms with Chameleon Hash Functions

Rui Zhang

Research Center for Information Security (RCIS)
National Institute of Advanced Industrial Science and Technology (AIST)
r-zhang@aist.go.jp

Abstract. We present two transforms to acquire chosen ciphertext security from tag based techniques. The first one requires the separability of underlying primitives. By separability, informally, we mean the encryption algorithm has special structures and can process the identity and the message independently. Compared with generic transforms [8], it significantly reduces the ciphertext size overhead with only marginal computation cost. Compared with [11], the only known technique which directly achieves chosen ciphertext secure public key encryption from separable identity based primitives, it only requires selective-Tag/ID security of underlying primitives. Our second transform is less efficient but performs generically. Both transforms preserve the public verifiability of underlying primitives, and can be extended to hierarchical identity based encryption (HIBE) and threshold settings. As an independent interest, we also investigate the security requirements of chameleon hash functions to build strongly unforgeable one-time signatures.

1 Introduction

Indistinguishability against chosen ciphertext attack (CCA) is the standard security notion for public key encryption (PKE) schemes, which was established in [19,25,27,17,4]. While it is comparatively easy in the random oracle model [5], constructing a CCA-secure PKE is usually hard in the standard model. Up to now, there are only a few methods known to solve the problem in the standard model: Naor-Yung paradigm [25,17,28], universal hash proof [15,16,24], and an approach from tag/identity based encryption (TBE/IBE) techniques [9,13,10,22,8].

Along the last trend, most recently, Boyen, Mei and Waters [11] proposed two efficient techniques, referred as the BMW IBE transform and the BMW key encapsulation mechanism (KEM) transform, both of which achieve CCA-security from separable IBEs. Here, by separability, informally, we mean the encryption algorithm has special structures and can process the identity and the message independently. Another direct construction of KEM from a separable TBE was discovered by Kiltz [22].

Let's review the rough idea of the BMW PKE transform, which is based on separable IBEs. Write the encryption algorithm into two independent functions,

$f_1(params, m, r)$ and $f_2(params, id, r)$, where $params$ is the public system parameter, m is a plaintext, id is an arbitrary identity and r is the internal coin by the encryption algorithm. The public key of the PKE is then $params$ and the secret key is msk, which is the master secret key of the according IBE. For encryption, one computes $u = f_1(pk, m, r)$ then hashes u into an "identity" id by a collision resistant (or injective) hash function. The ciphertext is then $c = \langle u, v \rangle$, where $v = f_2(pk, id, r)$. Adaptive chosen-ID security is required here because the challenge identity is decided after the adversary selects the pair of chosen plaintexts. For decryption on $c = \langle u, v \rangle$, first reconstruct id from u, and decrypt c using the IBE decryption algorithm under identity id. Intuitively, to attack this PKE, an adversary may submit c' which maybe regarded as a ciphertext with different identity id' for the underlying IBE, however, this will not provide useful information to it, because of the semantic security of underlying IBEs.

Though the BMW PKE transform is very efficient in ciphertext size, the range of its application is quite limited, because it demands adaptive chosen-ID security. Additionally, note that the BMW PKE transform doesn't give a direct security reduction to that of the underlying IBE, since the simulator is supposed to submit the challenge identity with the pair of chosen plaintexts to its challenger, and the identity id is in fact determined by the challenge ciphertext. One may find it difficult to prove the security of the resulting PKE scheme sometimes, e.g., the BMW PKE transform + Gentry IBE [18].

We note that selective-Tag/ID security suffices for the TBE/IBE to KEM transform, because the challenge session key, thus the challenge identity, can be chosen even before interacting with the adversary. For the time being, we feel it no rush to build new KEMs, since BMW KEM [11] and Kiltz KEM [22] are already efficient enough for most applications requiring chosen ciphertext security at hand. On the other hand, we argue that such KEMs may be not publicly verifiable, thus may not fit into threshold settings in their original forms, which forms another motivation of this work.

Sketch of Our Ideas. Instead of a normal collision resistant hash function, we use a collision resistant chameleon hash function to construct the identity id. Interestingly, this simple modification achieves CCA-secure PKEs with public verifiability, whose security can be reduced to underlying IBEs. Recall a collision resistant chameleon hash function hashes a message x together with auxiliary randomness w, such that with a trapdoor, one can efficiently find a collision on (x', w'), such that (x, w) and (x', w') will be hashed to the same vale. While without the trapdoor, it is computationally infeasible to do so.

In the new strategy, for setup, the simulator generates a pair of public/secret keys (hk, td) for a chameleon hash function. The simulator then commits to an identity id^*, hashed from a dummy challenge ciphertext $\bar{u}$ with some randomness $\bar{r}$, and submits id^* to its challenger, who generates the public key and returns $params$ to the simulator. The public key of PKE now consists of $params$ and an additional hash key hk. For correctly reconstructing the identity id, the randomness r for hash function should be also appended to the ciphertext. Later upon receiving the real challenge ciphertext $c = \langle u^*, v^* \rangle$ from the IBE challenger, it finds a collision (u^*, r^*) using td, i.e., both (u^*, r^*) and $(\bar{u}, \bar{r})$ will be hashed to id^*. In this way, the simulator can embed its own challenge into the challenge ciphertext for an IBE adversary, in other words, the security of the PKEs can be reduced to the underlying IBEs. Note that the BMW PKE transform doesn't give such a direct security reduction, since the target id is in fact determined by the challenge ciphertext itself. Alternatively, we can construct an adversary against the chameleon hash. Because this time the simulator does not have the trapdoor, while the challenge is partially determined according to the adversary, the chameleon hash needs to be collision resistant.

Moreover, as a supplement to our initial idea, a similar observation as [22] shows that a TBE scheme with selective-Tag and chosen ciphertext security suffices for the job. Since TBE is a possibly weaker primitive than IBE where the extract algorithm is not explicitly used, one may obtain efficient schemes using corresponding transforms. In fact, the BMW PKE scheme, an IBE private key is never generated, since it is more efficient to decrypt directly using the master key. Instantiate the method with Kiltz TBE [22], we obtain a corresponding PKE scheme with publicly verifiability and without pairings. Interestingly, all constructions of IBE schemes in the standard model in fact satisfy separability.

We go on to show how to remove the limitation of separability by sacrificing a little efficiency of above transform. Actually, a chameleon hash may also serve as a one-time signature by hashing with online key generation and switching. With similar idea of [13], one acquires a generic transform based on TBE/IBE and chameleon hash functions.

Compared with the BMW PKE transform, our transforms result in a slightly larger ciphertext overhead. We believe this is tolerable, since our requirement of underlying primitive is much weaker. Both transforms preserve the public verifiability of underlying primitives. Especially it transforms CPA-secure $(\ell{+}1)$-level HIBEs to CCA-secure ℓ-level HIBEs. It is worth noting that the chosen ciphertext security of resulting ℓ-level HIBE can be reduced to the semantic security of $(\ell{+}1)$-level HIBEs similarly.

Finally, as an independent interest, we investigate the necessary security requirements of building one-time strongly unforgeable signatures from chameleon hash functions. Although this intuition hides behind a lot of work, but it has not been formalized before.

Related Work. First CCA-secure construction of public key encryption, also known as Naor-Yung paradigm, was due to [25,17], and further generalized by Sahai [28], which is quite inefficient. Cramer and Shoup gave the first practical CCA-secure public key encryption scheme [15] in the standard model and later generalized to universal hash proof system [16]. These two approaches utilize certain non-interactive proofs of "well-formness". Apart from the above, Canetti, Halevi and Katz [13] presented a generic construction from weak IBEs, where no such non-interactive proofs are needed. Boneh and Katz [10] further improved the efficiency of [13]. The full version of [13,10] appeared as [8].

The notion of identity based encryption was due to Shamir, whose original intention was to simplify the management of public key certificates. Boneh and Franklin [9] defined the first formal security notion for identity based encryption, namely indistinguishability against chosen-ID and chosen ciphertext attack (IND-ID-CCA) and gave the first functional scheme based on pairings. Independently, Cocks [14] proposed another identity based encryption based on decisional quadratic residue assumption. Canetti, Halevi and Katz [12] defined another useful security notion with weaker attack model, namely security against selective-ID attack and chosen plaintext/ciphertext attack (IND-sID-CPA/CCA). On the other hand, Gentry and Silverberg generated the notion of identity based encryption to hierarchical identity based encryption (HIBE).

Boneh and Boyen proposed two efficient IBE schemes (referred as BB1 and BB2) with selective-ID security [6]. They further generalized to adaptive chosen-ID security [7], but the scheme is quite inefficient. Waters subsequently presented the first practical IBE scheme [30]. Most recently, Gentry gave a more efficient IBE scheme, which however, relies on a very strong assumption [18].

Most recently, independently from our work, Abe, Cui, Imai and Kiltz [1] considered building tag-KEM [2] from special ID-Based KEMs (IBKEMs), called partitioned IBKEMs, which are essentially the same as separability. A tag-KEM is more flexible in building secure hybrid encryptions, which differs slightly from our goal of building PKEs, however. We remark that we are additionally interested in directly building PKEs, from black-box TBEs/IBEs using chameleon hash functions.

Organizations. The rest of the paper will be arranged as follows: we give some definitions in Section 2, then give our transforms and analyze their securities in Section 3 and Section 4. We explain applications of our results in Section 5. Finally in Section 6, we give detailed comparisons among some typical schemes.

2 Preliminary

In this section, we give some notations and definitions, then review some hard problems related to pairings.

Notations. If x is a string, let $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If S is a set then $s \leftarrow S$ denotes the operation of picking an element s of S uniformly at random. We write $z \leftarrow \mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $(x, y, \ldots)$ and an output z. Denote $x||y$ as the string concatenation of x and y. If $k \in \mathbb{N}$, a function $f(k)$ is negligible if $\exists\ k_0 \in \mathbb{N}$, $\forall\ k > k_0$, $f(k) < 1/k^c$, where $c > 0$ is a constant.

2.1 Public Key Encryption

A public key encryption scheme consists of three algorithms $\mathcal{PKE} = (\mathsf{PKEkg}, \mathsf{PKEenc}, \mathsf{PKEdec})$. The randomized key generation algorithm taking a security parameter k as the input, generates a public key pk and a corresponding secret key sk, which is denoted as $(pk, sk) \leftarrow \mathsf{PKEkg}(1^{\mathsf{k}})$. The randomized encryption algorithm taking pk and a plaintext m taken from the message space as inputs, with internal coin flipping r, outputs a ciphertext c, which is denoted as $c \leftarrow \mathsf{PKEenc}(pk, m, r)$, in brief $c \leftarrow \mathsf{PKEenc}(pk, m)$. The deterministic decryption algorithm taking sk and a ciphertext c as input, outputs the corresponding m, or "$\perp$" (indicating invalid ciphertext), denoted as $m \leftarrow \mathsf{PKEdec}(sk, c)$. We require a PKE scheme should satisfy the standard correctness requirement, namely for all $(pk, sk) \leftarrow \mathsf{PKEkg}(1^k)$ and all m, $\mathsf{PKEdec}(sk, \mathsf{PKEenc}(pk, m)) = m$.

IND-CCA-Security. We say a public key encryption scheme is (ϵ, q, T)-IND-CCA secure, if the advantage of any adversary $\mathcal{A}$ with at most q queries to a decryption oracle $\mathcal{DO}$, is at most ϵ within time T in the following experiment.

$$\begin{aligned} \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE},\mathcal{A}} \overset{\mathsf{def}}{=} \Pr[&(pk, sk) \leftarrow \mathsf{PKEkg}(1^k); (m_0, m_1, s) \leftarrow \mathcal{A}^{\mathcal{DO}}(pk); \\ &b \leftarrow \{0,1\}; c^* \leftarrow \mathsf{PKEenc}(pk, m_b); b' \leftarrow \mathcal{A}^{\mathcal{DO}}(c^*, s) : b' = b] - 1/2 \end{aligned}$$

where $\mathcal{DO}$ returns the corresponding decryption result on a query on ciphertext c, whereas $\mathcal{A}$ is forbidden to query c^* at $\mathcal{DO}$. We say a PKE is IND-CCA-secure, if for polynomially bounded q and T, ϵ is negligible.

2.2 Tag Based Encryption

Informally, a tag based encryption (TBE) is a public key encryption scheme where the encryption and the decryption operations take an additional "tag" which is public binary string with proper length.

A TBE scheme consists of three algorithms $\mathcal{TBE} = (\mathsf{TBEkg}, \mathsf{TBEenc}, \mathsf{TBEdec})$. The randomized key generation algorithm TBEkg, taking a security parameter k as the input, outputs a public key pk and a corresponding secret key sk, denoted as $(pk, sk) \leftarrow \mathsf{TBEkg}(1^k)$. The randomized encryption algorithm TBEenc taking a public key pk, a tag t and a plaintext m taken from the message space as inputs, with internal coin flipping r, outputs a ciphertext c, which is denoted as $c \leftarrow \mathsf{TBEenc}(pk, t, m, r)$, in brief $c \leftarrow \mathsf{TBEenc}(pk, t, m)$. The deterministic algorithm TBEdec taking a secret key sk, a tag t and a ciphertext c as inputs, outputs a plaintext m, or "$\perp$" indicating invalid ciphertext, which is denoted $m \leftarrow \mathsf{TBEdec}(sk, t, m)$. We require for all $(pk, sk) \leftarrow \mathsf{TBEkg}(1^k)$, all m and all t, $\mathsf{TBEdec}(sk, t, \mathsf{TBEenc}(pk, t, m)) = m$.

Separability. A TBE is said to be sparable if the encryption algorithm can be arranged in two parts, such that one part is uniquely determined by pk, m and the random coin r, in brief $u \leftarrow f_1(pk, m, r)$, and the other part is uniquely determined by the pk, t and r, in brief $v \leftarrow f_2(pk, t, r)$. The ciphertext is $c = \langle u, v \rangle$.

IND-sTag-CCA-Security. We consider a weak security called indistinguishability against selective-tag and chosen ciphertext attack IND-sTag-CCA. Namely, the tag to be used in the challenge is chosen before the key generation phase. Again, the scheme is (ϵ, q, T)-IND-sTag-CCA-secure if any adversary $\mathcal{A}$ with at most q queries to a decryption oracle $\mathcal{DO}$, has advantage at most ϵ within time T in the following experiment.

$$\begin{aligned}\mathrm{Adv}^{\textsf{ind-stag-cca}}_{\mathcal{TBE},\mathcal{A}} \stackrel{\mathsf{def}}{=} \Pr[&(t^*, s_0) \leftarrow \mathcal{A}(1^k); (pk, sk) \leftarrow \mathsf{TBEkg}(1^k);\\ &(m_0, m_1, s_1) \leftarrow \mathcal{A}^{\mathcal{DO}}(pk, s_0);\\ b \leftarrow \{0,1\}; c^* \leftarrow \mathsf{TBEenc}&(pk, t^*, m_b); b' \leftarrow \mathcal{A}^{\mathcal{DO}}(c^*, s_1) : b' = b] - 1/2\end{aligned}$$

where $\mathcal{DO}$ returns the corresponding decryption result on a query of a ciphertext c under tag t, whereas $\mathcal{A}$ is forbidden to query any ciphertext under tag t^* at $\mathcal{DO}$. We say a TBE is IND-sTag-CCA-secure, if for polynomially bounded q and T, ϵ is negligible.

2.3 Identity Based Encryption

An identity based encryption (IBE) can be regarded as a special tag based encryption equipped with an additional extraction algorithm, with inputs a master secret key and an tag, outputs a secret key that is capable to decrypt ciphertext corresponding to this tag.

An IBE scheme consists of four algorithms $\mathcal{IBE} = (\mathsf{IBEkg}, \mathsf{IBEext}, \mathsf{IBEenc}, \mathsf{IBEdec})$. The randomized key generation algorithm IBEkg, taking a security parameter k as the input, outputs a public parameter $params$ and a master secret key msk, denoted as $(params, msk) \leftarrow \mathsf{TBEkg}(1^k)$. The extract algorithm, possibly randomized, takes inputs of $params$, msk and an identity id, outputs a secret key sk_{id} for id, denoted as $sk_{id} \leftarrow \mathsf{IBEext}(params, msk, id)$, in brief $sk_{id} \leftarrow \mathsf{IBEext}(msk, id)$. The randomized encryption algorithm TBEenc takes $params$, an identity id and a plaintext m taken from the message space as inputs, with internal coin flipping r, outputs a ciphertext c, which is denoted as $c \leftarrow \mathsf{IBEenc}(params, id, m, r)$, in brief $c \leftarrow \mathsf{IBEenc}(params, id, m)$. The deterministic algorithm IBEdec takes a secret key sk_{id}, an identity id and a ciphertext c as inputs, outputs a plaintext m, or a special symbol "$\perp$", which is denoted $m \leftarrow \mathsf{IBEdec}(sk_{id}, id, c)$. We require for all $(params, msk) \leftarrow \mathsf{IBEkg}(1^k)$, $sk_{id} \leftarrow \mathsf{IBEext}(msk, id)$ and all m, we have $\mathsf{IBEdec}(sk_{id}, id, \mathsf{IBEenc}(params, id, m)) = m$.

Separability. An IBE is said to be sparable if the encryption algorithm can be arranged in two parts, such that one part is uniquely determined by $params$, m and the random coin r, in brief $u \leftarrow f_1(params, m, r)$, and the other part is uniquely determined by the $params$, id and r, in brief $v \leftarrow f_2(params, id, r)$. The ciphertext is $c = \langle u, v \rangle$.

IND-sID-CPA-Security. We consider security of indistinguishability against selective-ID and chosen plaintext attack (IND-sID-CPA). We say an identity based encryption is (ϵ, q, T)-IND-sID-CPA-secure if the advantage of any adversary $\mathcal{A}$

is at most ϵ, with access q times to an extraction oracle $\mathcal{EO}$ within time T in the following experiment.

$$\begin{aligned}\mathrm{Adv}^{\textsf{ind-sid-cpa}}_{\mathcal{IBE},\mathcal{A}} \overset{\mathsf{def}}{=} \Pr[&(id^*, s_0) \leftarrow \mathcal{A}(1^k); (params, msk) \leftarrow \mathsf{IBEkg}(1^k);\\ &(m_0, m_1, s_1) \leftarrow \mathcal{A}^{\mathcal{EO}}(params, s_0); b \leftarrow \{0,1\};\\ &c^* \leftarrow \mathsf{IBEenc}(params, id^*, m_b); b' \leftarrow \mathcal{A}^{\mathcal{EO}}(c^*, s_1) : b' = b] - 1/2\end{aligned}$$

where $\mathcal{EO}$ returns the corresponding secret key on a query on identity id, whereas $\mathcal{A}$ is forbidden to query id^* at $\mathcal{EO}$. We say an IBE is IND-sID-CPA-Secure, if for polynomially bounded q and T, ϵ is negligible.

2.4 Digital Signature

A signature scheme consists of three algorithms $\mathcal{S} = (\mathsf{G}, \mathsf{S}, \mathsf{V})$. The randomized key generation algorithm G takes a security parameter k, and generates signing key $sigk$ and verification key vk. The possibly randomized signing algorithm S takes as inputs $sigk$ and $m \in \{0,1\}^*$, where m is a message, and outputs a signature σ. The deterministic verification algorithm V takes as inputs vk, m and σ, and outputs a symbol $\beta \in \{\texttt{accept}, \texttt{reject}\}$, denoted as $\beta \leftarrow \mathsf{V}(vk, m, \sigma)$. We require that for all $(sigk, vk) \leftarrow \mathsf{G}(1^k)$, all $m \in \{0,1\}^*$, $\mathsf{V}(vk, m, \mathsf{S}(sigk, m)) = \texttt{accept}$.

(Strong) Unforgeability. We consider strong unforgeability against adaptive chosen message attack sUF-CMA [3]. Let $\mathcal{S} = (\mathsf{G}, \mathsf{S}, \mathsf{V})$ be a signature scheme. Let $\mathcal{A}$ and k be an adversary and a security parameter, respectively.

Denote $\{(\sigma_i, m_i)\}_{q_s}$ as the set contains all q_s pairs of queries and replies between $\mathcal{A}$ and $\mathcal{SO}$, where $\mathcal{SO}$ is a signing oracle which for a given message m, returns a corresponding signature σ. The success probability of $\mathcal{A}$ is defined as

$$\begin{aligned}\mathrm{Suc}^{\textsf{suf-cma}}_{\mathcal{S},\mathcal{A}}(k) \overset{\mathsf{def}}{=} \Pr[&(vk, sigk) \leftarrow \mathsf{Gen}(1^k); (\sigma^*, m^*) \leftarrow \mathcal{A}^{\mathcal{SO}}(vk)\\ &: \mathsf{V}(vk, m^*, \sigma^*) = \texttt{accept} \wedge (\sigma^*, m^*) \notin \{(\sigma_i, m_i)\}_{q_s}]\end{aligned}$$

We say $\mathcal{S}$ is (t, ϵ)-sUF-CMA secure if for any $\mathcal{A}$ in time bound t, $\mathcal{A}$'s success probability is at most ϵ. Especially, we say that $\mathcal{S}$ is sUF-CMA secure if ϵ is negligible.

2.5 Collision Resistant Chameleon Hash Function

A collision resistant chameleon hash function consists of three algorithms $\mathcal{CMH} = (\mathsf{CMkg}, \mathsf{CMhash}, \mathsf{CMswch})$. The randomized key generation algorithm CMkg taking a security parameter k as the input, outputs a hash key hk and a trapdoor td, denoted as $(hk, td) \leftarrow \mathsf{CMkg}(1^k)$. The randomized hashing algorithm takes inputs a public key hk, an auxiliary random coin w drawn from space $\mathcal{R}$ and a value $x \in \{0,1\}^*$, outputs a binary string y of fixed length l, denoted as $y \leftarrow \mathsf{CMhash}(hk, x, w)$. The switch algorithm CMswch takes inputs the trapdoor

td, a pair of messages x, x', the corresponding auxiliary random coin w, outputs a pair of (x', w') with $x' \neq x$, such that $\mathsf{CMhash}(hk, x, w) = \mathsf{CMhash}(hk, x', w')$, denoted as $r' \leftarrow \mathsf{CMswch}(td, x, w, x')$. Finally, for all $x, x' \leftarrow \{0,1\}^*$ and $w \in \mathcal{R}$, we require $w' \leftarrow \mathsf{CMswch}(td, x, w, x')$ is uniformly distributed in $\mathcal{R}$ and we call this property the uniformness of a chameleon hash function. We next give two flavors of security requirements for a chameleon hash, namely collision resistance (CR) and oracle collision resistance (OCR).

Collision Resistance. We say a chameleon hash function is $(\epsilon_{\mathcal{H}}, T_{\mathcal{H}})$-collision resistant (CR) if any adversary $\mathcal{A}$ without access to the trapdoor td, the success probability of finding collisions is at most $\epsilon_{\mathcal{H}}$ within time T in the following experiment.

$$\begin{aligned}\mathrm{Succ}^{\mathsf{cr}}_{\mathcal{CMH},\mathcal{A}} \stackrel{\mathsf{def}}{=} \Pr[&(hk, td) \leftarrow \mathsf{CMkg}(1^k); x \leftarrow \mathcal{A}(hk);\\ &w \leftarrow \mathcal{R}; y \leftarrow \mathsf{CMhash}(hk, x, w); (x', w') \leftarrow \mathcal{A}(hk, x, w)\\ &: (x', w') \neq (x, w) \wedge y = \mathsf{CMhash}(hk, x', w')]\end{aligned}$$

We say a chameleon hash function is collision resistant, if for polynomially bounded $T_{\mathcal{H}}$, $\epsilon_{\mathcal{H}}$ is negligible.

Oracle Collision Resistance. We say a chameleon hash function is $(\epsilon_{\mathcal{H}}, T_{\mathcal{H}})$-oracle collision resistant (OCR) if any adversary $\mathcal{A}$ without access to the trapdoor td, the success probability of finding a pair of collisions is at most $\epsilon_{\mathcal{H}}$ within time $T_{\mathcal{H}}$ in the following experiment.

$$\begin{aligned}\mathrm{Succ}^{\mathsf{ocr}}_{\mathcal{CMH},\mathcal{A}} \stackrel{\mathsf{def}}{=} \Pr[&(hk, td) \leftarrow \mathsf{CMkg}(1^k); \bar{x} \leftarrow \{0,1\}^*; \bar{w} \leftarrow \mathcal{R};\\ &y \leftarrow \mathsf{CMhash}(hk, \bar{x}, \bar{w}); x \leftarrow \mathcal{A}(hk, y); w \leftarrow \mathsf{CMswch}(td, \bar{x}, \bar{w}, x);\\ &(x', w') \leftarrow \mathcal{A}(hk, x, w) : (x', w') \neq (x, w) \wedge y = \mathsf{CMhash}(hk, x', w')]\end{aligned}$$

We say a chameleon hash function is oracle collision resistant, if for polynomially bounded $T_{\mathcal{H}}$, $\epsilon_{\mathcal{H}}$ is negligible.

Discussions. Oracle collision resistance has not been formally discussed before. However, it seems a very natural definition for chameleon hash functions, since it considers a collision cannot be found even after the adversary gets some hint from a switch oracle. Simultaneously, compared with the game defining collision resistance, the adversary seems to have quite limit power in choosing its target, since it is required to find collisions on a specified random hash value. At present, we don't know whether there are implications or separations between the above two notions. However, there are practical implementations of such chameleon hash functions provably secure under proper assumptions. We provide such an example here: A chameleon hash by combining Pedersen commitment with a normal collision resistant hash function appeared in [23], and it has been proven to be collision resistant under the discrete log assumption [23]. But one can easily come with a proof that it is also oracle collusion resistant under one-more discrete log assumption [26], which is equivalent to discrete log assumption in the generic group model [29]. We omit the details here.

Chameleon Hash as One-Time Signature. A strongly existentially unforgeable [3] one-time signature can be derived from an oracle collision resistant chameleon hash function. For key generation, run $(hk, td) \leftarrow \mathsf{CMkg}(1^k)$, select random $(\bar{x}, \bar{w})$ from corresponding spaces and compute $y = \mathsf{CMhash}(hk, \bar{x}, \bar{w})$. The verification key is $vk = (hk, y)$ and the signing key is $sigk = (td, \bar{x}, \bar{w})$. For signing, on message m, compute $s \leftarrow \mathsf{CMswch}(td, \bar{x}, \bar{w}, m)$, the signature on m is s. The correctness of the construction is easily verified.

To see strong unforgeability, a forger, given vk, will try to output (m', s'), such that $(m', s') \neq (m, s)$, where s is the signature on m chosen by the forger, and (m', s') a valid message/signature pair under vk. Then if $y = \mathsf{CMhash}(hk, m', s')$, a collision occurs for (m, s), which is against the assumption of oracle collision resistance. We then conclude the above signature scheme is strongly existentially unforgeable against (exactly one-time) adaptive chosen message attack.

2.6 Hard Problems

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p and g be a generator of $\mathbb{G}_1$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ satisfies the following properties: (i) *Bilinearity*: For all $x, y \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, $e(x^a, y^b) = e(x, y)^{ab}$. (ii) *Non-degeneracy*: $e(g, g) \neq 1$. (iii) *Computability*: There is an efficient algorithm to compute $e(x, y)$ for any $x, y \in \mathbb{G}_1$. We review some hard problems related to bilinear maps: the decision bilinear Diffie-Hellman (DBDH) problem, the decision bilinear Diffie-Hellman inversion (DBDHI) problem, the decision linear (DLIN) problem and the decision augmented bilinear Diffie-Hellman exponent (DABDHE) problem. We say an assumption holds, if the advantage ϵ of any probabilistic polynomial bounded algorithm is negligible for the corresponding problem.

DBDH Problem. We say that the DBDH problem is (ϵ, T)-hard in $(\mathbb{G}_1, \mathbb{G}_2)$, if given 5-tuple $(g, g^a, g^b, g^c, w) \in (\mathbb{G}_1)^4 \times \mathbb{G}_2$ as input, any randomized algorithm $\mathcal{A}$ with running time at most T, distinguishes the BDH-tuple from a random tuple with advantage at most ϵ.

$$\mathrm{Adv}^{\mathsf{dbdh}}_{\mathbb{G}_1,\mathbb{G}_2,\mathcal{A}} \overset{\mathsf{def}}{=} \left|\Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, w) = 0]\right|$$

DBDHI Problem. We say that the DBDHI problem is (ϵ, q, T)-hard in $(\mathbb{G}_1, \mathbb{G}_2)$, if given $(q+2)$-tuple $(g, g^x, g^{x^2}, ..., g^{x^q}, w) \in (\mathbb{G}_1)^{q+1} \times \mathbb{G}_2$, where $x \in \mathbb{Z}_p$ as input, any randomized algorithm $\mathcal{A}$ with running time at most T, decides whether $w = e(g, g)^{1/x}$ with advantage at most ϵ.

$$\mathrm{Adv}^{\mathsf{dbdh}}_{\mathbb{G}_1,\mathcal{A}} \overset{\mathsf{def}}{=} |\Pr[\mathcal{A}(g, g^x, g^{x^2}, ..., g^{x^q}, e(g,g)^{1/x}) = 0] - \Pr[\mathcal{A}(g, g^x, g^{x^2}, ..., g^{x^q}, w) = 0]|$$

DLIN Problem. We say that the DLIN problem is (ϵ, T)-hard in $\mathbb{G}_1$, if given 6-tuple $(g_1, g_2, g_1^{r_1}, g_2^{r_2}, z, w) \in (\mathbb{G}_1)^6$ as input, where $(r_1, r_2) \in (\mathbb{Z}_p)^2$, any randomized algorithm $\mathcal{A}$ with running time at most T, decides whether $w = z^{r_1+r_2}$ with advantage at most ϵ.

$$\mathrm{Adv}^{\mathsf{dbdh}}_{\mathbb{G}_1,\mathcal{A}} \overset{\mathsf{def}}{=} \left|\Pr[\mathcal{A}(g_1, g_2, g_1^{r_1}, g_2^{r_2}, z, z^{r_1+r_2}) = 0] - \Pr[\mathcal{A}(g_1, g_2, g_1^{r_1}, g_2^{r_2}, z, w) = 0]\right|$$

It is believed that DLIN problem is hard even in a bilinear group pair where pairing is efficiently computable and its security can be proven in the generic group model.

DABDHE Problem. We say that the DABDHE problem is (ϵ, q, T)-hard in $(\mathbb{G}_1, \mathbb{G}_2)$, if given $(q+4)$-tuple $(g_1, g_2, g_1^x, ..., g_1^{x^q}, g_1^{x^{q+2}}, w) \in (\mathbb{G}_1)^{q+3} \times \mathbb{G}_2$, where $x \in \mathbb{Z}_p$ as input, any randomized algorithm $\mathcal{A}$ with running time at most T, decides whether $w = e(g_1, g_2)^{q+1}$ with advantage at most ϵ.

$$\begin{aligned}\mathrm{Adv}^{\mathsf{dabdhe}}_{\mathbb{G}_1,\mathbb{G}_2,\mathcal{A}} \overset{\mathsf{def}}{=} |\Pr[\mathcal{A}(g_1, g_2, g_1^x, ..., g_1^{x^q}, g_1^{x^{q+2}}, e(g_1, g_2)^{x^{q+1}}) = 0] \\ - \Pr[\mathcal{A}(g_1, g_2, g_1^x, ..., g_1^{x^q}, g_1^{x^{q+2}}, w) = 0]|\end{aligned}$$

3 Separable TBE/IBE to PKE Transforms

3.1 The Transforms

We give Transform T1 that achieves chosen ciphertext security from separable tag based primitives in Figure 1, then analyze its security in Theorem 1.

Theorem 1. *The public key encryption acquired via* T1 *is* $(\epsilon + \epsilon_{\mathcal{H}}, q, T + T_{\mathcal{H}} + O(qk))$-$\mathsf{IND\text{-}CCA}$-*secure assuming the separable tag based encryption is* (ϵ, q, T)-$\mathsf{IND\text{-}sTag\text{-}CCA}$-*secure and the collision resistant chameleon hash function is* $(\epsilon_{\mathcal{H}}, T_{\mathcal{H}})$-*collision resistant.*

Proof. We show how to build an adversary $\mathcal{B}$ breaks either the TBE or the chameleon hash by interacting with a PKE adversary. Denote $\langle u^*, v^*, r_2^* \rangle$ as the challenge ciphertext for $\mathcal{A}$. We distinguish two types of adversaries:

Type 1: For any valid decryption query $\langle u^{(i)}, v^{(i)}, r_2^{(i)} \rangle$, where $1 \le i \le q$, there is $(u^{(i)}, r_2^{(i)}, t^{(i)}) \neq (u^*, r_2^*, t^*)$, where $t^{(i)} = \mathsf{CMhash}(hk, u^{(i)}, r_2^{(i)})$ and $t^* = \mathsf{CMhash}(hk, u^*, r_2^*)$. We construct an adversary that breaks the TBE.

Type 2: There is at least one valid decryption query $\langle u^{(i)}, r_2^{(i)}, t^{(i)} \rangle = \langle u^*, r_2^*, t^* \rangle$ for some $1 \le i \le q$. We construct an adversary that breaks the collision resistance of chameleon hash.

Type 1 Adversary: Define $\mathcal{B}$ as follows:

Setup: $\mathcal{B}$ runs $(hk, td) \leftarrow \mathsf{CMkg}(1^k)$. Let $u' = f_1(pk, m', r_1')$, where m' is a dummy message and r_1' is random coin for TBEenc. $\mathcal{B}$ computes $t^* = \mathsf{CMhash}(hk, u', r_2')$, where r_2' is an auxiliary random coin for chameleon hash. $\mathcal{B}$ then submits t^* to its own challenger as the tag to be challenged. After receiving public key pk' from its challenger, $\mathcal{B}$ sets $pk = (pk', hk)$ and sends pk as the public key to a PKE adversary $\mathcal{A}$.

Transform: T1		
$\mathsf{PKEkg}(1^k)$:	$\mathsf{PKEenc}(pk, m)$:	$\mathsf{PKEdec}(sk, c)$:
$(pk', sk') \leftarrow \mathsf{TBEkg}(1^k)$	$pk = \langle pk', hk \rangle$	$c = \langle u, v, r_2 \rangle$
$hk \leftarrow \mathsf{CMkg}(1^k)$	$r_1 \leftarrow \mathcal{R}_1$	$t \leftarrow \mathsf{CMhash}(hk, u, r_2)$
$pk \leftarrow \langle pk', hk \rangle$	$r_2 \leftarrow \mathcal{R}_2$	$m \leftarrow \mathsf{TBEdec}(sk, t, u\|\|v)$
$sk \leftarrow sk'$	$u \leftarrow f_1(pk', m, r_1)$	return m
return (pk, sk)	$t \leftarrow \mathsf{CMhash}(hk, u, r_2)$	
	$v \leftarrow f_2(pk', t, r_1)$	
	$c \leftarrow \langle u, v, r_2 \rangle$	
	return c	

[†] The trapdoor for chameleon hash function can be erased since it is not used elsewhere. $\mathcal{R}_1$ and $\mathcal{R}_2$ are corresponding spaces of random coins for TBEenc and CMhash.

Fig. 1. The Separable TBE to PKE Transform

Encryption Query: When $\mathcal{B}$ receives from $\mathcal{A}$ a pair of plaintexts (m_0, m_1) that $\mathcal{A}$ wants to be challenged on, $\mathcal{B}$ forwards (m_0, m_1) to its own challenge oracle. After receiving its challenge ciphertext $\langle u^*, v^* \rangle$ (under tag t^*), $\mathcal{B}$ computes $r_2^* \leftarrow \mathsf{CMswch}(td, u', r_2', u^*)$. $\mathcal{B}$ then sends $c^* = \langle u^*, v^*, r_2^* \rangle$ to $\mathcal{A}$ as the challenge ciphertext. Due to the uniformness of the chameleon hash, the distribution of the challenge is exactly as a real attack.

Decryption Queries: For decryption query $c = \langle u^{(i)}, v^{(i)}, r_2^{(i)} \rangle$, $\mathcal{B}$ checks whether $(u^{(i)}, r_2^{(i)}) = (u^*, r_2^*)$ and $v^{(i)} \neq v^*$. If yes, this is an invalid ciphertext and $\mathcal{B}$ rejects. Otherwise, $\mathcal{B}$ sends $\langle u^{(i)}, v^{(i)}, t^{(i)} \rangle$ to its own decryption oracle and forwards to $\mathcal{A}$ whatever its decryption oracle replies.

Guess: When $\mathcal{A}$ outputs a guess b' on c^*, $\mathcal{B}$ outputs the same bit as its answer.

From the description of $\mathcal{B}$, it is easily verified that the key generation is simulated perfectly. Furthermore, because of the uniformness property, r_2^* is uniformly distributed, thus the challenge oracle is also perfectly simulated. Finally, if $\mathcal{A}$ succeeds, $\mathcal{B}$ also succeeds.

Type 2 adversary: For Type 2 adversary has many similarities with Type 1, so we only give the sketch. For setup, $\mathcal{B}$ receives hk from its challenger. $\mathcal{B}$ then generates $(pk', sk') \leftarrow \mathsf{TBEkg}(1^k)$ and sets the public key as (pk', hk). $\mathcal{B}$ keeps sk' as the secret key. Since $\mathcal{B}$ has sk', all decryption queries can be answered perfectly. For challenge, upon receiving (m_0, m_1) from $\mathcal{A}$, $\mathcal{B}$ first picks $b \leftarrow \{0, 1\}$ and sets $u^* = f_1(pk', m_b, r)$, where r is chosen uniformly from corresponding randomness space. $\mathcal{B}$ then outputs u^* to its hash challenger. After receiving r_2^* from the challenger, $\mathcal{B}$ sets $t^* \leftarrow \mathsf{CMhash}(hk, u^*, r_2^*)$ and $v^* = f_2(pk', t^*, r)$, and sends $\langle u^*, v^*, r_2^* \rangle$ to $\mathcal{A}$ as the challenge ciphertext. One can verify this is a valid challenge. Finally, when decryption query $\langle u^{(i)}, v^{(i)}, r_2^{(i)} \rangle$ is queried for some i, where $\mathsf{CMhash}(hk, u^{(i)}, r_2^{(i)}) = t^*$ and $(u^{(i)}, r_2^{(i)}) \neq (u^*, r_2^*)$, $\mathcal{B}$ outputs $(u^{(i)}, r_2^{(i)})$ as a collision for its challenger. We conclude $\mathcal{B}$ break collision resistance with the same probability as $\mathcal{A}$'s advantage in guessing b.

Summarizing two cases, we have the claimed results. $\square$

We further present another transform (T1′) based on IBE in Figure 2. Since an (ϵ, q, T)-IND-sID-CPA-Secure identity based encryption implies an (ϵ, q, T)-IND-sTag-CCA-secure tag based encryption, we have an immediate corollary for the security of this transform.

Transform: T1′		
$\mathsf{PKEkg}(1^k)$: $(params, msk) \leftarrow \mathsf{IBEkg}(1^k)$ $hk \leftarrow \mathsf{CMkg}(1^k)$ $pk \leftarrow \langle params, hk \rangle$ $sk \leftarrow msk$ return (pk, sk)	$\mathsf{PKEenc}(pk, m)$: $pk = \langle params, hk \rangle$ $r_1 \leftarrow \mathcal{R}_1$ $r_2 \leftarrow \mathcal{R}_2$ $u \leftarrow f_1(params, m, r_1)$ $t \leftarrow \mathsf{CMhash}(hk, u, r_2)$ $v \leftarrow f_2(params, id, r_1)$ $c \leftarrow \langle u, v, r_2 \rangle$ return c	$\mathsf{PKEdec}(sk, c)$: $c = \langle u, v, r_2 \rangle$ $id \leftarrow \mathsf{CMhash}(hk, u, r_2)$ $sk_{id} \leftarrow \mathsf{IBEext}(sk, id)$ $m \leftarrow \mathsf{IBEdec}(sk_{id}, id, u\|\|v)$ return m

† Again, td can be erased. $\mathcal{R}_1$ and $\mathcal{R}_2$ are corresponding spaces of random coins for IBEenc and CMhash.

Fig. 2. The Separable IBE to PKE Transform

Corollary 1. *The public key encryption acquired via* T1′ *is* $(\epsilon + \epsilon_{\mathcal{H}}, q, T + T_{\mathcal{H}} + O(qk))$-IND-CCA-*secure assuming the separable identity based encryption is* (ϵ, q, T)-IND-sID-CPA-*Secure and the collision resistant chameleon hash function is* $(\epsilon_{\mathcal{H}}, T_{\mathcal{H}})$-*collision resistant.*

3.2 Further Improvements

Note that the performance of T1 and T1′ can be further optimized by replacing chameleon hash functions using practical hash functions (e.g. SHA-1, or SHA-2 for higher security). We have to modify our assumptions and corresponding proofs. Instead of possessing td of a chameleon hash function, the simulator $\mathcal{B}$ is given one chance of accessing a collision oracle (possibly inefficient) that finds collision on the hash function. The uniformness of such hash functions should also be rephrased accordingly. With these changes, it is still possible to prove the security of the modified constructions according to previous strategies. The details are omitted here.

4 A Generic TBE to PKE Transform

We give the description of a generic TBE to PKE transform using chameleon hash in Figure 3, and naturally it can be rewritten to fit IBE case accordingly. The transform mimics [13] with a one-time signature replaced by a chameleon hash function that is discussed in Section 2.5. This transform shows another tradeoff between computational cost and ciphertext size for generic TBE/IBE

```
Transform: T2

PKEkg($1^k$):
    $(pk, sk) \leftarrow$ TBEkg($1^k$)
    return $(pk, sk)$

PKEenc($pk, m$):
    $(hk, sk) \leftarrow$ CMkg($1^k$)
    $\bar{u} \leftarrow \mathcal{C}$
    $\bar{r} \leftarrow \mathcal{R}$
    $t' \leftarrow$ CMhash($hk, \bar{u}, \bar{r}$)
    $t \leftarrow hk||t'$
    $u \leftarrow$ TBEenc($pk, t, m$)
    $r \leftarrow$ CMswch($td, \bar{u}, \bar{r}, u$)
    $c \leftarrow \langle u, t, r \rangle$
    return $c$

PKEdec($sk, c$):
    $c = \langle u, hk||t', r \rangle$
    test if $t' \overset{?}{=}$ CMhash($hk, u, r$)
    if invalid, return "$\perp$"
    $m \leftarrow$ TBEdec($sk, t, u$)
    return $m$
```

† $\mathcal{C}$ and $\mathcal{R}$ are corresponding spaces of ciphertext and random coins of CMhash respectively.

Fig. 3. A Generic TBE to PKE Transform Using Chameleon Hash

transforms. It is not hard to come up with an IBE version of T2, just as we did previously to T1. We omit the details here. Finally, Theorem 2 guarantees the security of T2.

Theorem 2. *The public key encryption acquired via* T2 *is* $(\epsilon + \epsilon_{\mathcal{H}}, q, T + T_{\mathcal{H}} + O(qk))$-IND-CCA*-secure assuming the tag based encryption is* (ϵ, q, T)-IND-sTag-CCA*-secure and the chameleon hash function is* $(\epsilon_{\mathcal{H}}, T_{\mathcal{H}})$*-oracle collision resistant.*

Proof Sketch. The idea of the proof is quite similar to [13]. Let $\mathcal{A}$ be a PKE adversary. Denote $\langle u^{(i)}, t^{(i)}, r^{(i)} \rangle$, where $1 \leq i \leq q$, as decryption queries by $\mathcal{A}$. Denote $\langle u^*, t^*, r^* \rangle$ be the challenge ciphertext for $\mathcal{A}$. We consider two types of adversaries.

Type 1: For any $\langle u^{(i)}, t^{(i)}, r^{(i)} \rangle$, where $t^{(i)} \neq t^*$, we build an adversary $\mathcal{B}$ against the underlying TBE.

Type 2: There exists a query $\langle u^{(i)}, t^*, r^{(i)} \rangle$, where $(u^{(i)}, r^{(i)}) \neq (u^*, r^*)$, $t^* = hk^*||t'^*$ and $t'^* =$ CMhash$(hk^*, u^{(i)}, r^{(i)})$. We construct an adversary $\mathcal{B}$ against the oracle collision resistance of the underlying chameleon hash.

Type 1 Adversary. For setup, $\mathcal{B}$ runs $(hk, td) \leftarrow$ CMkg(1^k), chooses dummy $\bar{u}$ and random coin $\bar{r}$ and sets $t' \leftarrow$ CMhash$(hk, \bar{u}, \bar{r})$. Then $\mathcal{B}$ outputs $t^* = (hk||t')$ as the selective tag for a TBE challenger. After receiving pk from the TBE challenger, $\mathcal{B}$ forwards pk to $\mathcal{A}$. For decryption, since there is no query with tag $t^{(i)} = t^*$, all decryption queries of $\mathcal{A}$ can be forwarded to the TBE challenger and answered perfectly. For challenge, after $\mathcal{A}$ submits a pair of chosen message (m_0, m_1), $\mathcal{B}$ forwards to the TBE challenger and receives a challenge ciphertext u^*. $\mathcal{B}$ then computes $r^* \leftarrow$ CMswch$(td, \bar{u}, \bar{r}, u^*)$, and gives $\langle u^*, t^*, r^* \rangle$ to $\mathcal{A}$ as the challenge. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs as its guess. From above descriptions we can see that $\mathcal{B}$ succeeds with exactly the probability of $\mathcal{A}$.

Type 2 Adversary. For setup, $\mathcal{B}$ runs $(pk, sk) \leftarrow \mathsf{TBEkg}(1^k)$. $\mathcal{B}$ then forwards pk to $\mathcal{A}$ as the public key of the PKE. Additionally, on $\mathcal{B}$'s request, a chameleon hash challenger runs $(hk^*, td^*) \leftarrow \mathsf{CMkg}(1^k)$ and computes $t'^* \leftarrow \mathsf{CMhash}(hk^*, \bar{u}, \bar{r})$, where $\bar{u}$ and $\bar{r}$ are uniformly sampled from corresponding spaces. Then $t^* = hk||t'^*$ is given to $\mathcal{B}$. For decryption, since $\mathcal{B}$ knows sk, all decryption queries will be handled perfectly. For challenge, $\mathcal{B}$ receives (m_0, m_1) from $\mathcal{A}$ and sets $u^* \leftarrow \mathsf{TBEenc}(pk, t^*, m_b)$ for $b \leftarrow \{0,1\}$. $\mathcal{B}$ submits u^* to the chameleon hash challenger, who computes $r^* \leftarrow \mathsf{CMswch}(td^*, \bar{u}, \bar{r}, u^*)$ and returns r^* to $\mathcal{B}$. $\mathcal{B}$ then gives $\langle u^*, t^*, r^* \rangle$ to $\mathcal{A}$ as a challenge. After receiving a ciphertext query of the form $\langle u^{(i)}, t^*, r^{(i)} \rangle$ where $t'^* = \mathsf{CMhash}(hk^*, u^{(i)}, r^{(i)})$ and $(u^{(i)}, r^{(i)}) \neq (u^*, r^*)$, $\mathcal{B}$ outputs $(u^{(i)}, r^{(i)})$ as a collusion for the chameleon hash. It is verified that $\mathcal{B}$ breaks the oracle collision resistance of chameleon hash at exactly $\mathcal{A}$'s advantage in correctly guess b.

Summarizing the above two cases we prove the claim. $\square$

Although additional computational cost may be involved in the key generation and evaluation of the chameleon hash, it can be improved by pre-computation. It is worth repeating that oracle collision chameleon hash functions can be built from many number theoretic assumptions, and the public verifiability of underlying primitives is preserved by using T2.

5 Applications

5.1 Practical CCA-Secure PKE Schemes

Our methods achieves CCA-security with tight reductions to underlying selective-Tag based primitives. Instantiate T1 with IND-sTag-CCA-secure TBE, one gets more efficient scheme with public verifiability. We give such a scheme based on Kiltz TBE [22]. To remark, the Cramer-Shoup encryption [15] can be viewed as applying the BMW transform [11] to a related TBE with adaptive chosen tag security. Finally, instantiate T1′ with Gentry IBE, one gets an efficient PKE based on DABDHE assumption.

The Scheme

```
PKEkg($1^k$):
    $g_1, h \leftarrow \mathbb{G}_1$
    $x_1, x_2, y_1, y_2 \leftarrow \mathbb{Z}_p^*$
    $g_2, z \leftarrow \mathbb{G}_2$
        s.t. $g_1^{x_1} = g_2^{x_2} = z$
    $u_1 \leftarrow g_1^{y_1}$
    $u_2 \leftarrow g_2^{y_2}$
    choose $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$
    $pk \leftarrow (g_1, g_2, u_1, u_2, z, h, H)$
    $sk \leftarrow (x_1, x_2, y_1, y_2)$
    return $(pk, sk)$
```

```
PKEenc($pk, m$):
    $r_1, r_2, r_3 \leftarrow \mathbb{Z}_p^*$
    $u_1 \leftarrow g_1^{r_1}$
    $u_2 \leftarrow g_2^{r_2}$
    $e \leftarrow M \cdot z^{r_1 + r_2}$
    $t \leftarrow H_2(g_1^{H_1(u_1, u_2, e)} h^{r_3})$
    $v_1 \leftarrow z^{tr_1} u_1^{r_1}$
    $v_2 \leftarrow z^{tr_2} u_2^{r_2}$
    $c \leftarrow \langle u_1, u_2, e, v_1, v_2, r_3 \rangle$
    return $c$
```

```
PKEdec($sk, c$):
    $c = \langle u_1, u_2, e, v_1, v_2, r_3 \rangle$
    $t \leftarrow H(g_1^{H(u_1, u_2, e)} h^{r_3})$
    test whether
        $v_1 = u_1^{tx_1 + y_1} \wedge v_2 = u_2^{tx_2 + y_2}$
    if invalid, return "$\perp$"
    otherwise    $m \leftarrow \frac{e}{u_1^{x_1} u_2^{x_2}}$
    return $m$
```

† $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair with prime order p. H_1, H_2 can be replaced by injective mappings.

Fig. 4. Secure PKE Based on DLIN Assumption

A Publicly Verifiable CCA-Secure PKE without Pairings. We instantiate our method with Kiltz TBE [22], and present an efficient IND-CCA-secure public key encryption scheme in Figure 4. We note that while Kiltz KEM [22] combined with a CCA-secure symmetric key encryption achieves better performance regarding chosen ciphertext security, but our scheme enjoys additionally a capability of threshold decryptions. Here we instantiate the chameleon hash by using a collision resistant hash function combined with the Pedersen commitment [23].

5.2 Extensions

CCA-Secure Hierarchical Identity Based Encryption. Since most IBEs in the standard model can be extended to the hierarchical IBE setting, our methods operate also on these HIBE schemes. Applying our transforms to an $(\ell+1)$-level semantically secure HIBE against chosen plaintext attack results in an ℓ-level CCA-secure HIBE with marginal computational cost, small ciphertext overhead and tight security reduction.

Table 1. Comparisons of Schemes

Scheme	Assumption	Ciphertext Overhead	Without Pairing?	Generic?	Public Verifiable?
KD	DDH	$2\|\mathbb{G}\| + \|\mathsf{Mac}\|$	yes	—	—
CHK/BB1	DBDH	$2\|\mathbb{G}_1\| + O(k^2)$	—	yes	yes
CHK/BB2	DBDHI	$2\|\mathbb{G}_1\| + O(k^2)$	—	yes	yes
BK/BB1	DBDH	$3\|\mathbb{G}_1\| + \|\mathsf{Mac}\|$	—	yes	—
BK/BB2	DBDHI	$3\|\mathbb{G}_1\| + \|\mathsf{Mac}\|$	—	yes	—
BK/Kiltz	DLIN	$5\|\mathbb{G}_1\| + \|\mathsf{Mac}\|$	yes	yes	—
Kiltz(KEM)	DLIN	$4\|\mathbb{G}_1\|$	yes	—	—
BMW/BB1	DBDH	$2\|\mathbb{G}_1\|$	—	—	—
BMW/Waters	DBDH	$2\|\mathbb{G}_1\|$	—	—	—
T1/Kiltz	DLIN	$4\|\mathbb{G}_1\| + \|r\|$	yes	—	yes
T1′/BB1	DBDH	$2\|\mathbb{G}_1\| + \|r\|$	—	—	yes
T1′/Gentry	DABDHE	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\| + \|r\|$	—	—	yes
T2/BB1	DBDH	$2\|\mathbb{G}_1\| + \|hk\| + \|t\| + \|r\|$	—	yes	yes

† KD is Kurosawa-Desmedt [24]. Kiltz is Kiltz TBE [22]. Waters is Waters IBE [30]. Gentry is Gentry IBE [18]. All schemes in the table are PKEs, except that BMW/BB1 and Kiltz(KEM) [22] are CCA-secure KEMs. Such KEMs can combine with symmetric key encryption (SKE) (e.g. block ciphers run in CMC mode [20] or EME mode [21]) which has no overhead, however, such operations are usually computationally less efficient than passively secure SKEs combined with Macs. The cost of computing one-time signatures and symmetric key primitives are neglected here. p is the order of $\mathbb{G}_1$. $\mathbb{G}$ is a group where DDH problem is hard. $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair. One may assume $|\mathbb{G}| = 1024$ (or 160 by using elliptic curve) and $|\mathbb{G}_1| = 512$ and $|\mathbb{G}_2| = 1024$ for symmetric bilinear group. r is a random coin for chameleon hash functions and is chosen uniformly from $\mathbb{Z}_p$, where p is the order of $\mathbb{G}_1$. hk is a hash key, and t' is a hash value. For practical implementations of chameleon hash, the size of hk and t' can be further optimized by reusing the system parameters. One-time signatures based on number-theoretic assumptions are no better than T2.

Non-interactive CCA-Secure Threshold Decryption. If the underlying tag based primitive is publicly verifiable, one can easily build non-interactive threshold encryption against chosen ciphertext attack with our method. It is quite natural if one follows previous work, e.g., [11], and the details and concrete instantiations are omitted here. We note that alternatively, this can be achieved by building a tag-KEM [2], combined with a semantically secure DEM. However, one may need to take care on the security of underlying primitives to build the tag-KEM. A recent work [1] addresses this in more details.

6 Comparisons

We compare some typical PKE schemes in Table 1. Note that our transforms can be further improved according to the discussions in Section 3.2, thus we come to conclusion that our transforms are efficient and widely applicable.

Acknowledgement

We thank Nuttapong Attrapadung, Jun Furukawa, Goichiro Hanaoka and Isamu Teranishi for helpful discussions. We thank Masayuki Abe and Eike Kiltz for pointing out the existence of Kiltz KEM and helping us discover unprecise presentations of Table 1 in a previous draft. We also thank anonymous referees of ACNS'07 for many helpful comments.

References

1. M. Abe, Y. Cui, H. Imai, and E. Kiltz. Efficient Hybrid Encryption from ID-Based Encryption. Cryptology ePrint Archive, `http://eprint.iacr.org/2007/023/`, 2007.
2. M. Abe, R. Gennaro, and K. Kurosawa. Tag-KEM/DEM: A New Framework for Hybrid Encryption. Cryptology ePrint Archive, `http://eprint.iacr.org/2005/027/`, 2005.
3. J.H. An, Y. Dodis, and Tal Rabin. On the Security of Joint Signature and Encryption. In *Eurocrypt'02*, volume 2332 of *LNCS*, pages 83–107. Springer, 2002.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO'98*, volume 1462 of *LNCS*. Springer, 1998.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, pages 62–73. ACM Press, 1993.
6. D. Boneh and X. Boyen. Efficient Selective-ID Identity Based Encryption without Random Oracles. In *EUROCRYPT'04*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.
7. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *CRYPTO'04*, volume 3152 of *LNCS*, pages 443–459, 2004.
8. D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security From Identity-Based Encryption. *SIAM Journal on Computing*, 36(5):915–942, 2006.
9. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO'01*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

10. D. Boneh and J. Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In *CT-RSA'05*, volume 3376, pages 87–103. Springer, 2005.
11. X. Boyen, Q. Mei, and B. Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. In *ACM CCS'05*, pages 320–329. ACM Press, 2005.
12. R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. In *EUROCRYPT'03*, volume 2656 of *LNCS*, pages 255–273. Springer, 2003.
13. R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *EUROCRYPT'04*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.
14. C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *the 8th IMA international conference on cryptography and coding*, volume 2260 of *LNCS*, pages 360–363. Springer, 2001.
15. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.
16. R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Chosen Ciphertext Secure Public Key Encryption . In *EUROCRYPT'02*, volume 2332 of *LNCS*, pages 45–64. Springer, 2002.
17. D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. In *STOC'91*, pages 542–552. ACM, 1991. Full version in SIAM Journal on Computing, 30(2):391-437, 2000.
18. C. Gentry. Practical Identity-Based Encryption Without Random Oracles. In *EUROCRYPT'06*, volume 4004 of *LNCS*, pages 445–464. Springer, 2006.
19. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
20. S. Halevi and P. Rogaway. A Parallelizable Enciphering Mode. In *Crypto'03*, volume 2729 of *LNCS*, pages 482–499. Springer, 2003.
21. S. Halevi and P. Rogaway. A Parallelizable Enciphering Mode. In *CT-RSA'04*, volume 2964 of *LNCS*, pages 292–304. Springer, 2004.
22. E. Kiltz. Chosen-Ciphertext Security from Tag-Based Encryption. In *TCC'06*, volume 1070 of *LNCS*, pages 581–600. Springer, 2006.
23. H. Krawczyk and T. Rabin. Chameleon Hashing and Signatures. 1997. Cryptology ePrint Archive Report. Available at `http://eprint.iacr.org/1998/010`.
24. K. Kurosawa and Y. Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *CRYPTO'04*, volume 3152 of *LNCS*, pages 426–442. Springer, 2004.
25. M. Naor and M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *STOC'90*, pages 427–437. ACM Press, 1990.
26. P. Paillier and D. Vergnaud. Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log. In *Asiacrypt'05*, volume 3788 of *LNCS*, pages 1–20. Springer, 2005.
27. C. Rackoff and D.R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, 1991.
28. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *FOCS'99*, pages 543–553. IEEE Computer Society, 1999.
29. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1997.
30. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

Certified E-Mail Protocol in the ID-Based Setting*

Chunxiang Gu, Yuefei Zhu, and Yonghui Zheng

Network Engineering Department, Zhengzhou Information Science and Technology Institute
P.O. Box 1001-770, Zhengzhou, 450002, P.R. China
gcxiang5209@yahoo.com.cn

Abstract. ID-based public key cryptosystem can be a good alternative for certificate-based public key setting, especially when efficient key management and moderate security are required. Certified e-mail protocols provide for fair exchange in which the intended recipient gets the e-mail content if and only if the mail originator receives an irrefutable receipt for the e-mail. In this paper we present an optimistic certified e-mail protocol in an ID-based setting. The protocol makes use of verifiable encryption of ID-based digital signatures as building blocks. We offer arguments for the fairness, efficiency, and provable security of our new protocol.

Keywords: ID-based cryptography, certified e-mail protocols, verifiably encrypted signatures, bilinear pairings.

1 Introduction

1.1 ID-Based Public Key Cryptography

ID-based public key cryptography (ID-PKC) is a paradigm proposed by Shamir in 1984 [1] to simplify key management and remove the necessity of public key certificates. In ID-PKC, an entity's public key is directly derived from certain aspects of its identity, such as an IP address belonging to a network host or an e-mail address associated with a user. That is, the user's public key can be calculated directly from his/her identity rather than being extracted from a certificate issued by a certificate authority. Private keys are generated for entities by a trusted third party, which is called a private key generator (PKG). The direct derivation of public keys in ID-PKC eliminates the need for certificates and some of the problems associated with them.

In 2001, the first entire practical and secure ID-based public key encryption scheme was presented in [2] by Boneh and Franklin, who took advantage of the properties of suitable bilinear maps (the Weil or Tate pairing) over supersingular elliptic curves. Since then, a rapid development of ID-PKC has taken place.

* Research supported by NSFC (No.60473021), the Natural Science Fundation of Henan Province (072300410260).

Many other ID-based primitives based on pairings have been proposed: digital signatures, authenticated key exchange, non-interactive key agreement, blind and ring signatures, signcryption, and so on. ID-based public key cryptography has become a good alternative for certificate-based public key setting, especially when efficient key management and moderate security are required.

1.2 Certified E-Mail Protocols

Exchanging items over insecure networks is considered a difficult problem, called the *fair exchange problem. Fairness* means that at no point during the execution of the protocol can either of the entities participating in the exchange gain any (significant) advantage over the other if the protocol is suddenly halted.

There have been two main approaches for achieving fair exchange. The first approach is to ensure that the exchange occurs simultaneously. One way of providing simultaneous exchange is to have the participants exchange information bit-by-bit in an interleaving manner [3].

The second approach is to ensure that the exchange will be completed even though one of the entities participating in the exchange refuses to continue. Fair exchange protocols which employ this approach require a trusted third party (TTP) as arbitrator. The use of the on-line TTP greatly reduces the efficiency of the protocol. With the assumption that the participators are honest in most situations, more preferable solutions, called *optimistic fair exchange protocols* based on off-line TTP, are proposed in [4,5]. In these protocols, the off-line TTP does not participate in the actual exchange protocol in normal cases, and is invoked only in abnormal cases to dispute the arguments.

Certified e-mail protocols are closely related to fair exchange protocols. In a certified e-mail system, the intended recipient gets the e-mail content if and only if the mail originator receives an irrefutable receipt for the e-mail from the recipient. A certified e-mail protocol should minimally provide [6]:

- **Fairness:** No party should be able to interrupt or corrupt the protocol to force an outcome to his/her advantage. That is, the protocol should terminate with either party having obtained the desired information, or with neither one acquiring anything. useful.
- **Monotonicity:** Each exchange of information during the protocol should add validity to the final outcome. The protocol should not require any messages, certificates, or signatures to be revoked to guarantee a proper termination of the protocol.
- **TTP invisibility:** A TTP is visible if the end result of an exchange makes it obvious that the TTP participated during the protocol.
- **Timeliness:** It guarantees that both parties will achieve their desired items in the exchange within finite time.
- **Confidentiality** (optional)**:** When confidentiality is needed, only the intended receiver can obtain the message content. No other parties including TTP can get it.

In recent years, fruitful achievements have been made in this field. On-line certified e-mail protocols are presented in [7,8,9]. Micali registered in U.S patent off-line optimistic protocols for fair exchange [10] including a certified e-mail protocol. Riordan and Schneier [11] present a protocol where the TTP acts as a public publishing location (which might be implemented as a secure database server). The authors describe both an on-line and an off-line version of the protocol. Ateniese et al. [12] suggested an off-line protocol which allows a stateless recipient: when a fairness problem occurs, a sender should take action to resolve the problem while the fairness on the recipients side is always guaranteed. Later, Park et al. [13] proposed an off-line protocol suitable for mobile environments which keeps the same properties including four passes as in Ateniese et al. [12] except that it reduces the recipient computation load. However, all these above works are in traditional certificate-based public key setting.

1.3 Contributions and Organization

In this paper, we present an optimistic certified e-mail protocol in ID-based public key setting. The paper is organized as follows. In Section 2, we present a verifiable encryption of ID-based signatures. In Section 3, we present an optimistic certified e-mail protocol in ID-based setting. We provide protocol analysis and comparisons in Section 4. Finally, we conclude in Section 5.

2 Cryptographic Blocks

2.1 Bilinear Maps

Let $(G_1,+)$ and $(G_2,\cdot)$ be two cyclic groups of order q, $\hat{e}: G_1 \times G_1 \rightarrow G_2$ be a map which satisfies the following properties.

1. Bilinear: $\forall P,Q \in G_1, \forall \alpha,\beta \in Z_q, \hat{e}(\alpha P, \beta Q) = \hat{e}(P,Q)^{\alpha\beta}$;
2. Non-degenerate: If P is a generator of G_1, then $\hat{e}(P,P)$ is a generator of G_2;
3. Computable: There is an efficient algorithm to compute $\hat{e}(P,Q)$ for any $P,Q \in G_1$.

Such an bilinear map is called an *admissible bilinear pairing.* The Weil pairings and the Tate pairings of elliptic curves can be used to construct efficient admissible bilinear pairings. Let P be a generator of G_1. The **computational Diffie-Hellman problem (CDHP)** is to compute abP for any given $P, aP, bP \in G_1$. We assume through this paper that there is no polynomial time algorithm to solve CDHP with non-negligible probability.

2.2 A Verifiable Encryption of ID-Based Signatures

In the design of optimistic fair exchange protocols, ***verifiably encrypted signature schemes*** (VESSs) are usually being used as the kernel building blocks. VESS is a special extension of general signature primitive, which enables user Alice to give user Bob a signature encrypted with an *adjudicator*'s public key, and enables Bob to verify that the encrypted signature indeed contains such a signature. The adjudicator is an off-line TTP, who can reveal the signature when needed.

Based on the ID-based signature scheme due to Cha and Cheon [14], we provide an efficient ID-based VESS as following:

- ***Setup*:** Given a security parameter $\lambda \in N$, generate $(G_1, G_2, q, \hat{e}, P)$, pick a random $s \in Z_q^*$ and set $P_{pub} = sP$. Choose two hash functions $H_1 : \{0,1\}^* \to G_1^*$, $H_2 : \{0,1\}^* \times G_1 \to Z_q$. The system parameters $\Omega = (G_1, G_2, q, \hat{e}, P, P_{pub}, H_1, H_2)$. The master key (PKG's private key) is s.
- ***Extract*:** Given an identity $ID_X \in \{0,1\}^*$, compute $Q_X = H_1(ID_X) \in G_1^*$, $D_X = sQ_X$. PKG uses this algorithm to extract the user secret key D_X, and gives D_X to the user by a secure channel.
- ***Sign*:** Given a private key D_X and a message m, pick $k \in Z_q^*$ at random, compute $U = k \cdot Q_X$, $h = H_2(m, U)$, $V = (k+h)D_X$, and output a signature (U, V).
- ***Verify*:** Given a signature (U, V) of an identity ID_X for a message m, compute $h = H_2(m, U)$, and accept the signature if and only if $\hat{e}(P, V) = \hat{e}(P_{pub}, U + h \cdot Q_X)$.
- ***Adj_KGen*:** Given an adjudicator's private key D_T, pick $s_T \in Z_q^*$ at random, compute $P_T = s_T \cdot P$, and $\varpi = Sign(D_T, P_T)$, output the adjudication warrant (P_T, ϖ) and the adjudication key s_T.
- ***VE_Sign*:** Given a secret key D_X, a message $m \in \{0,1\}^*$ and an adjudication warrant (P_T, ϖ) of ID_T,
  1. choose $k_1, k_2 \in Z_q^*$ at random, and compute $U = k_1 \cdot Q_X$, $Y = k_2 P$,
  2. compute $h = H_2(m, U)$, $V' = (k_1 + h)D_X + k_2 P_T$
  3. output the verifiably encrypted signature (U, V', Y, P_T, ϖ).
- ***VE_Verify*:** Given a verifiably encrypted signature (U, V', Y, P_T, ϖ) of ID_X for message m, compute $h = H_2(m, U)$, and accept the signature if and only if $Verify(ID_T, P_T, \varpi) = 1$ and $\hat{e}(P, V') = \hat{e}(P_{pub}, U + h \cdot Q_X) \cdot \hat{e}(Y, P_T)$.
- ***Adjudication*:** Given the adjudication key s_T, and a valid verifiably encrypted signature (U, V', Y, P_T, ϖ) of ID_X for message m, compute $V = V' - s_T Y$, and output the original signature (U, V)

Validity requires that verifiably encrypted signatures and adjudicated verifiably encrypted signatures verify as ordinary signatures, i.e., for $\forall m \in \{0,1\}^*$, $ID_X, ID_T \in \{0,1\}^*$, $D_X = Extract(ID_X, s)$, $D_T = Extract(ID_T, s)$, $((P_T, \varpi), s_T) = Adj\_KGen(D_T)$, satisfying:

1. $VE\_Verify(ID_X, ID_T, m, VE\_Sign(D_X, m, (P_T, \varpi))) = 1$;
2. $Verify(ID_X, m, Adjudication(s_T, VE\_Sign(D_X, m, (P_T, \varpi)))) = 1$.

The correctness of our scheme is easily proved as follows. For a verifiably encrypted signature (U, V', Y, P_T, ϖ) of an identity ID_X for a message m, $Verify(ID_T, P_T, \varpi) = Verify(ID_T, P_T, Sign(D_T, P_T)) = 1$, and

$$\begin{aligned}\hat{e}(P, V') &= \hat{e}(P, (k_1 + h)D_X + k_2 P_T)) \\ &= \hat{e}(P_{pub}, (k_1 + h) \cdot Q_X) \cdot \hat{e}(k_2 \cdot P, P_T) \\ &= \hat{e}(P_{pub}, U + h \cdot Q_X) \cdot \hat{e}(Y, P_T)\end{aligned}$$

That is $VE\_Verify(ID_X, ID_T, m, VE\_Sign(D_X, m, (P_T, \varpi))) = 1$.

On the other hand, $V = V' - s_T Y = (k_1 + h)D_X + k_2 P_T - s_T k_2 P = (k_1 + h)D_X$. So we have $\hat{e}(P, V) = \hat{e}(P, (k_1 + h)D_X) = \hat{e}(P_{pub}, U + h \cdot Q_X)$. Hence, $Verify(ID_X, m, Adjudication(s_T, VE\_Sign(D_X, m, (P_T, \varpi)))) = 1$.

Readers can see that (***Setup***, ***Extract***, ***Sign***, ***Verify***) in the above scheme constitute the Cha-Cheon's ID-based signature scheme [14]. In fact, the above way of constructing ID-based VESS can be applied to a kind of ID-based signature schemes with the following property:

– Given input an identity ID_X and a message m, the signing algorithm generates a signature (σ_1, σ_2), and the verification equation can be described as $\hat{e}(P, \sigma_2) = f(ID_X, m, \sigma_1)$, where $f(.)$ is a determinable polynomial-time function.

The Construction

– ID-based signature scheme IBS={*Setup,Extract,Sign,Verify*}.
– ***Adj_KGen*:** the same as that in the above scheme.
– ***VE_Sign*:** Given a secret key D_X, a message m and a warrant (P_T, ϖ),
1. compute $(\sigma_1, \sigma_2) = Sign(m, D_X)$.
2. choose $k_2 \in Z_q^*$ at random, and compute $Y = k_2 P$, $\gamma = \sigma_2 + k_2 P_T$;
3. output the verifiably encrypted signature $(\sigma_1, \gamma, Y, P_T, \varpi)$.
– ***VE_Verify*:** Given a $(\sigma_1, \gamma, Y, P_T, \varpi)$ of ID_X for message m, compute $h = H_2(m, U)$, and accept the signature if and only if $Verify(ID_T, P_T, \varpi) = 1$ and $\hat{e}(P, \gamma) = f(ID_X, m, \sigma_1) \cdot \hat{e}(Y, P_T)$.
– ***Adjudication*:** Given the adjudication key s_T, and a (U, V', Y, P_T, ϖ) of ID_X for message m, compute $\sigma_2 = \gamma - s_T Y$ and output (σ_1, σ_2)

The construction can be applied to many existing ID-based signature schemes, such as Paterson's scheme [15], Hess's scheme [16], Cheon-Kim-Yoon's scheme [17], and so on.

Some general performance enhancements can be applied to our scheme. Pairings are usually been constructed with the Weil pairings or the Tate pairings of (hyper)elliptic curves. For a fixed $R \in G_1$, there are efficient algorithms [18] to compute kR by pre-computation. We may assume that such a computation is at most 1/5 an ordinary scalar multiplication in $(G_1, +)$. In our scheme, P, P_{pub} are fixed. The signer's private key and public key are fixed for himself. In most instances, P_T is also fixed and the verifier need not re-verify (P_T, ϖ) every time. We compare our ID-based VESS with the schemes in [19,20] (not ID-based) in the following table. (Denote by M a scalar multiplication in $(G_1, +)$, by E an Exp. operation in $(G_2, .)$, and by $\hat{e}$ a computation of the pairing. For RSA-based scheme [20], denote by Exp an Exp. operation.)

	VE_Sign	*VE_Verify*	*Adjudication*	certificate
Boneh [19]	$1.4M$	$3\hat{e}$	$1M$	needed
Nenadic [20]	$6Exp$	$2Exp$	$1Exp$	needed
Proposed	$0.8M$	$3\hat{e} + 1M$(or $5\hat{e} + 2M$)	$1\hat{e} + 1M$	not needed

In fact, the basic idea of the encryption is similar to the work of Zhang et.al. [21] and the work of Gu et.al. [22]. However, in their works, there is only one TTP (i.e. adjudicator) in the system, and his key pair is not ID-based.

3 An Efficient ID-Based Optimistic Protocol

In a certified e-mail protocol, the sender Alice sends an e-mail m to Bob and wants to obtain a receipt for it. The recipient Bob should not obtain m without issuing the receipt.

Using the ID-based VESS described in Section 2 as building block, we present an ID-based optimistic certified e-mail protocol. We will assume that the communication is carried over private and authenticated channels.

Let Alice be the sender with identity ID_A and secret key D_A. Bob is the recipient whose identity is ID_B and secret key is D_B. The identity of TTP is ID_T and the corresponding secret key is D_T. TTP runs $Adj\_KGen(D_T)$ to generate adjudication warrant (P_T, ϖ) and adjudication key s_T. $H(.)$ is a suitable hash function. Our new ID-based optimistic certified e-mail protocol works as following (shown in Figure 1):

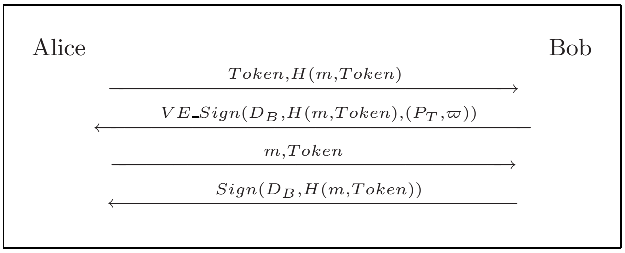


Fig. 1. ID-based optimistic certified e-mail protocol

- Step1: Alice makes a $Token$ which contains relevant information such as ID_A, ID_B and ID_T and other pertinent information the protocol, and sends $(Token, H(m, Token))$ to Bob.
- Step2: Bob computes a verifiably encrypted signature

$$\pi_B = VE\_Sign(D_B, H(m, Token), (P_T, \varpi)),$$

  and sends π_B to Alice.
- Step3: Alice sends $(m, Token)$ to Bob if

$$VE\_Verify(ID_B, ID_T, H(m, Token), \pi_B) = 1.$$

  Otherwise, Alice aborts or ask Bob to re-send.
- Step4: Bob computes an ordinary signature $\delta_B = Sign(D_B, H(m, Token))$ as receipt, and sends δ_B to Alice.

If Bob does not send the receipt δ_B in Step 4, then Alice can contact the TTP for adjudication by running the following **Dispute** protocol:

- Step1: Alice sends π_B and $(m, Token)$ to TTP.
- Step2: TTP reads the $Token$ and verifies the validity of π_B. Then, he computes $\delta_B = Adjudication(s_T, \pi_B)$, and sends δ_B to Alice and $(m, Token)$ to Bob.

In case of dispute, Alice has to reveal the message m to the TTP. If message privacy has to be preserved, it is sufficient to substitute m with $En_B(m)$ in the protocol, where $En_B()$ represents the ID-based encryption under Bob's identity. To improve the efficiency, $En_B(m)$ can be implemented with $(E_k(m), IBE_B(k))$, where $IBE_B(.)$ is an ID-based encryption, such as Boneh-Franklin scheme [2], and $E_k(.)$ is a symmetric-key encryption algorithm, such as AES.

In the protocol, TTP works in an *optimistic* way. That is, TTP does not participate in the actual exchange protocol in normal cases (no argument appears), and is invoked only in abnormal cases to dispute the arguments for fairness. If no dispute occurs, only Alice and Bob need to participate in the exchange.

4 Analysis and Comparisons

It is easy to show that the protocol above is a certified e-mail protocol which provides TTP invisibility, monotonicity and timeliness. Moreover, the protocol optionally provides confidentiality of the message.

– **TTP invisibility:** Clearly our protocol provides TTP invisibility since the structure of the receipt does not indicate whether the TTP was involved or not in dispute resolutions.
– **Monotonicity:** The protocol provides also monotonicity since any signature (including the receipt) will not be revoked in order to guarantee a proper termination of the protocol.
– **Confidentiality:** Confidentiality is achieved by encrypting the actual message content in such a way that only the recipient can open it and this is achieved through standard encryption technology.
– **Timeliness:** We assume only resilient channels. A resilient channel will eventually deliver a message sent through it within a time lapse which may be arbitrarily long, yet finite. Moreover, the recipient does not need to include any time limit into the signature SB and the sender A has the ability to decide to abort the protocol and adopt a scheme for protocol resolution that can be executed in a finite period of time. Therefore, our protocol provides timeliness.

4.1 Fairness

The protocol's fairness is built around the assumption that the sender Alice can verify that the verifiable encryption indeed contains a valid receipt. Only the TTP can recover the receipt from the verifiable encryption. That is, the ID-based VESS is secure.

Besides the ordinary notion of signature security in the signature component, Boneh et.al. [19] proposed two security properties of verifiably encrypted signatures: **Unforgeability** and **Opacity**. Informally, Unforgeability requires that it is difficult to forge a valid verifiably encrypted signature for a new message, while Opacity means it is difficult, given a verifiably encrypted signature, to extract an ordinary signature on the same message. In this section, we extend these security notions to ID-based VESS.

We consider an adversary $\mathcal{F}$ which is assumed to be a polynomial time probabilistic Turing machine which takes as input the global scheme parameters and a random tape. To aid the adversary we allow it to query the following oracles:

– Extract oracle $E(.)$: For input an identity ID_X, this oracle outputs the corresponding secret key D_X
– Adj_KGen oracle $AK(.)$: For input an adjudicator's identity ID_T, this oracle computes and outputs an adjudication warrant (P_T, ϖ) of ID_T.
– VE_Sign oracle $VS(.)$: For input $(ID_X, m, (P_T, \varpi))$, this oracle computes and outputs a verifiably encrypted signature π.
– Adjudication oracle $A(.)$: For input ID_X, m and a valid verifiable encrypted signature π of ID_X for m with an adjudication warrant (P_T, ϖ) of ID_T, this oracle computes and outputs the corresponding ordinary signature δ.

Note: An ordinary signing oracle is not provided, because it can be simulated by a call to $VS(.)$ followed by a call to $A(.)$. In the random oracle model, $\mathcal{F}$ also has the ability to issue queries to the hash function oracles $H_1(.), H_2(.)$ adaptively.

Definition 1. *The advantage in existentially forging a verifiably encrypted signature of an adversary $\mathcal{F}$ is defined as*

$$Adv_{\mathcal{F}}^{EUF}(k) = Pr\left[\begin{array}{c} \Omega \leftarrow Setup(1^\lambda), \\ (ID_X, ID_T, m, \pi) \leftarrow \mathcal{F}^{H_1(.),H_2(.),E(.),AK(.),VS(.),A(.)}(\Omega) : \\ VE\_Verify(ID_X, ID_T, m, \pi) = 1, \\ (ID_X, .) \notin E_l, (ID_X, m, .) \notin \mathcal{O}_l, \end{array}\right]$$

*where E_l is the query and answer list coming from $E(.)$, and $\mathcal{O}_l$ is the query and answer lists of $AK(.)$, $VS(.)$ and $A(.)$ during the attack. The probability is taken over the coin tosses of the algorithms, of the oracles, and of the forger. An ID-based VESS is said to be **existential unforgeable**, if for any adversary $\mathcal{F}$, $Adv_{\mathcal{F}}^{EUF}(k)$ is negligible.*

Definition 2. *The advantage in opacity attack of an algorithm $\mathcal{F}$ is defined as*

$$Adv_{\mathcal{F}}^{OPA}(k) = Pr\left[\begin{array}{c} \Omega \leftarrow Setup(1^\lambda), \\ (ID_X, ID_T, m, \pi, \delta) \leftarrow \mathcal{F}^{H_1(.),H_2(.),E(.),AK(.),VS(.),A(.)}(\Omega) : \\ VE\_Verify(ID_X, ID_T, m, \pi) = 1, Verify(ID_X, m, \delta) = 1, \\ A(ID_X, m, \pi) = \delta, (ID_X, .) \notin E_l, \\ (ID_X, m, .) \notin A_l, (ID_X, m, .) \notin AK_l \end{array}\right]$$

where A_l and AK_l are the query and answer lists coming from $A(.)$ and $AK(.)$ during the attack (We note that $\mathcal{F}$ can request $VS(.)$ with input $(ID_X, m, .)$). The probability is taken over the coin tosses of the algorithms, of the oracles, and of the forger. An ID-based VESS is said to be ***opaque****, if for any adversary $\mathcal{F}$, $Adv_{\mathcal{F}}^{OPA}(k)$ is negligible.*

Theorem 1. *In the random oracle model, if there is an adversary $\mathcal{F}_0$ which performs, within a time bound T, an existential forgery against our ID-based VESS with probability ε, then there is an adversary $\mathcal{F}_1$ which performs an existential forgery against Cha-Cheon's scheme with probability no less than ε, within a time bound $T + (3n_{AK} + 2n_{VS} + n_A)M$, where n_{VS}, n_A and n_{AK} are the number of queries that $\mathcal{F}_0$ can ask to $VS(.)$, $A(.)$ and $AK(.)$ respectively, M denotes a scalar multiplication in G_1.*

Proof. see the appendix A.

Theorem 2. *In the random oracle mode, let $\mathcal{F}_0$ be an adversary which has running time T and success probability ε in opaque attack. We denote by n_{h_1}, n_E, n_{AK}, n_A and n_{VS} the number of queries that $\mathcal{F}_0$ can ask to the oracles $H_1(.)$, $E(.)$, $AK(.)$, $A(.)$ and $VS(.)$ respectively. Then there is a polynomial-time Turing machine $\mathcal{F}_1$ who can solve the computational Diffie-Hellman problem within expected time $T + (n_{h_1} + n_E + 3n_{AK} + n_A + 5n_{VS})M$ with probability $\varepsilon/(n_{h_1} \cdot n_{AK} \cdot n_{VS})$.*

Proof. see the appendix B.

Now, let's discuss the fairness of our protocol. At the end of Step 2, if Alice aborts after receiving $\pi_B = VE\_Sign(D_B, H(m, Token), (P_T, \varpi))$, Alice can't get the receipt $\delta_B = Sign(D_B, H(m, Token))$ by himself. If Alice requests to TTP for dispute with valid π_B and $(m, Token)$, TTP computes $\delta_B = Adjudication(s_T, \pi_B)$, sends $(m, Token)$ to Bob and sends δ_B to Alice. That is, either party gets what he wants, or neither party does. At the end of Step3, if Alice has send Bob with $(m, Token)$ while hasn't received the receipt δ_B, Alice can request to TTP for dispute with valid π_B and $(m, Token)$. As a result, Alice will receive a receipt on $(m, Token)$ and Bob will get the message $(m, Token)$.

4.2 Comparison

We compare now our protocol with some previously proposed protocols.

On-line certified e-mail protocols, such as [7,8,9], require TTP participates in all the transactions. The use of the on-line TTP greatly reduces the efficiency of such protocols.

Some of the off-line protocols are not monotonic, for instance, the protocol in [23] requires signatures to be revoked in order to guarantee fairness. Some solutions, such as [11], require a visible TTP, since the form of the receipt changes depending on whether the trusted entity was invoked or not. The work of Micali [10] shows that it is possible to achieve a simple certified e-mail protocol with only three messages. However, the protocol needs to transmit data with an actual message size more than two times, and a time limit has to be incorporated into the message by the sender to force the recipient to send the receipt within a specified period of time. Furthermore, for each message received, the recipient is forced to communicate with the trusted intermediary in case of dispute and such a communication has to happen before the time limit expires. Asokan *et.al.* [24] present a fair-exchange protocol which is provably secure in the random oracle model. However, the scheme seems expensive in terms of communication complexity, performance, and amount of data transmitted. This is mainly due to the cut-and-choose interactive proof technique employed to achieve a verifiable escrow.

Recently, Ateniese et al. [6] suggested an off-line protocol with four passes which allows a stateless recipient. In fact, our protocol keeps almost the same properties as that in [6]. Comparatively, our protocol works in ID-based setting, and is more efficient in performance. What more, ID-based cryptography can bring us simple key management and remove the necessity of public key certificates in our certified e-mail systems. However, our protocol also have the key escrow problem which is a inherent property of ID-based cryptography. This can be solved to a certain extent by the introduction of multiple PKGs and the use of threshold techniques.

5 Conclusion

In this paper we present an optimistic certified e-mail protocol in ID-based setting. In the protocol, the trusted third party (TTP) works in an *optimistic* way. That is, TTP does not participates in the actual exchange protocol in normal cases (no argument appears), and is invoked only in abnormal cases to dispute the arguments for fairness. The protocol provides fairness, TTP invisibility, monotonicity, timeliness and confidentiality (optionally). What's more, we believe it is easy to amend the protocol to be used in exchanging other digital items and the corresponding digital signatures in the e-commerce with fairness and efficiency in ID-based setting.

References

1. A. Shamir. Identity-based cryptosystems and signature schemes. In Advances in Cryptology - CRYPTO'84, LNCS 196, pages 47-53. Springer-Verlag, 1984.
2. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, Advances in Cryptology- CRYPTO 2001, LNCS 2139, pages 213-229. Springer-Verlag, 2001.

3. T.Okamoto and K.Ohta. How to Simultaneously Exchange Secrets by General Assumption. In Proc. of the 2nd ACM Conference on Computer and Communications Security, pages 184-192, 1994.
4. F.Bao, R.Deng, W.Mao. Efficient and practical fair exchange protocols with off-line TTP. In: Proc. of the 1998 IEEE Symp. on Security and Privacy. Oakland: IEEE Computer Press, 77-85. 1998.
5. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. IEEE J. Selected Areas in Comm., 18(4):593-610, 2000.
6. G. Ateniese and C. Nita-Rotaru, Stateless-Recipient Certified E-mail System Based on Verifiable Encryption, CT-RSA 2002, LNCS vol.2271, pp.182-199, 2002.
7. A. Bahreman and J. D. Tygar, Certified electronic mail, Proc. of Symposium on Network and Distributed Systems Security, pp. 3-19, 1994.
8. R. H. Deng, L. Gong, A. Lazar, and W. Wang, Practical protocols for certified electronic e-mail, Journal of Networks and Systems Management, vol. 4, no. 3, pp. 279-297, 1996.
9. J. Zhou and D. Gollmann, Certified electronic mail, In Proc. of Computer Security - ESORICS96, pp. 55-61, 1996.
10. S. Micali, Simultaneous electronic transacions, US Patent, no.5666420, 1997. http://www.delphion.com/cgi-bin/viewpat.cmd/US566420.
11. J. Riordan and B. Schneier, A certified e-mail protocol, in 13th Annual Computer Security Applications Conference, pp. 100-106, 1998.
12. G. Ateniese, Efficient verifiable encryption (and fair exchange) of digital signatures. In: Proc. the 6th ACM Conference on Computer and Communications Security, ACM Press, New York, USA, 138-146. 1999.
13. J. M. Park, I. Ray, E. K. P. Chong, and H. J. Siegel, A Certified E-Mail Protocol Suitable for Mibile Environments, IEEE Global Telecommunications Conference, vol.22, no.1, pp.1394-1398, 2003.
14. J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In: Public Key Cryptography - PKC 2003, LNCS 2567, pages 18-30. Springer-Verlag, 2003.
15. K. G. Paterson. ID-Based Signatures from Pairings on Elliptic Curves. Electron. Lett., Vol. 38, No. 18, pp. 1025-1026, 2002.
16. F. Hess. Efficient identity based signature schemes based on pairings. In: SAC 2002, LNCS 2595, pages 310-324. Springer-Verlag, 2003.
17. J. H. Cheon, Y. Kim, H. Yoon. Batch Verifications with ID-based Signatures. In: ICISC 2004, LNCS 3506, pp.233-248. Springer-Verlag, 2005.
18. Y.Sakai, K.Sakurai. Efficient Scalar Multiplications on Elliptic Curves without Repeated Doublings and Their Practical Performance. In: ACISP 2000, LNCS 1841, pages 59-73. Springer-Verlag Berlin, 2000.
19. D. Boneh, C. Gentry, B. Lynn and H. Shacham. Aggregate and Verifiably Encrypted Signature from Bilinear Maps. Eurocrypt 2003, LNCS 2248, pp. 514-532, Springer-Verlag, 2003.
20. A. Nenadic N. Zhang, B. Cheetham, and C. Goble. An RSA-based Security Protocol for Certified E-goods Delivery, Proc. IEEE International Conference on Information Technology 2004, Las Vegas, USA, IEEE Computer Society, 22-28. 2004.
21. Z. Zhang, D. Feng, J. Xu, and Y. Zhou. Efficient ID-based Optimistic Fair Exchanges with Provable Security. In: Proc. of ICICS 2005, LNCS 3783, pp. 14-26. Springer-Verlag, 2005.

22. Chunxiang Gu, Yuefei Zhu, and Yajuan Zhang. An ID-based Optimistic Fair Signature Exchange Protocol from Pairings. CIS 2005, Part II, LNAI 3802, pp. 9-16. Springer-Verlag, 2005.
23. N. Asokan, V. Shoup, and M. Waidner, Asynchronous protocols for optimistic fair exchange, Proc. IEEE Symposium on Research in Security and Privacy, pp. 86-99, 1998.
24. N. Asokan, V. Shoup, and M. Waidner, Optimistic fair exchange of digital signatures, IEEE Journal on Selected Area in Communications, 2000.

A Proof of Theorem 1

Proof. Without any loss of generality, we may assume that $\mathcal{F}_0$ queries $VS(.)$ with (ID_X, m, P_T, ϖ) or queries $A(.)$ with $(ID_X, m, \pi = (U, Y, V', P_T, \varpi))$ only if $\mathcal{F}_0$ has queried $AK(.)$ with some identity ID_T and got reply (P_T, ϖ). From $\mathcal{F}_0$, we can construct an adversary $\mathcal{F}_1$ of Cha-Cheon's scheme as follows:

1. A challenger $\mathcal{C}$ runs $Setup(1^\lambda)$ of Cha-Cheon's scheme and gives the system parameters $\Omega = (G_1, G_2, q, \hat{e}, P, P_{pub}, H_1, H_2)$ to $\mathcal{F}_1$.
2. $\mathcal{F}_1$ runs $\mathcal{F}_0$ with input Ω. During the execution, $\mathcal{F}_1$ emulates $\mathcal{F}_0$'s oracles as follows:
   – $H_1(.), H_2(.), E(.)$: $\mathcal{F}_1$ replaces these oracles with his own $H_1(.), H_2(.), E(.)$ oracles respectively. That is , $\mathcal{F}_1$ asks his $H_1(.), H_2(.), E(.)$ oracles with the inputs of $\mathcal{F}_0$, and lets the outputs be the replies to $\mathcal{F}_0$, respectively.
   – $AK(.)$: For input ID_T, $\mathcal{F}_1$ randomly selects $s_T \in Z_q^*$, computes $P_T = s_T P$, requests to his signing oracle $Sign(.)$ with (ID_T, P_T) and gets reply ϖ. $\mathcal{F}_1$ outputs (P_T, ϖ) as the reply.
   – $VS(.)$: For input $(ID_X, m, (P_T, \varpi))$, $\mathcal{F}_1$ requests to his own signing oracle $Sign(.)$ with input (ID_X, m) and gets reply (U, V), then he picks randomly $r \in Z_q^*$, and outputs $(U, V + rP_T, rP, (P_T, \varpi))$ as the reply.
   – $A(.)$: For input $(ID_X, m, \pi = (U, V', Y, P_T, \varpi))$, $\mathcal{F}_1$ computes $V = V' - s_T Y$, (with assumption, (P_T, ϖ) is the reply of some request to $AK(.)$, so $\mathcal{F}_1$ knows s_T.) and replies with (U, V).
3. If $\mathcal{F}_0$ outputs $(ID^*, m^*, (U, V', Y, P_T, \varpi))$, then $\mathcal{F}_1$ outputs $(ID^*, m^*, (U, V))$, where $V = V' - s_T Y$.

If $\mathcal{F}_0$'s has not queried oracles $E(.)$ with ID^*, and has not queried oracles $AK(.)$, $A(.)$ and $VS(.)$ with input (or part of input) (ID^*, m^*), then $\mathcal{F}_1$ has not queried oracles $E(.)$ and $Sign(.)$ with input ID^* and (ID^*, m^*) respectively. If (U, V', Y, P_T, ϖ) is a valid verifiably encrypted signature of ID^* for m^*, then $(U, V' - s_T Y)$ is a valid signature of ID^* for m^*. That is , $\mathcal{F}_1$ succeeds in existential forgery against Cha-Cheon's scheme with probability no less than that of $\mathcal{F}_0$ succeeds in his game.

$\mathcal{F}_1$'s running time is roughly the same as $\mathcal{F}_0$'s running time plus the time taken to respond to $\mathcal{F}_0$'s oracle queries. Neglect operations other than the pairing $\hat{e}(*, *)$ and the scalar multiplication in $(G_1, +)$, the total running time is $T + (3n_{AK} + 2n_{VS} + n_A)M$ as required.

B Proof of Theorem 2

Proof. We may assume that for any ID, $\mathcal{F}_0$ queries $H_1(.)$ with ID before ID is used as (part of) an input of any query to $E(.)$, $AK(.)$, $VS(.)$, or $A(.)$. From the adversary $\mathcal{F}_0$, we construct a Turing machine $\mathcal{F}_1$ which can solve the CDHP as follows:

1. A challenger $\mathcal{C}$ generates $(G_1, G_2, q, \hat{e})$ and selects randomly $P, aP, bP \in G_1$. $\mathcal{C}$ gives $(G_1, G_2, q, \hat{e}, P, aP, bP)$ to $\mathcal{F}_1$ as inputs.
2. $\mathcal{F}_1$ selects hash functions $H_1 : \{0,1\}^* \rightarrow G_1^*$, $H_2 : \{0,1\}^* \times G_1 \rightarrow Z_q$, selects randomly $\mu \in Z_q^*$ and sets $P_{pub} = \mu bP$.
3. $\mathcal{F}_1$ sets variables $v = 1, j = 1, r = 1$ and list $V_l = \varPhi$ ($\varPhi$ denotes NULL).
4. $\mathcal{F}_1$ picks randomly t, γ and ι satisfying $1 \leq t \leq n_{h_1}$, $1 \leq \gamma \leq n_{AK}$, $1 \leq \iota \leq n_{VS}$, and picks randomly $x_i \in Z_q, i = 1, 2, ...n_{h_1}$.
5. $\mathcal{F}_1$ gives $\varOmega = (G_1, G_2, q, \hat{e}, P, P_{pub}, H_1, H_2)$ to $\mathcal{F}_0$ as input and lets $\mathcal{F}_0$ run on. During the execution, $\mathcal{F}_1$ simulates $\mathcal{F}_0$'s oracles as follows:
   - $H_1(.)$: For input ID, $\mathcal{F}_1$ checks if $H_1(ID)$ is defined. If not, he defines $H_1(ID) = x_v P$ and sets $ID_v \leftarrow ID$, $v \leftarrow v + 1$. $\mathcal{F}_1$ returns $H_1(ID)$ to $\mathcal{F}_0$.
   - $H_2(.)$: For input (m, U), $\mathcal{F}_1$ checks if $H_2(m, U)$ is defined. If not, he picks a random $h \in Z_q$, and sets $H_2(m, U) = h$. $\mathcal{F}_1$ returns $H_2(m, U)$ to $\mathcal{F}_0$.
   - $E(.)$: For input ID_i, $\mathcal{F}_1$ lets $d_i = x_i P_{pub}$ be the reply to $\mathcal{F}_0$.
   - $AK(.)$: For input ID_T, if $r = \gamma$, $\mathcal{F}_1$ lets $P_T = bP$; otherwise, $\mathcal{F}_1$ selects randomly $s_T \in Z_q^*$, $P_T = s_T P$. $\mathcal{F}_1$ computes $\varpi = Sign(D_T, P_T)$, sets $r \leftarrow r + 1$ and replies with (P_T, ϖ).
   - $VS(.)$: For input ID_i, message m and warrant (P_T, ϖ), $\mathcal{F}_1$ emulates the oracle as follows:
     - If $j = \iota$, $i = t$ and $P_T = bP$,
       a. Pick randomly $k_2, h \in Z_q^*$;
       b. $U = aP - h(H_1(ID_i))$, $Y = \mu(k_2 P - aP)$, $V' = \mu k_2(bP)$;
       c. If $H_2(m, U)$ has been defined, $\mathcal{F}_1$ aborts (a collision appears). Otherwise, set $H_2(m, U) = h$.
       d. Add $(i, j, ., U, V', Y, P_T, \varpi)$ to V_l.
     - Otherwise,
       a. Pick randomly $r_2, z_j, h \in Z_q^*$;
       b. Compute $U = z_j P - h(H_1(ID_i))$, $Y = k_2 P$, $V' = z_j P_{pub} + k_2 P_T$;
       c. If $H_2(m, U)$ has been defined, $\mathcal{F}_1$ aborts (a collision appears). Otherwise, set $H_2(m, U) = h$.
       d. Add $(i, j, k_2, U, V', Y, P_T, \varpi)$ to V_l.

     Set $j \leftarrow j + 1$ and let (U, V', Y, P_T, ϖ) be the reply to $\mathcal{F}_0$.
   - $A(.)$: For input ID_i, m and valid verifiably encrypted signature $\pi = (U, V', Y, P_T, \varpi)$, $\mathcal{F}_1$ obtains the corresponding item (i, j, k_2, π) (or $(i, j, ., \pi)$) from the V_l. If $i = t$, $j = \iota$ and $P_T = bP$, $\mathcal{F}_1$ declares failure and aborts. Otherwise, $\mathcal{F}_1$ computes $V = V' - k_2 P_T$, and replies to $\mathcal{F}_0$ with (U, V).

6. If $\mathcal{F}_0$'s output is $(ID_i, m^*, \pi^* = (U, V', Y, P_T, \varpi), \delta^* = (U, V))$, then $\mathcal{F}_1$ obtains the corresponding item (i, j, k_2, π^*) (or $(i, j, ., \pi^*)$) on the V_l. If $i = t$, $j = \iota$ and $P_T = bP$, $\mathcal{F}_1$ computes and successfully outputs $abP = \mu^{-1}V$. Otherwise, $\mathcal{F}_1$ declares failure and aborts.

This completes the description of $\mathcal{F}_1$.

Because of the randomness of U, the probability of $\mathcal{F}_1$ aborts as a result of collision of $H_2(m, U)$ is negligible. If $\mathcal{F}_0$ succeeds in his attack with output $(ID_i, m^*, \pi^* = (U, V', Y, P_T, \varpi), \delta^* = (U, V))$, and the corresponding item in V_l is $(t, \iota, ., \pi^*)$ and $P_T = bP$, then $\mu^{-1}V = \mu^{-1}a\mu bP = abP$. $\mathcal{F}_0$ succeed in his attack, so $\mathcal{F}_0$ has not query $A(.)$ with $(ID_t, m^*, .)$. Hence, $\mathcal{F}_1$'s simulations are indistinguishable form $\mathcal{F}_0$'s real oracles. Because t is chosen randomly in 1 and n_{h_1}, γ is chosen randomly in 1 and n_{AK}, and ι is chosen randomly in 1 and n_{VS}, $\mathcal{F}_1$ can output abP with probability $\varepsilon/(n_{h_1}n_{AK}n_{VS})$. It is easy to see that $\mathcal{F}_1$'s running time is roughly $T + (n_{h_1} + n_E + 3n_{AK} + n_A + 5n_{VS})M$ as required.

Efficient Content Authentication in Peer-to-Peer Networks*

Roberto Tamassia[1] and Nikos Triandopoulos[2]

[1] Department of Computer Science, Brown University
[2] Institute for Security Technology Studies, Dartmouth College

Abstract. We study a new model for data authentication over peer-to-peer (p2p) storage networks, where data items are stored, queried and authenticated in a totally decentralized fashion. The model captures the security requirements of emerging distributed computing applications. We present an efficient construction of a *distributed Merkle tree* (DMT), which realizes an authentication tree over a p2p network, thus extending a fundamental cryptographic technique to distributed environments. We show how our DMT can be used to design an *authenticated distributed hash table* that is secure against replay attacks and consistent with the update history. Our scheme is built on top of a broad class of existing p2p overlay networks and achieves generality by using only the basic functionality of object location. We use this scheme to design the first efficient *distributed authenticated dictionary.*

1 Introduction

Peer-to-peer (p2p) networks provide the basis for the design of fully decentralized systems, where data and computing resources are shared among participating peers. Properties of p2p networks include scalability, self-stabilization, data availability, load balancing, and efficient searching. As p2p networks become more mature and established and new applications emerge for them, the need increases for their security.

In this paper, we study data authentication in p2p networks, where a data set originated at a trusted source is shared and dispersed over the nodes of a p2p network and is queried and retrieved by users through the API exported by the network. We focus our study on the basic `put-get` functionality that is realized by any distributed data structure built over overlay p2p networks, including the important class of *distributed hash tables* (DHTs) (e.g., [31, 36]). In particular, we are interested in guarding users against malfunctioning or malicious network nodes that incorrectly execute `get` and `put` operations.

* Research supported in part by IAM Technology, Inc., the National Science Foundation under grants IIS–0324846 and CCF–0311510, and the Institute for Information Infrastructure Protection (I3P) under an award from the Science and Technology Directorate at the Department of Homeland Security. Points of view or opinions in this document are those of the authors and do not represent the official position or policies of the U.S. Department of Homeland Security.

Current authentication techniques for data stored in p2p networks are static and centralized. Also, several authentication methods based on signatures on a per-object basis are vulnerable to *replay attacks*, where an old, out-of-date or invalid, data object is returned as the answer to a get operation.

In this paper, we introduce a new model for *distributed data authentication* in p2p networks and present an efficient realization of this model for securely performing dictionary operations on a p2p network. Our authentication structure and protocol are resilient against replay attacks and extend the functionality of p2p networks by supporting authenticated versions of operations put, get and remove, thus providing a transparent security layer to higher-level p2p applications.

By using only the basic operation of object location, our technique achieves generality and can be applied to a broad class of p2p architectures (e.g., existing DHT implementations). Our authentication scheme is based on the design of an efficient *distributed Merkle tree (DMT)*—the first distributed version of Merkle's authentication tree [23]. Thus, our construction can serve as a general-purpose authentication structure for decentralized computing architectures with minimal trust assumptions.

1.1 Motivation

Data storage and retrieval are essential tasks in p2p systems, where large data collections (e.g., documents, media files, database records) are shared over a network among participating peers. An important security problem in p2p system is data authentication in the presence of faulty or malicious network nodes. For instance, adversarial network nodes may wish to degrade the performance of a p2p storage system by providing false responses to queries.

We wish to ensure the integrity of shared data and to provide cryptographically sound techniques that allow a user to verify that retrieved data from the system is authentic and unaltered. Moreover, in a dynamic setting, where data evolve over time through updates, we want to also ensure that data items retrieved by queries have the most up-to-date versions. We consider the standard query model in p2p storage systems, where data items are stored as key-value pairs of the type (k, x) (keys are unique identifiers and values are associated with keys) and managed through operation put(k, x) (which inserts a new pair in the system) and query get(k) (which returns the value associated with key k). This is the core functionality exported by distributed data structures.

Assuming an established PKI, a straightforward approach to authentication is to individually sign each data item stored in the p2p system: when the data source wishes to add (k, x), it computes the signature σ of pair (k, x) using its private key and inserts $(k, (\sigma, x))$ into the data structure. A query for key k now returns the pair (σ, x), where σ allows the user to verify whether x is the valid answer. However, this "sign-all" approach is vulnerable to *replay attacks* for old values because it does not provide any mechanism for invalidating signatures on currently invalid pairs, such as pairs that have expired, were removed from the p2p system, or whose values have been modified. Therefore, in response to operation get(k), a malicious network node can return an invalid (old or out-of-date) value that is still verifiable. Note that most p2p systems do not support explicit item deletion; e.g., DHTs only keep a soft state, where data items expire after a time interval and are removed from the system (thus, to maintain these items, the source has to reinsert them). Nevertheless, even when some form of item deletion is supported, replay attacks are still possible: invalid signed pairs can simply be cached and never deleted. In general, we need a mechanism ensuring that only recent signatures are used to validate answers to queries and that deletions are correctly handled by the system.

Replay attacks can be prevented by introducing time-stamps in the signed values and a validity period for the signature, called *time quantum* [26]. However, this extension of the "sign-all" approach incurs a significant computational overhead: after each time quantum, each of all the valid pairs that currently reside in the system need to be retrieved, resigned by data source and then reinserted in the system. Thus, it is preferable to maintain at all times a *global authentication state* of the system that includes only the currently valid data items and essentially authenticates that data is properly updated. This is achieved by *signature amortization*, the state-of-the-art technique for dynamic data authentication, where a data source signs only one digest (short cryptographic description) of the entire collection of (valid) stored data items owned by this source. The canonical method for amortizing one signature over a large data set is Merkle's authentication tree [23]; however, there is currently no distributed implementation of this scheme. Existing p2p storage systems and DHT implementations that support an authentication service for the stored data are all using "sign-all" techniques. Thus, a replay attack is feasible for malicious network nodes; and if, instead, signature refreshing is used to solve the problem, this leads to inefficient and impractical authentication schemes.

1.2 Related Work

Merkle tree. The Merkle tree [23] is a simple and widely-used cryptographic construction for efficiently certifying set membership. The idea is to use a balanced tree and a cryptographic collision-resistant hash function (e.g., SHA-1) to produce a short cryptographic description of a large data set. Elements of the set are stored at the tree leaves and internal nodes store the result of applying the hash function to the concatenation of the values stored at the children nodes. The root value is signed and, when verified, the collision-resistant property propagates authentication from the root to the leaves. Certifying that an element is in the set is performed by using a *verification path* to recompute the authentic root value. This path consists (of the hash values) of the siblings of the nodes in the path from the leaf associated with the element to the root. Updates in the Merkle tree are handled with complexity proportional to the height of the tree (e.g., [26]). An extension to the symmetric-key setting is given in [12], where it is shown that verification along a path can be performed in parallel. No distributed implementation for Merkle trees currently exists.

Authenticated data structures. Authenticated data structures (ADSs) (see, e.g., [37, 11, 21, 26]) use a three-party model where data created by a trusted data source is replicated at several untrusted responders that answer query from users on behalf of the source. Signature amortization is used, similarly to the Merkle tree, but specially designed according to the supported query type. A significant amount of work has been done on developing efficient authenticated data structures for various type of queries (e.g., [6, 11, 21, 5, 1, 38]). The related model of *outsourced database* (ODB) systems studies the special case where SQL queries (essentially, range queries over indexes) are issued over databases published at remote sites (e.g., [18, 25, 24, 28]). Both models involve servers that keep a copy of the entire data set. Thus, they do not capture the architecture of p2p networks, where data is shared and distributed on a per-item basis.

Distributed hash tables. Distributed hash tables (DHTs) are a popular class of p2p storage networks that support the dictionary functionality (e.g., [36, 16, 33, 34, 20, 40, 29, 31]). Using distributed routing techniques over an overlay network, a DHT can locate the value associated with a query key with $O(\log n)$ expected communication steps, where n is the number of participating nodes, each maintaining $O(\log n)$ routing information. Based on DHTs, several practical distributed storage systems over p2p networks have been developed that support efficient retrieval (e.g., [7, 31, 4, 14, 32]). Other distributed data structures with similar efficiency provide more elaborate functionalities over p2p networks; for instance, skip-graphs [2] and their extensions (e.g., [13, 10]) support searching over ordered keys.

Trees over p2p overlay networks. The development of DHTs was followed by the design of various search and aggregation trees build over DHTs or other type of distributed trees (e.g., [15, 19, 30]). However, these trees can neither be used to implement a distributed Merkle tree nor meet the requirements for efficient data authentication over p2p networks, since these constructions are static or correspond to special-purpose search trees inappropriate to efficiently realize an authentication tree—which is sensitive to node losses or structural changes because of the use of the cryptographic hash function.

Security in p2p systems. Security issues related to p2p systems are discussed in [35, 39], where the authentication problem is treated simply using per-object signatures. Although with respect to routing and searching, numerous DHTs have been shown to tolerate significant network-node failures—random (e.g., [36, 16, 33, 31]) or malicious (e.g., [3, 8, 27])—data authentication has not been systematically studied in p2p networks. Existing p2p storage systems (e.g., [4, 7, 29, 31, 32]) support an elementary authentication service for retrieved data which is of the "sign-all" type, where stored contents are individually signed by their source. Often, authentication involves the so-called *self-certified data* [9], where large data items (e.g., a file system) get partitioned into blocks, which are stored as separate objects in the system and are bound together using collision-resistant hashing in some tree-like hierarchy, and where the root-block is signed. Although this technique resembles a Merkle tree, it only implements signature amortization among a large item and not among all data items owned by a source, which are still separately signed. Additionally, this authentication structure is static (no updates are supported) and, generally, unbalanced (e.g., parts of file systems can be flat and other can be extremely skewed). Overall, currently used authentication solutions are vulnerable to replay attacks (even if item removal is supported, as, e.g., in [32]) and lack efficiency for supporting signature refreshing and updates. Finally, privacy and anonymity issues or other security issues (e.g., the Sybil attack) related to p2p systems have been studied.

1.3 Paper Outline and Our Contributions

In Section 2, we introduce a new model for distributed data authentication over p2p networks, where management and retrieval of shared data resources are totally decentralized, yet *cryptographically verifiable* by the interested parties. In this model, data objects that are originated at a trusted source become available to users through an untrusted p2p storage network and authentication protocols guarantee the correct functionality of the underlying storage system. We present an efficient realization of this model for the basic dictionary operations performed on a dynamic set of data objects and describe data authentication protocols that allow users to verify the integrity of the data objects retrieved by the network and allow the source to verify the integrity of updates executed by the network.

The main idea behind our security solution is conceptually simple: we use an authentication tree to produce a cryptographic commitment of the stored data set, against which any update or query operation is checked for correctness. Implementing this idea in a dynamic distributed environment entails certain challenges. First, we design a balanced tree-like authentication structure that can be dispersed among network nodes and, at the same time, provide efficient retrieval of verification paths and allow efficient structural adjustments after updates. Additionally, we ensure that the commitment is updated correctly after any changes on the data set, even in the presence of malicious network nodes.

In Section 3, we present our main result, the first efficient scheme for implementing a fully dynamic *distributed Merkle tree* (DMT), using only the object-location functionality exported by any p2p system or DHT. Our scheme has certain properties that allow its efficient distribution over a p2p network and is designed to support locality for answer verification and facilitate the use of caching, thus achieving efficiency and resilience against malicious nodes. Moreover, balance is maintained at low cost over the course of updates on the tree. We analyze its performance and compare it with naive implementations. Our scheme, designed for both bottom-up and top-down access, constitutes a new, general-purpose, dynamic distributed tree.

In Section 4 and based on our DMT construction, we realize an efficient *authenticated distributed hash table* (ADHT), which extends DHTs in various ways. In particular, we show how our DMT can be extended to an authentication structure that provides authenticated and efficient versions of operations get, put, and also operation remove, supporting *authenticated deletions*, the first of this type. We compare ADHT with the "sign-all" solution. Our ADHT provides efficient distributed storage, secure against replay attacks and consistent with the update history. Our ADHT can in turn support a more general data authentication scheme for dictionary operations. In particular, we present the first efficient *distributed authenticated dictionary*. In a totally distributed setting over a p2p network with n nodes and using only the basic object-location operation, we show how to authenticate membership queries in a fully dynamic set of m data elements in $O(\log n \log m)$ time using $O(m \log m)$ storage, with similar complexities for supporting updates.

In Table 1, we summarize the comparison of our work with existing methods for distributed data authentication. Our scheme is the first to provide secure and efficient data authentication in totally decentralized environments over p2p networks. We conclude and discuss future work in Section 5.

Table 1. Qualitative comparison of our method with existing authentication models

	decentralized	replay-safe	efficient
"Sign-all" method in p2p networks	●		●
"Sign-all" method in p2p networks & timestamps	●	●	
ADS & ODB data authentication models		●	●
Our results	●	●	●

2 Authentication Model

We introduce a new model for distributed data authentication, where data items are stored, queried and authenticated in a totally decentralized fashion. This model captures fundamental security requirements that arise in p2p distributed storage systems. The model consists of:

– a trusted *data source* S, originator of a dynamic data set D;
– an untrusted distributed *p2p network* $\mathcal{N}$ that exports a specific functionality for storing, accessing and retrieving shared data resources; the nodes of $\mathcal{N}$ distributively store set D and answer queries about D on behalf of source S; D evolves over time through updates submitted by S to network $\mathcal{N}$; and
– *users* who issue queries about D by accessing any node of network $\mathcal{N}$.

In our model, network nodes are untrusted and can exhibit adversarial behavior in updates submitted by the source or queries posed by a user. A network node responsible for an update of D may maliciously fail to perform the update and cause the p2p system to become inconsistent with the sequence of updates issued by S (e.g., $\mathcal{N}$ attempts an "universe-split" attack, where two different states of set D are represented in the system). Also, a network node responsible for a query on D may maliciously falsify the returned answer (e.g., $\mathcal{N}$ attempts to compromise data integrity or launch a replay attack).

To guard the source and the users against attacks, our goal is to design an *authentication scheme*, a set of authentication structures and corresponding protocols that augment the functionality of network $\mathcal{N}$ with verification features.

In particular, the scheme allows the source to check that an update has been correctly performed by $\mathcal{N}$, according to the history of previously performed updates. The scheme also allows a user to verify that the answer to a query is authentic, as if it was coming directly from S.

An authentication scheme should be *secure*, informally meaning that the verification protocols reliably characterize the behavior of the network. Security, implies that when the p2p system is not under attack, then any update or query operation always passes the verification test (*completeness*); and, for any polynomial-time (on some security parameter) adversary that controls $\mathcal{N}$ and observes a chosen history of polynomial size of updates on D, succeeding in falsifying the verification test for an update or query is an event of negligible (on the security parameter) probability (*soundness*). Note that the above notion of security includes safety against replay attacks.

An authentication scheme should also be *decentralized*. Protocols and data-management procedures that are associated with data authentication should be distributed, designed as much as possible in accordance with the underlying p2p architecture. Or else, a security solution in a decentralized setting would be centralized, which would automatically diminish the advantages of the system.

An authentication scheme should finally be *efficient*, imposing low computational, communication and storage overhead to the parties participating in the protocols and network $\mathcal{N}$. Cost parameters that should be minimized are: the *storage cost*, the amount of authentication information stored at S and $\mathcal{N}$ or needed by a user to verify an answer; the *update* and *query costs*, the computational and communication costs incurred due to authentication at $\mathcal{N}$ after updates and queries on data set D; and the *verification cost*, the computational cost incurred by S or a user to verify the correctness of an update or query.

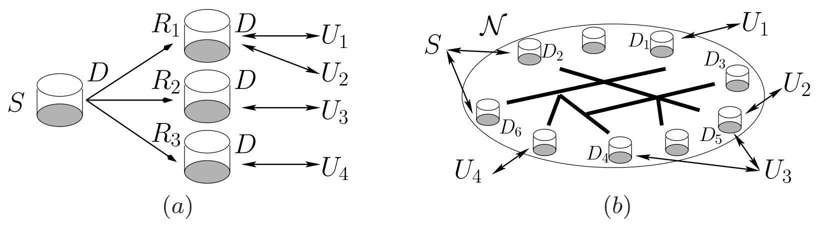


Fig. 1. (a) ADS model: each responder R_i stores set D on behalf of source S and answers users' queries. (b) Distributed authentication: D is dispersed as $D_1, D_2, \ldots$ over p2p network $\mathcal{N}$; updates and queries are performed by contacting any node of $\mathcal{N}$.

Our model drastically differs from those of authenticated data structures (ADS) (see Figure 1) and outsourced database systems (ODB) in that it is inherently decentralized. Data and authentication information are distributed over a p2p storage network at the data-item level and they are accessed by the source and the users through the interface of the network by contacting any of its nodes. In contrast, ADS model involves data replication at remote responder-servers, thus data distribution occurs only at the data-set level, and ODB model involves data outsourcing to a designated server, thus adopting centralized data management. Thus, we extend the client-server model of data authentication to a distributed authentication model that operates over any p2p network.

In developing an authentication scheme, it is desirable that as few as possible assumptions are made about network $\mathcal{N}$. Ideally, the underlying network should be any structured p2p network, for instance any DHT. By designing an authentication scheme using popular and well-studied distributed data structures, we leverage a broad class of existing p2p architectures, thus providing p2p systems with a transparent security layer at the application level. In this paper, we follow this principle and achieve generality by building our authentication scheme over the primitive search operation locate, which returns the network node corresponding to a given abstract object identifier. Since our constructions do not depend on the details of the p2p system implementation, we gain simplicity, extensibility and usability.

In what follows, we denote with $\mathcal{N}$ a DHT over which we wish to build an authentication scheme. Note that the scheme inherits the following properties shared by most DHT implementations: (1) a DHT with n network nodes uses $O(\log n)$ storage per node and performs a locate operation (also, put and get) in $O(\log n)$ network hops (node-to-node communication steps) with high probability; (2) node additions, deletions, and failures are handled dynamically through a distributed algorithm that incrementally updates the routing information; (3) some form of redundancy is used, which replicates data objects to a constant number of neighboring nodes so that node failures are tolerated also with respect to data recovery; and (4) caching techniques are used to improve data retrieval.

Finally, we consider all other types of misbehavior by network nodes (e.g., against routing or the DHT functionality) to be denial-of-service attacks, a distinct or orthogonal problem to data authentication. Of course, these attacks can also limit the functionality of the authentication scheme; but they will not compromise its security. Actually, since our authentication scheme is agnostic of the implementation of the underlying DHT, we can strengthen the resilience against malicious nodes in our model by using a specific DHT that tolerates certain DoS attacks (e.g, a DHT that authenticates routing information).

Our authentication scheme implements the signature amortization technique. In particular, we assume that the users of the system know and trust the public key of the source S. Using Merkle's authentication tree, S maintains at all times a digest of the data set. Queries are authenticated in the standard way: along with the answer, $\mathcal{N}$ returns the digest signed by S and a proof linking the answer to the digest. Updates are authenticated by using the digest to check that the current state of the data set is consistent with the update history.

3 An Efficient Distributed Merkle Tree

In this section, we present a distributed Merkle tree (DMT), an efficient implementation of an authentication tree built over a p2p network realizing a DHT. This is the core construction in our authentication scheme and a result of independent interest: since numerous security protocols and cryptographic constructions are based on Merkle's tree, a DMT yields distributed versions of such protocols and constructions. We first discuss our design goals.

Build on a data set D, the distributed authentication tree should be dynamic, allowing for efficient hash updates after changes in D. It should also be balanced, providing verification paths (membership proofs) of size that is logarithmic in $|D|$, and its height balance should be efficiently maintainable after updates. In order to optimize its verification properties and further improve its usability, we are particularly interested in facilitating the construction (location) of the verification paths of the DMT; that is, the verification path of any data item should be retrieved by the network as fast as possible. All the above should be implemented in a distributed way, using only the locate operation, which is provided by the network and takes $O(\log n)$ time for a network of n nodes. Accordingly, cost parameters we wish to minimize are: the *path location cost*, the cost for constructing a verification path (proof), the *update cost*, the cost for maintaining the authentication structure after updates on D, and the *storage cost*. Both the location and update costs each consists of (1) processing cost, i.e., computational cost for the participating nodes in the system, and (2) communication cost, i.e., cost of locate operations or direct communications between nodes.

To gain some intuition behind our construction, assume that set $\{x_1, \ldots, x_n\}$ is distributed over the network (each element stored at a unique network node) and consider a Merkle tree computed over this set. All hash values in the tree need be distributed in the network. The first problem to consider is how the hash values are indexed, i.e., with which keys they are stored in the system. The hash value is a value that is unknown to network nodes, thus the value itself cannot be used as a key. Additionally, for immediate location of verification paths, network node storing element x_i should also store information about the verification path that corresponds to x_i. We briefly describe two less efficient approaches for realizing a DMT. One solution is to replicate the tree structure to all involved network nodes and use unique identifiers for storing hash values in the DHT. The cost to construct a verification path is $O(\log n)$ locate operations, however, the cost to maintain the tree after structural updates is high, $O(n \log n)$, and also the $O(n^2)$ total storage is prohibitive. Alternatively, a network node can store the entire verification path (hashes) of the element it stores. The path location cost is $O(1)$, however any update on the data set now incurs $O(n \log n)$ cost; the total storage cost is $O(n \log n)$. Our approach is to distribute information about routes in the network that construct verification paths, essentially combining the above ideas: using unique identifiers, hash values are stored in the network and the network node storing an element also stores the identifiers of the corresponding verification path. For efficiency in the dynamic case, we use a weight-balanced ($BB[\alpha]$) hash tree. Details about the construction and its analysis follow.

3.1 Our Construction

We consider the more general case, where an authentication structure over m data items $\{x_1, x_2, \ldots, x_m\}$, owned by the same source and stored in a p2p network of size $n \geq m$ that realizes a distributed hash table. We design our distributed Merkle tree using the primitive locate operation over the network. Without loss of generality, we assume that objects are stored at distinct network nodes. Our results generalize to the cases where more than one data item are stored at nodes and also where more than one sources produce these items.

Our scheme is described as follows (see Figure 2). For convenience, tree nodes are denoted by lower-case letters and network nodes by capital letters.

Let T be a balanced binary tree built over the elements of (dynamic) data set $\{x_1, x_2, \ldots, x_m\}$, with one-to-one correspondence between leaves and elements, and let h be a cryptographic hash function. Each tree node u in T has a unique identified id_u (drawn efficiently from some space). Conventionally, the identifier of a leaf is set to be the corresponding element. Tree T is used as a hashing structure in the standard way: each non-leaf tree node u with children nodes v_1 and v_2 in T is associated with (or stores, conceptually) hash value $L(u)$, which equals to $h(L(v_1)\|L(v_2))$, i.e., the hash of the hash values that v_1, v_2 are associated with, and each leaf w_i stores the hash value $L(w_i) = h(x_i)$ of the corresponding element x_i. We augment the hashing structure as follows: we require that each internal tree node also stores the hash values of its children.

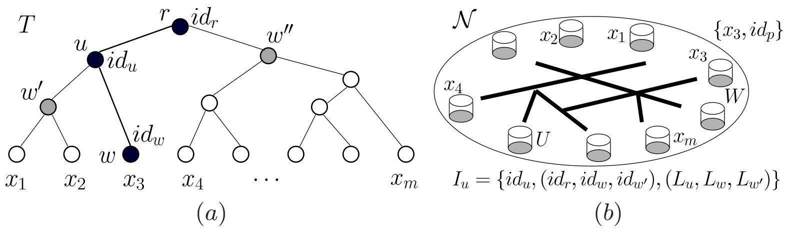


Fig. 2. (a) Hash tree T over elements $x_1, \ldots, x_m$: node u, with identifier id_u storing hash L_u, is mapped to network node U and leaf w, storing x_3 with verification path p, to network node W. (b) Distribution of T over the network: U stores information I_u related to u; W stores $\{x_3, id_p\}$ and structural/balancing information of p.

Next, we use the tree identifiers for distributing tree T into the nodes of the underlying p2p network. Each non-leaf tree node u is mapped to a network node U through a function f and according to id_u, i.e., $U = f(id_u)$. Node U stores some information I_u related to u: the (three) hash values associated with node u, the tree identifiers of the parent tree node and the children of u and local structural information about node u. Moreover, a leaf node w_i, corresponding to element x_i, is also mapped to a network node $W_i = f(x_i)$ through function f.[1] Along with x_i, node W_i stores some information I_{x_i} related

[1] We impose no restrictions on $f(\cdot)$; in general, it is a known function used by the underlying p2p network for mapping objects to network nodes (many DHTs use the SHA-1 function), implemented by the locate operation supported by the network.

to the (verification) path p_i in T from w_i to the root r of T; in particular, this information includes:

- the ids of the tree nodes of path p, denoted as id_p;
- structural and balancing information of tree nodes in p; for each node u in p with children v_1 and v_2, W_i stores: (1) whether v_1 or v_2 belongs in p; (2) the balancing information of node u, which is a pair (b_1, b_2), expressing balancing information related to the subtrees defined by v_1 and v_2 respectively.

The above scheme distributes tree nodes and verification paths over a p2p network and correctly implements a DMT. Our construction is designed mainly for bottom-up tree traversal, which is appropriate for most security-related and cryptographic applications, although it can be easily extended to support also top-down traversal, similar to search tree (see Section 4). Accordingly, our tree is accessed very efficiently: given a data element x_i, the corresponding verification path is distributively retrieved using $O(\log m)$ locate operations for mapping identifiers in id_p to network nodes. Using route distribution, that is, maintaining the invariance that each network node knows the route for its verification path, we can actually achieve extra efficiency, as we discuss at the end of the section.

Finally, we choose our tree T to be a weight-balanced tree and, in particular, a $BB[\alpha]$ tree [22] (weight-balanced trees with important balancing properties). This choice is related to efficiency in maintaining the structure of the hash tree after updates in the data set (element insertions or deletions). We consider two types of updates along a verification path: hash updates (due to rehashing) and structural updates (due to re-balancing changes, i.e., rotations). Route distribution supports efficient hash updates, since $O(\log m)$ locate operations suffice in updating the hash values of a path id_p. Similarly, structural updates can be executed by successively contacting the network nodes corresponding to id_p and performing any necessary local update at node U using information I_u, however, the incur an additional cost. Any rotation at level k of T triggers an extra update cost for publishing the new routes to the $O(2^k)$ in total involved network nodes. Using a $BB[\alpha]$ tree, where α is a balancing parameter bounding the ratio of the weights (i.e., size of corresponding subtree) of neighboring tree nodes, we achieve that on average $O(1)$ rotations occur after an update and they occur more often at nodes closer to leaves than at nodes higher in T. This gives on average a very good performance in an amortized sense, since expensive reconstructions happen rarely. The following lemma formalizes the above argument.

Lemma 1. *For any series of m update operations on an initially empty set, the DMT based on a $BB[\alpha]$ hash tree T, with $\alpha \in (\frac{1}{4}, 1 - \frac{\sqrt{2}}{2})$, has $O(\log n \log m)$ amortized structural update cost. Moreover, during these operations, structural updates at level k of T with cost $O(2^k)$ occur with frequency $O(\frac{1}{2^k})$.*

Proof. (Sketch.) The proof is based on the update technique in our scheme and the properties of $BB[\alpha]$ trees (e.g., see analysis in [22]). Consider any update in the data set that results in a structural update in the tree T, in particular, along the verification path p_i of element x_i, and assume that the corresponding network node W_i storing x_i has been located. Node W_i can initiate a structural update along p_i by examining p_i in bottom-up fashion and contacting the appropriate network nodes. If p_i is structurally updated to p'_i and a rotation occurs at node u in T, let T_u be the subtree in T defined by u. Each network node corresponding to leaf node x_i in T_u or a neighboring node v of u in T_u must respectively update its local information I_{x_i} and I_v. In total, $O(|T_u|)$ network nodes must be notified about the updates in T. This is possible by accessing T_u in a top-down fashion (starting at u), contacting the corresponding network nodes and communicating the necessary updates, a process completed after $O(|T_u|)$ locate operations using $O(|T_u| \times \log n)$ communication. Overall, the structural update cost is $O(|T_u|)$ locate operations. By the properties of $BB[\alpha]$ trees with parameter α in the appropriate range, we have that the total cost for updating all verification paths in the DHT, for a sequence of t update operations (insertions or deletions) on an initially empty set, is $O(t \log t)$. Thus, for the same series of update operations, the total structural update cost is $O(\log n \times t \times \log t)$ (time and communication). For $t = O(m)$, we get that the amortized overall structural update cost is $O(\log n \log m)$ over a sequence of operations of size linear on m. Using the additional property shown in [22], namely that costly rotations at levels close to the root occur rarely with frequency inversely proportional to the corresponding subtree size, the proof is completed. □

The following theorem summarizes the efficiency of our DMT and our main result. A p2p network with n nodes is called *efficient* if location operations take time $O(\log n)$.

Theorem 1. *There exists a scheme for implementing a distributed Merkle tree T on a data set of size m over a peer-to-peer network $\mathcal{N}$ with n nodes ($m \leq n$) with the following properties:*

1. *Tree T uses space $O(m \log m)$, distributed over $O(m)$ network nodes, and incurs $O(\log m)$ storage overhead per network node.*
2. *A verification path of T has size $O(\log m)$ and can be accessed with $O(\log m)$ locate operations on $\mathcal{N}$; thus, if $\mathcal{N}$ is efficient, the expected computational and communication cost for accessing a verification path is $O(\log n \log m)$.*
3. *A hash update on tree T involves $O(\log m)$ location operations; thus, for an efficient network $\mathcal{N}$, the expected computational and communication cost of a hash update is $O(\log n \log m)$.*
4. *A structural update on tree T involves $O(m \log m)$ location operations, amortized over a series of $O(m)$ structural updates on an initially empty tree; thus, for an efficient network $\mathcal{N}$, the expected amortized computational and communication cost of a structural update is $O(\log n \log m)$.*

Proof. (Sketch.) Tree T is balanced, thus verification paths have $O(\log m)$ size. The storage complexity is $O(m \log m)$, since internal tree nodes require $O(1)$ storage and leaves $O(\log m)$ storage. Through route distribution, constructing any verification path requires only $O(\log m)$ locate operations, performed by the initiating node according to p_i. Similarly, hash updates are performed by bottom-up traversal of nodes in p_i. Structural updates are performed as in Lemma 1. □

Note that above, storage is optimal for route distribution and in accordance with storage requirements for an efficient network, where each network node stores $O(\log n)$ routing information; thus, our DMT does not asymptotically increase the storage requirements of the underlying network.

Improvement through caching. We discuss a simple extension that under a reasonable assumption, can improve the costs of for path location and update. Assuming that network-node failures occur less often than queries and updates on the DMT, we can improve the efficiency of our scheme as follows. The goal is to transform the multiplicative $O(\log n)$ factor (introduced due to locate operations for retrieving or updating hash values) into an additive term in the complexity of our scheme. This is achieved by caching network node identifiers: the idea is to have each network node corresponding to a leaf of T to cache in its memory the identifiers of the $O(\log m)$ network nodes that store the hash values of its corresponding verification path. That is, once such a network node is first contacted, its identifier is remembered. Since network nodes can fail or go down, it is possible that cached nodes are no longer nodes of the network. In this case, we have a cache miss which will trigger a location operation. Although we can still use some techniques to avoid this overhead (e.g., by caching neighboring nodes storing the same information due to redundancy), we observe that when the rate of network node failures is sufficiently small then we can actually amortize the $O(\log n)$ factor due to occasional location operations (cache misses) in the cost for operating on the tree. In particular, if network nodes fail independently with probability $O(\frac{1}{\log m})$ during the time interval of a tree traversal, then the expected number of network node failures that occur during a path location or update is $O(1)$. Thus, using caching the expected complexity for path location and updates on the tree is $O(\log n + \log m)$.

Table 2 summarizes the comparison between the various schemes for implementing a DMT. We see that our scheme provides an very efficient solution that, using caching and under reasonable assumptions, can be asymptotically optimal in an amortized sense. Finally, we note that, when $m > n$, we can appropriate extend our scheme by having each network node maintaining an additional data structure (for locating the stored elements). Our scheme supports authentication of data collections of one data source; we can support multiple data sources simply by using multiple instantiations of our DMT.

Table 2. Efficiency comparison of various schemes for realizing a DMT for a data set of size m over a p2p network of size n. Expected complexity is denoted with * and amortized expected complexity is denoted with **.

	storage	path location	hash update	structural update
tree replication	$O(m^2)$	$O(\log n \ \log m)^*$	$O(\log n \log m)^*$	$O(m \log n)^*$
path replication	$O(m \log m)$	$O(1)$	$O(m \log n)^*$	$O(m \log n)^*$
route distribution	$O(m \log m)$	$O(\log n \log m)^*$	$O(\log n \log m)^*$	$O(\log n \log m)^{**}$
w/ caching	$O(m \log m)$	$O(\log n)^*$	$O(\log n)^*$	$O(\log n)^{**}$

4 An Efficient Authenticated Distributed Hash Table

In this section, we design an authentication scheme for membership operations on dynamic sets in our authentication model. First, we use our DMT construction to realize authentication protocols for verifying the basic operations of any DHT and we design an efficient *authenticated distributed hash table* (*ADHT*).

Consider a source S that produces m data objects as key-value pairs, which are stored in a DHT that supports the basic put-get operations. We design protocols for augmenting this functionality to a new p2p storage system that provides better information assurance, supporting the following authenticated operations:

- auth_put: a key-value pair is inserted in the system by source S in an authenticated way, so that S verifies the correctness of the insertion;
- auth_get: the value of an existing in the system key is retrieved by a user in an authenticated way, so that the user verifies the authenticity of the value;
- auth_remove: an existing key-value pair is removed from the system by source S in an authenticated way, so that S verifies the correctness of the removal.

Implementation. To realize the above functionality of an ADHT, we use the technique of signature amortization over the data set that exists in the system. The main idea is use the DMT of the previous section as a distributed authentication structure for implementing the following invariant: at all times, source S maintains (by storing locally) a cryptographic digest of the currently correct (up-to-date) data set. Query verification is achieved by having S signing the digest along with a fresh timestamp and storing this information in the system. Update verification is achieved by having the source computing the new digest that correctly corresponds to the new data set after the update, using authentication information that is first verified against the current digest.

The digest is defined as the root hash value of the tree T that is build over the data set and that corresponds to the DMT maintained in the p2p system. We augment the construction of T as follows. First, we add an additional level of hashing in T: the leaf node corresponding to pair (k, x) now stores hash value $h(h(k)\|h(x))$. Moreover, each non-leaf tree node u with children nodes v_1 and v_2 in T stores a hash value $L(u)$ that also encodes the structural and balancing information of u, that is, $L(u) = h(L(v_1)\|L(v_2)\|h(s_u))$, where s_u encodes whether u is a left or right child and its balancing information.[2] Additionally, we augment T to also serve as a *search tree*[3]: key-value pairs are sorted according to their keys (we assume they are drawn from a totally order set) and are stored at the leaves matching their left-to-right ordering; also, each non-leaf node u stores a corresponding search key, e.g., the maximum key stored at the leaves of T_u. We assume that, using bootstrapping techniques, both S and the users have access (through direct connection) to an active node of the p2p network and that there exists a publicly known function for creating identifiers for the new nodes in T.

We next describe the exact protocols for the authenticated operations.

[2] For $BB[\alpha]$ trees, this is the ratio $|T_{v_1}|/|T_u|$, assuming that v_1 is the left child of u.

[3] Not needed, if the DHT supports searches for both exact and near matches [2, 13, 10].

Queries. Queries are performed by any user by first contacting a network node and issuing a get request on a key. For queries, the verification path of a leaf is simply the corresponding leaf-to-root path. A path retrieval query is executed by the system over the DMT and what is returned to the user is: (1) the corresponding value, (2) the verification path (collection of hash values and relative information for computing the root hash) and (3) the signed digest. The user accepts the answer if and only if (*i*) hashing over the value and the verification path results in a hash value that equals the root hash (digest) and (*ii*) the signature on the digest is valid and contains a fresh timestamp.

Updates. Updates are performed by the source S by first contacting a network node and issuing an update request. For updates, the verification path of a leaf is augmented to also include the sibling nodes of the nodes in leaf-to-root path. The system then reports to S the verification path p_ℓ of the leaf ℓ of the DMT T that is related to the update, i.e., the sibling leaf of the new leaf (put operation) or the leaf to be deleted (remove operation). For put operations, ℓ can be located through top-down traversal of T; we assume that S stores, and updates when needed, the identifier of the root. Path p_ℓ contains all the necessary information for computing the digest, given the information stored at ℓ (note that, in the case of a put operation, it contains the information stored at the sibling of ℓ). In particular, p_ℓ contains all the structural and balancing information that is needed for any hash or structural update on it. This information serves as a *consistency proof* for the source S and is used to compute the new digest in three steps. First, the verification path is checked to be authentic by hashing along the path and recomputing the current digest; this also verifies that the structural and balancing information is also authentic, consistent with the current digest stored by S. In the check fails, the protocol rejects: the system failed to correctly execute the previous update. Otherwise, S locally executes the tree update by operating on path p_ℓ; this is feasible because both hash updates and structural tree adjustments only happen along this path in a bottom-up fashion. Finally, S hashes over the new tree path and computes and stores the new digest. Once the new digest is computed, S timestamps and signs it and returns the signed copy for storage in the system. Then a regular hash-tree update is performed by the system to execute the put or remove operation. By this interaction, S needs only to keep $O(1)$ authentication information, the current signed digest. Asymptotically, no additional computational or communication cost is introduced by this extra interaction between the system and the source.

Security. The security of our protocols can be proved with using standard reductions to the security of the cryptographic primitives that are used in our authentication scheme, under standard hardness assumptions. By using a family of collision-resistant hash functions and a signature scheme secure against adaptive chosen-message attacks, we have that the authentication scheme in ADHT is secure: any successful attack by the network against the security of our scheme corresponds to either a forged signature or a hash collision.

Before stating the main result of this section, we recall that a p2p network with n nodes is called efficient if location operations take time $O(\log n)$.

Theorem 2. *There exists an authenticated distributed hash table over an efficient peer-to-peer network with n nodes that supports authenticated operations* auth_put, auth_get *and* auth_remove *on a data set of size $m \leq n$ and has the following properties:*

1. *The distributed authentication scheme is secure.*
2. *The storage at the source is $O(1)$; the storage at the network is $O(m \log m)$.*
3. *The query cost is $O(\log m)$, that is, $O(\log m)$ locate operations; or, equivalently, the expected time and communication complexity to answer a query is $O(\log n \log m)$.*
4. *The amortized update cost is $O(\log m)$, that is, $O(\log m)$ locate operations; or, equivalently, the amortized expected time and communication complexity of an update is $O(\log n \log m)$.*

Table 3 summarizes the comparison of our ADHT and its extension ADHT-c using caching with the existing data authentication schemes for dynamic content in p2p storage networks. Existing authentication methods support data integrity by separately signing all data items stored by the same source, but they are either vulnerable to replay attacks ("Sign-all"), since old items can still be incorrectly verified, or induce a significant update cost when timestamping is used to eliminate replay attacks ("Sign-all"-t), since all data items should be resigned after any update or refreshed at regular time intervals. Using a logarithmic space overhead per network node, ADHT provides a secure, efficient and distributed new authentication scheme for verifying basic operations over p2p storage systems and offers a transparent security layer that is independent of the exact implementation of the system.

Table 3. Comparison of ADHT with "sign-all" schemes for authenticating queries on data set of size m over a network of size n, $m \leq n$. Expected complexity is denoted with * and amortized expected complexity is denoted with **.

	storage	signing cost	query cost	update cost	replay-safe
"Sign-all"	$O(m)$	$O(m)$	$O(\log n)^*$	$O(\log n)^*$	no
"Sign-all"-t	$O(m)$	$O(m)$	$O(\log n)^*$	$O(m \log n)^*$	yes
ADHT	$O(m \log m)$	$O(1)$	$O(\log n \log m)^*$	$O(\log n \log m)^{**}$	yes
ADHT-c	$O(m \log m)$	$O(1)$	$O(\log n + \log m)^*$	$O(\log n + \log m)^{**}$	yes

Distributed Authenticated Dictionary. An immediate application of our ADHT is a *distributed authenticated dictionary*, where membership queries on a dynamic data set of key-value pairs are authenticated. Suppose that keys are drawn from a totally ordered universe. Our DMT is built on top of the data elements sorted according to their keys. To support authentication of negative answers, we use the technique of [17]: pairs of keys that are consecutive in the ordering used in the Merkle tree are inserted in the system for proving non-membership in the set. E.g., if (k, v), (k', v') are members of the set and k' is the immediate successor key of k, then value v contains also key k'. The DMT is again used also as a search tree. This scheme has asymptotically the same performance as the ADHT described above, given by Theorem 2.

Load-balance issues. Although our authentication structure achieves load balance with respect to data distribution over the p2p network (given the properties of the underlying DHT), as described, it does not achieve load balance with respect to network access. For instance, network nodes that store the tree root are accessed much more often than other network nodes. This turns out to be an important issue that appears to hold in general: all existing techniques for achieving authentication over DHTs that use signature amortization, including our technique or techniques based on self-certified data, introduce congestion at certain network nodes. The problem is challenging, since load-balancing and efficient content authentication in p2p systems correspond to contradictory design goals: signature amortization introduces heavily accessed points in the system, whereas for load-balancing we wish network nodes to be accessed with uniform, rather than skewed, distribution. However, we propose the following simple solution for load-balance in an amortized sense: after any structural update at node u in the tree, we choose new tree identifiers for all nodes in the corresponding subtree T_u. Asymptotically no extra cost is incurred, since the structural update already propagates over T_u, thus the system can afford redistributing T_u to new network nodes. Identifiers are chosen according to a random, well-defined and unpredictable way, such that no significant communication overhead is introduced in the structure. Effectively, over time, we expect to achieve smoother (closer to uniform) access patterns for network nodes.

5 Conclusions and Future Work

We consider the problem of data authentication in p2p storage networks. We introduce a new model for authenticating data in decentralized computing environments that extends the model of authenticated data structures and better captures the security needs of existing distributed systems. We design the first efficient implementation of a distributed Merkle tree (DMT) and show how it can be applied to the design of an authenticated distributed hash table that supports efficient verification of membership and update operations over dynamic data sets and eliminates the threat of replay attacks.

As future work, we plan to implement the DMT construction and experimentally test its efficiency and study load-balance, concurrency and other performance issues. We leave as open problems the design of authenticated schemes for more general queries, beyond set-membership, and the study of additional security issues in this new model, such as DoS attacks and Byzantine behavior.

Acknowledgments

We would like to thank David Croston, Michael Goodrich, John Jannotti and John Nuber for useful discussions on the subject of this paper.

References

[1] A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia. Persistent authenticated dictionaries and their applications. In *Proc. ISC*, pages 379–393, 2001.

[2] J. Aspnes and G. Shah. Skip graphs. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, 2003.

[3] M. Castro, P. Druschel, A. J. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured p2p overlay networks. In *Proc. OSDI*, pages 299–314, 2002.

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. SOSP*, pages 202–215, 2001.

[5] P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. Stubblebine. Flexible authentication of XML documents. In *Proc. CCS*, pages 136–145, 2001.

[6] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic data publication over the Internet. *Journal of Computer Security*, 11(3):291–314, 2003.

[7] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Proc. HOTOS*, page 75, 2001.

[8] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *Proc. of Symposium on Discrete Algorithms*, pages 1–23, 2002.

[9] K. Fu, M. F. Kaashoek, and D. Mazieres. Fast and secure distributed read-only file system. *Computer Systems*, 20(1):1–24, 2002.

[10] M. T. Goodrich, M. J. Nelson, and J. Z. Sun. The rainbow skip graph: a fault-tolerant constant-degree distributed data structure. In *Proc. SODA*, pages 384–393, 2006.

[11] M. T. Goodrich, R. Tamassia, N. Triandopoulos, and R. Cohen. Authenticated data structures for graph and geometric searching. In *Proc. of RSA Conference*, pages 295–313, 2003.

[12] E. Hall and C. S. Julta. Parallelizable authentication trees. In Cryptology ePrint Archive, December 2002.

[13] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. SkipNet: A scalable overlay network with practical locality properties. In *Proc. USENIX Symp. on Internet Technologies and Systems*, 2003.

[14] R. Huebsch, B. Chun, J. Hellerstein, B. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. Yumerefendi. The architecture of PIER: an internet-scale query processor. In *Proc. of CIDR*, pages 28–43, 2005.

[15] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. Baton: a balanced tree structure for peer-to-peer networks. In *Proc. VLDB*, pages 661–672, 2005.

[16] F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Proc. of 2nd International Workshop on Peer-to-Peer Systems*, 2003.

[17] P. C. Kocher. On certificate revocation and validation. In *Proc. of International Conference on Financial Cryptography*, pages 172–177, 1998.

[18] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In *Proc. of ACM SIGMOD*, pages 121–132, 2006.

[19] J. Li, K. Sollins, and D.-Y. Lim. Implementing aggregation and broadcast over distributed hash tables. *ACM SIGCOMM Computer Communication Review*, 35(1):81–92, 2005.

[20] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proc. USENIX Symp. on Internet Technologies and Systems*, pages 127–140, 2003.

[21] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1):21–41, 2004.

[22] K. Mehlhorn. *Data Structures and Algorithms 1: Sorting and Searching*, volume 1 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Heidelberg, Germany, 1984.

[23] R. C. Merkle. A certified digital signature. In *Proc. CRYPTO*, pages 218–238, 1989.

[24] G. Miklau and D. Suciu. Implementing a tamper-evident database system. In *Proc. Asian Computing Science Conference, ASIAN*, pages 28–48, 2005.

[25] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *Proc. of Network and Distr. System Security*, 2004.

[26] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proc. of USENIX Security Symposium*, pages 217–228, 1998.

[27] M. Naor and U. Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *Proc. SPAA*, pages 50–59, 2002.

[28] G. Nuckolls. Verified query results from hybrid authentication trees. In *Proc. of Database Security*, pages 84–98, 2005.

[29] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. SPAA*, pages 311–320, 1997.

[30] S. Ramabhadran, J. Hellerstein, S. Ratnasamy, and S. Shenker. Prefix hash tree - an indexing data structure over distributed hash tables. In *Proc. of ACM PODC*, pages 368–368, 2004.

[31] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. SIGCOMM*, pages 161–172, 2001.

[32] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A public DHT service and its uses. In *Proc. SIGCOMM*, pages 73–84, 2005.

[33] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *LNCS*, 2218:329, 2001.

[34] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.

[35] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proc. of International Workshop on P2P Systems*, pages 261–269, 2002.

[36] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proc. SIGCOMM*, pages 149–160, 2001.

[37] R. Tamassia. Authenticated data structures. In *Proc. of European Symposium on Algorithms*, pages 2–5, 2003.

[38] R. Tamassia and N. Triandopoulos. Computational bounds on hierarchical data processing with applications to information security. In *Proc. ICALP*, pages 153–165, 2005.

[39] D. S. Wallach. A survey of peer-to-peer security issues. In *Proc. of International Symposium on Software Security*, 2002.

[40] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, Apr. 2001.

An Identity-Based Signcryption Scheme for Multi-domain Ad Hoc Networks

Fagen Li[1,2], Yupu Hu[1], and Chuanrong Zhang[1,3]

[1] Key Laboratory of Computer Networks and Information Security,
Xidian University, Xi'an 710071, China
[2] School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu 610054, China
[3] Telecommunication Engineering Institute,
Air Force Engineering University, Xi'an 710077, China
`fagenli@mail.xidian.edu.cn`

Abstract. As various applications of wireless ad hoc networks have been proposed, security has become one of the big research challenges and is receiving increasing attention. Recently, Several security schemes for wireless ad hoc networks have been proposed using identity-based signcryption schemes. However, almost all identity-based signcryption schemes that have been proposed until now are based on a single private key generator, which is not suitable for multi-domain ad hoc networks. In this paper, we propose a new identity-based signcryption scheme based on multiple private key generators, which is more suitable for multi-domain ad hoc networks. We prove its semantical security under the Decisional Bilinear Diffie-Hellman assumption in the random oracle model.

Keywords: Ad hoc networks, identity-based signcryption, bilinear pairings, provable security.

1 Introduction

An ad hoc network is a collection of autonomous nodes that communicate with each other by forming a multi-hop wireless network. The property of not relying on the support from any fixed infrastructure makes it useful for a wide range of applications, such as instant consultation between mobile users in the battlefields, emergency, and disaster situations, where geographical or terrestrial constraints demand totally distributed networks. While ad hoc network provides a great flexibility for establishing communications, it also brings a lot of research challenges. One of the important issues is the security due to all the characteristics of these networks, such as the vulnerability of the wireless links, the limited physical protection of each node and the dynamically changing topology. Recently, Several security schemes for ad hoc networks have been proposed using identity-based (ID-based) signcryption schemes, such as key management scheme [18], authenticated broadcasting scheme [5], and routing protocols

TIDS [11], ISSRP [23] and ISMANET [24]. The using of ID-based signcryption has the following advantages:

1. There is no need to authenticate a public key because of using the ID-based cryptography.
2. Confidentiality, integrity, non-repudiation and authentication are provided simultaneously because of using the signcryption technique.
3. Computational costs and communication overheads can be reduced.

However, almost all ID-based signcryption schemes that have been proposed until now are based on a single private key generator (PKG), which is not suitable for multi-domain ad hoc networks [15] formed by a consortium of different organizations. It is unrealistic to assume that different organizations use a single PKG. Therefore, it is necessary to find an ID-based signcryption scheme based on multiple PKGs.

1.1 Related Work

ID-based cryptography was introduced by Shamir in 1984 [26]. The distinguishing property of ID-based cryptography is that a user's public key can be any binary string, such as an email address that can identify the user. This removes the need for senders to look up the recipient's public key before sending out an encrypted message. Usually, private keys of users' are issued by a trusted authority called the PKG. ID-based cryptography is supposed to provide a more convenient alternative to conventional public key infrastructure. In 2001, Boneh and Franklin [6] proposed the first practical ID-based encryption scheme using pairings on elliptic curves. Since then, most researches on ID-based cryptography are based on this system.

Confidentiality, integrity, non-repudiation and authentication are the important requirements for many cryptographic applications. A traditional approach to achieve these requirements is to sign-then-encrypt the message. Signcryption, first proposed by Zheng in 1997 [28], is a cryptographic primitive that performs digital signature and public key encryption simultaneously, at lower computational costs and communication overheads than the signature-then-encryption approach. Several efficient signcryption schemes have been proposed since 1997 [3,29,12,25,22,14,27,21] and a first example of formal security proof in a formal security model was published in 2002 [2]. However, until 2002, none of these schemes were ID-based. Malone-Lee [20] proposed a first method to achieve an ID-based signcryption solution. Libert and Quisquater [19] pointed out that Malone-Lee's scheme [20] is not semantically secure because the signature of the message is visible in the signcrypted message. Chow et al. [10] designed an ID-based signcryption scheme that provides both public verifiability and forward security. Boyen [7] presented an ID-based signcryption scheme that provides not only public verifiability and forward security but also ciphertext unlinkability and anonymity. In [9], Chen and Malone-Lee improved Boyen's scheme [7] in efficiency. In [4], Barreto et al. constructed the most efficient ID-based signcryption scheme to date.

1.2 Our Contribution

In this paper, we present an ID-based signcryption scheme based on multiple PKGs. We prove its semantical security under the Decisional Bilinear Diffie-Hellman assumption in the random oracle model. We believe that our scheme is more suitable for multi-domain ad hoc networks than previously proposed schemes.

1.3 Organization

The rest of this paper is organized as follows. Some preliminary works are given in Section 2. The formal model of ID-based signcryption is described in Section 3. The proposed ID-based signcryption scheme is given in Section 4. We analyze the proposed scheme in Section 5. Finally, the conclusions are given in Section 6.

2 Preliminaries

In this section, we briefly describe the basic definition and properties of the bilinear pairings.

Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. A bilinear pairing is a map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with the following properties:

1. Bilinearity: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$, $a, b \in Z_q$.
2. Non-degeneracy: There exists P and $Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$.
3. Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all P,$Q \in G_1$.

The modified Weil pairing and the Tate pairing [6] are admissible maps of this kind. The security of our scheme described here relies on the hardness of the following problems.

Definition 1. *Given two groups G_1 and G_2 of the same prime order q, a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ and a generator P of G_1, the Decisional Bilinear Diffie-Hellman problem (DBDHP) in $(G_1, G_2, \hat{e})$ is to decide whether $h = \hat{e}(P, P)^{abc}$ given (P, aP, bP, cP) and an element $h \in G_2$. We define the advantage of a distinguisher against the DBDHP like this*

$$\begin{aligned} Adv(D) = |&P_{a,b,c,\in_R Z_q, h\in_R G_2}[1 \leftarrow D(aP, bP, cP, h)] \\ &-P_{a,b,c,\in_R Z_q}[1 \leftarrow D(aP, bP, cP, \hat{e}(P, P)^{abc})]|. \end{aligned}$$

Definition 2. *Given two groups G_1 and G_2 of the same prime order q, a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ and a generator P of G_1, the Computational Bilinear Diffie-Hellman problem (CBDHP) in $(G_1, G_2, \hat{e})$ is to compute $h = \hat{e}(P, P)^{abc}$ given (P, aP, bP, cP).*

The decisional problem is of course not harder than the computational one. However, no algorithm is known to be able to solve any of them so far.

3 Formal Model of ID-Based Signcryption

3.1 Generic Scheme

A generic ID-based signcryption scheme based on multiple PKGs consists of the following five algorithms. We suppose that there are two trusted authorities, say PKG_1 and PKG_2.

Setup1: Given a security parameter k, this algorithm generates the system's public parameters *params*.

Setup2: Given the system's public parameters *params*, the PKG_1 generates a master secret key s_1 and a corresponding public key P_{pub}^1. Similarly, The PKG_2 generates a master secret key s_2 and a corresponding public key P_{pub}^2.

Extract: Given an identity ID in $\text{PKG}_l (l = 1, 2)$, the PKG_l computes the corresponding private key S_{ID} and transmits it to its owner in a secure way.

Signcrypt: To send a message m to Bob, Alice obtains the ciphertext σ by computing $\mathbf{Signcrypt}(m, S_{ID_A}, ID_B)$.

Unsigncrypt: When Bob receives σ, he computes $\mathbf{Unsigncrypt}(\sigma, ID_A, S_{ID_B})$ and obtains the plaintext m or the symbol $\perp$ if σ is an invalid ciphertext between identities ID_A and ID_B.

For consistency, we of course require that if $\sigma = \mathbf{Signcrypt}(m, S_{ID_A}, ID_B)$, then we have $m = \mathbf{Unsigncrypt}(\sigma, ID_A, S_{ID_B})$.

3.2 Security Notions

Malone-Lee [20] defines the security notions for ID-based signcryption schemes. These notions are indistinguishability against adaptive chosen ciphertext attacks and unforgeability against adaptive chosen messages attacks. We modify his definitions slightly to adapt for our ID-based signcryption scheme based on multiple PKGs.

Definition 3 (Confidentiality). *An ID-based signcryption scheme based on multiple PKGs (IDSCMP) is said to have the indistinguishability against adaptive chosen ciphertext attacks property (IND-IDSCMP-CCA2) if no polynomially bounded adversary has a non-negligible advantage in the following game.*

1. *The challenger $\mathcal{C}$ runs the* **Setup1** *and* **Setup2** *algorithms with a security parameter k and sends the system parameters to the adversary $\mathcal{A}$.*
2. *$\mathcal{A}$ performs a polynomially bounded number of queries (these queries may be made adaptively, i.e. each query may depend on the answer to the previous queries).*
   - *Key extraction queries: $\mathcal{A}$ chooses an identity ID in $PKG_l (l = 1, 2)$. $\mathcal{C}$ computes $S_{ID} = \mathbf{Extract}(ID)$ and sends S_{ID} to $\mathcal{A}$.*
   - *Signcryption queries: $\mathcal{A}$ produces two identities ID_i, ID_j and a plaintext m. $\mathcal{C}$ computes $S_{ID_i} = \mathbf{Extract}(ID_i)$ and $\sigma = \mathbf{Signcrypt}(m, S_{ID_i}, ID_j)$ and sends σ to $\mathcal{A}$.*

   – *Unsigncryption queries: $\mathcal{A}$ produces two identities ID_i and ID_j, and a ciphertext σ. $\mathcal{C}$ generates the private key $S_{ID_j} = \mathbf{Extract}(ID_j)$ and sends the result of $\mathbf{Unsigncrypt}(\sigma, ID_i, S_{ID_j})$ to $\mathcal{A}$ (this result can be the $\perp$ symbol if σ is an invalid ciphertext).*

3. *$\mathcal{A}$ generates two equal length plaintexts m_0, m_1 and two identities ID_A and ID_B on which he wants to be challenged. He cannot have asked the private key corresponding to ID_B in the first stage.*
4. *$\mathcal{C}$ takes a bit $b \in_R \{0,1\}$ and computes $\sigma = \mathbf{Signcrypt}(m_b, S_{ID_A}, ID_B)$ which is sent to $\mathcal{A}$.*
5. *$\mathcal{A}$ can ask a polynomially bounded number of queries adaptively again as in the first stage. This time, he cannot make a key extraction query on ID_B and cannot make an unsigncryption query on σ to obtain the corresponding plaintext.*
6. *Finally, $\mathcal{A}$ produces a bit b' and wins the game if $b' = b$.*

The advantage of $\mathcal{A}$ is defined as $Adv(\mathcal{A}) = |2P[b' = b] - 1|$, where $P[b' = b]$ denotes the probability that $b' = b$.

Notice that the adversary is allowed to make a key extraction query on identity ID_A in the above definition. This condition corresponds to the stringent requirement of insider security for confidentiality of signcryption [1]. On the other hand, it ensures the forward security of the scheme, i.e. confidentiality is preserved in case the sender's private key becomes compromised.

Definition 4 (Unforgeability). *An ID-based signcryption scheme based on multiple PKGs (IDSCMP) is said to have the existential unforgeability against adaptive chosen messages attacks (EUF-IDSCMP-CMA) if no polynomially bounded adversary has a non-negligible advantage in the following game.*

1. *The challenger $\mathcal{C}$ runs the* **Setup1** *and* **Setup2** *algorithms with a security parameter k and sends the system parameters to $\mathcal{A}$.*
2. *$\mathcal{A}$ performs a polynomially bounded number of queries just like in the Definition 3.*
3. *Finally, $\mathcal{A}$ produces a new triple (σ, ID_A, ID_B) (i.e. a triple that was not produced by the signcryption oracle), where the private key of ID_A was not asked in the second stage. $\mathcal{A}$ wins the game if the result of $\mathbf{Unsigncrypt}(\sigma, ID_A, S_{ID_B})$ is not the $\perp$ symbol.*

The advantage of $\mathcal{A}$ is defined as the probability that it wins.

Note that the adversary is allowed to make a key extraction query on the identity ID_B in the above definition. Again, this condition corresponds to the stringent requirement of insider security for signcryption [1].

4 An ID-Based Signcryption Scheme for Multiple PKGs

In this section, we look at signcryption between members of separate domains. This idea was first suggested in an authenticated key agreement protocol [8].

We present an ID-based signcryption scheme for multiple PKGs, which is more suitable for multi-domain ad hoc networks than previously proposed schemes. The following shows the details of our scheme.

Setup1: Define G_1, G_2 and $\hat{e}$ as in previous section. Let H_1, H_2 and H_3 be three cryptographic hash functions where $H_1 : \{0,1\}^* \to G_1$, $H_2 : G_2 \to \{0,1\}^n$ and $H_3 : \{0,1\}^* \to Z_q^*$. Let P be a generator of G_1. Note that the system's public parameters $\{G_1, G_2, n, \hat{e}, P, H_1, H_2, H_3\}$ are globally agreed, e.g., recommended by an international standards body.

Setup2: The PKG$_1$ chooses a master secret key $s_1 \in_R Z_q^*$ and computes $P_{pub}^1 = s_1 P$. The PKG$_1$ publishes P_{pub}^1 and keeps the master secret key s_1 secret. Similarly, the PKG$_2$ chooses a master secret key $s_2 \in_R Z_q^*$ and computes $P_{pub}^2 = s_2 P$. The PKG$_2$ publishes P_{pub}^2 and keeps the master secret key s_2 secret.

Extract: Suppose that Alice registers with PKG$_1$ and gets her private key $S_{ID_A} = s_1 Q_{ID_A}$, where $Q_{ID_A} = H_1(ID_A)$, and Bob registers with PKG$_2$ and gets his private key $S_{ID_B} = s_2 Q_{ID_B}$, where $Q_{ID_B} = H_1(ID_B)$.

Signcrypt: To send a message m to Bob, Alice follows the steps below.

1. Choose $x \in_R Z_q^*$ and compute $U = xP$.
2. Compute $\tau = \hat{e}(P_{pub}^2, Q_{ID_B})^x$.
3. Compute $k = H_2(\tau)$.
4. Compute $c = m \oplus k$.
5. Compute $r = H_3(m, U, k)$.
6. Compute $V = xP_{pub}^1 + rS_{ID_A}$.

The ciphertext is $\sigma = (c, U, V)$.

Unsigncrypt: When receiving $\sigma = (c, U, V)$, Bob follows the steps below.

1. Compute $\tau = \hat{e}(U, S_{ID_B})$.
2. Compute $k = H_2(\tau)$.
3. Recover $m = c \oplus k$.
4. Compute $r = H_3(m, U, k)$.
5. Accept the message if and only if the following equation holds:

$$\hat{e}(P, V) = \hat{e}(U, P_{pub}^1)\hat{e}(P_{pub}^1, Q_{ID_A})^r.$$

Note that we accept the assumption in [8] that the two PKGs share common system parameters and differ in the master secret key. Of course, our scheme can use different system parameters by using the method in [16,17].

5 Analysis of the Scheme

5.1 Correctness

The correctness can be easily verified by the following equations.

$$\hat{e}(U, S_{ID_B}) = \hat{e}(xP, s_2 Q_{ID_B}) = \hat{e}(xs_2 P, Q_{ID_B}) = \hat{e}(P_{pub}^2, Q_{ID_B})^x$$

and

$$\begin{aligned}\hat{e}(P,V) &= \hat{e}(P, xP_{pub}^1 + rS_{ID_A}) = \hat{e}(P, xP_{pub}^1)\hat{e}(P, rs_1Q_{ID_A}) \\ &= \hat{e}(xP, P_{pub}^1)\hat{e}(s_1P, Q_{ID_A})^r \\ &= \hat{e}(U, P_{pub}^1)\hat{e}(P_{pub}^1, Q_{ID_A})^r\end{aligned}$$

5.2 Security

Theorem 1 (Confidentiality). *In the random oracle model, we assume we have an IND-IDSCMP-CCA2 adversary called $\mathcal{A}$ that is able to distinguish ciphertext during the game of Definition 3 with an advantage ϵ when running in a time t and asking at most q_{H_1} identity hashing queries, at most q_{H_2} H_2 queries, at most q_{H_3} H_3 queries, at most q_K key extraction queries, q_S signcryption queries and q_U unsigncryption queries. Then, there exists a distinguisher $\mathcal{C}$ that can solve the Decisional Bilinear Diffie-Hellman problem in a time $O(t + (q_{H_3}q_S + q_S^2 + 4q_U)T_{\hat{e}})$ with an advantage*

$$Adv(\mathcal{C})^{DBDH(G_1,P)} > \frac{\epsilon(2^k - q_U) - q_U}{q_{H_1}2^{k+1}},$$

where $T_{\hat{e}}$ denotes the computation time of the bilinear map.

Proof. We assume the distinguisher $\mathcal{C}$ receives a random instance (P, aP, bP, cP, h) of the Decisional Bilinear Diffie-Hellman problem. His goal is to decide whether $h = \hat{e}(P,P)^{abc}$ or not. $\mathcal{C}$ will run $\mathcal{A}$ as a subroutine and act as $\mathcal{A}$'s challenger in the IND-IDSCMP-CCA2 game. During the game, $\mathcal{A}$ will consult $\mathcal{C}$ for answers to the random oracles H_1, H_2 and H_3. Roughly speaking, these answers are randomly generated, but to maintain the consistency and to avoid collision, $\mathcal{C}$ keeps three lists L_1, L_2, L_3 respectively to store the answers. The following assumptions are made.

1. $\mathcal{A}$ will ask for $H_1(ID)$ before ID is used in any key extraction query, signcryption query and unsigncryption query.
2. Ciphertext returned from a signcryption query will not be used by $\mathcal{A}$ in an unsigncryption query.

At the beginning of the game, $\mathcal{C}$ gives $\mathcal{A}$ the system parameters with $P_{pub}^2 = cP$ and $P_{pub}^1 = dP$, where $d \in_R Z_q^*$. Note that c and d are unknown to $\mathcal{C}$. This value simulates the master secret key value for the PKG$_2$ and PKG$_1$ in the game. Then, $\mathcal{C}$ chooses a random number $j \in \{1, 2, \ldots, q_{H_1}\}$. $\mathcal{A}$ asks a polynomially bounded number of H_1 queries on identities of his choice. At the j-th H_1 query, $\mathcal{C}$ answers by $H_1(ID_j) = bP$ (We suppose that the identity ID_j belongs to PKG$_2$, otherwise we exchange P_{pub}^2 for P_{pub}^1). For queries $H_1(ID_e)$ with $e \neq j$, $\mathcal{C}$ chooses $b_e \in_R Z_q^*$, puts the pair (ID_e, b_e) in list L_1 and answers $H_1(ID_e) = b_eP$.

We now explain how the other kinds of queries are treated by $\mathcal{C}$.

- H_2 queries: On a $H_2(\tau_e)$ query, $\mathcal{C}$ searches a pair (τ_e, k_e) in the list L_2. If such a pair is found, $\mathcal{C}$ answers k_e, otherwise he answers $\mathcal{A}$ by a random binary sequence $k \in_R \{0,1\}^n$ such that no entry $(\cdot, k)$ exists in L_2 (in order to avoid collisions on H_2) and puts the pair (τ_e, k) into L_2.
- H_3 queries: On a $H_3(m_e, U_e, k_e)$ query, $\mathcal{C}$ checks if there exists (m_e, U_e, k_e, r_e) in L_3. If such a tuple is found, $\mathcal{C}$ answers r_e, otherwise he chooses $r \in_R Z_q^*$, gives it as an answer to the query and puts the tuple (m_e, U_e, k_e, r) into L_3.
- Key extraction queries: When $\mathcal{A}$ asks a question **Extract**(ID_e), if $ID_e = ID_j$, then $\mathcal{C}$ fails and stops. If $ID_e \neq ID_j$, then the list L_1 must contain a pair (ID_e, b_e) for some b_e (this indicates $\mathcal{C}$ previously answered $H_1(ID_e) = b_eP$ on a H_1 query on ID_e). The private key corresponding to ID_e is then $b_eP_{pub}^2 = cb_eP$ or $b_eP_{pub}^1 = db_eP$. It is computed by $\mathcal{C}$ and returned to $\mathcal{A}$.
- Signcryption queries: At any time, $\mathcal{A}$ can perform a signcryption query for a plaintext m and identities ID_A and ID_B. We have the following three cases to consider.
  - Case 1: $ID_A \neq ID_j$. $\mathcal{C}$ computes the private key S_{ID_A} corresponding to ID_A by running the key extraction query algorithm. Then $\mathcal{C}$ answers the query by a call to **Signcrypt**(m, S_{ID_A}, Q_{ID_B}).
  - Case 2: $ID_A = ID_j$ and $ID_B \neq ID_j$. $\mathcal{C}$ chooses $x, r \in_R Z_q^*$ and computes $U = xP - rQ_{ID_A}$, $V = xP_{pub}^1$, and $\tau = \hat{e}(U, S_{ID_B})$($\mathcal{C}$ could obtain S_{ID_B} from the key extraction algorithm because $ID_B \neq ID_j$). $\mathcal{C}$ runs the H_2 simulation algorithm to find $k = H_2(\tau)$ and computes $c = m \oplus k$. $\mathcal{C}$ then checks if L_3 already contains a tuple (m, U, k, r') with $r' \neq r$. In this case, $\mathcal{C}$ repeats the process with another random pair (x, r) until finding a tuple (m, U, k, r) whose first three elements do not appear in a tuple of the list L_3. This process repeats at most $q_{H_3} + q_S$ times as L_3 contains at most $q_{H_3} + q_S$ entries ($\mathcal{A}$ can issue q_{H_3} H_3 queries and q_S signcryption queries, while each signcryption query contains a single H_3 query). When an appropriate pair (x, r) is found, the ciphertext (c, U, V) appears to be valid from $\mathcal{A}$'s viewpoint. $\mathcal{C}$ has to compute one pairing operation for each iteration of the process.
  - Case 3: $ID_A = ID_j$ and $ID_B = ID_j$. $\mathcal{C}$ chooses $x^*, r^* \in_R Z_q^*$, computes $U^* = x^*P - r^*Q_{ID_A}$, $V^* = x^*P_{pub}^1$, and chooses $\tau^* \in_R G_2$ and $k^* \in_R \{0,1\}^n$ such that no entry $(\cdot, k^*)$ is in L_2 and computes $c^* = m \oplus k^*$. $\mathcal{C}$ then checks if L_3 already contains a tuple (m, U^*, k^*, r') with $r' \neq r^*$. If not, $\mathcal{C}$ puts the tuple (m, U^*, k^*, r^*) into L_3 and (τ^*, k^*) into L_2. Otherwise, $\mathcal{C}$ chooses another random pair (x^*, r^*) and repeats the process as above until he finds a tuple (m, U^*, k^*, r^*) whose first three elements do not appear in an entry of L_3. Once an appropriate pair (x^*, r^*) is found, $\mathcal{C}$ gives the ciphertext $\sigma^* = (c^*, U^*, V^*)$ to $\mathcal{A}$. As $\mathcal{A}$ will not ask for the unsigncryption of σ^*, he will never see that σ^* is not a valid ciphertext of the plaintext m for identities ID_A and ID_B.

– Unsigncryption queries: For a unsigncryption query on a ciphertext $\sigma' = (c', U', V')$ for identities ID_A and ID_B. We have the following two cases to consider.
  - Case 1: $ID_B = ID_j$. $\mathcal{C}$ always answers $\mathcal{A}$ that σ' is invalid.
  - Case 2: $ID_B \neq ID_j$. $\mathcal{C}$ computes $\tau' = \hat{e}(U', S_{ID_B})$ ($\mathcal{C}$ could obtain S_{ID_B} from the key extraction algorithm because $ID_B \neq ID_j$). $\mathcal{C}$ then runs the H_2 simulation algorithm to obtain $k' = H_2(\tau')$ and computes $m' = c' \oplus k'$. Finally, $\mathcal{C}$ runs the H_3 simulation algorithm to obtain $r' = H_3(m', U', k')$ and checks if $\hat{e}(P, V') = \hat{e}(U', P^1_{pub})\hat{e}(P^1_{pub}, Q_{ID_A})^{r'}$ holds. If the above equation does not hold, $\mathcal{C}$ rejects the ciphertext. Otherwise $\mathcal{C}$ returns m'.

  It is easy to see that, for all queries, the probability to reject a valid ciphertext does not exceed $q_U/2^k$.

After the first stage, $\mathcal{A}$ picks a pair of identities on which he wishes to be challenged. Note that $\mathcal{C}$ fails if $\mathcal{A}$ has asked a key extraction query on ID_j during the first stage. We know that the probability for $\mathcal{C}$ not to fail in this stage is $\frac{q_{H_1}-q_K}{q_{H_1}}$. Further, with a probability exactly$\frac{1}{q_{H_1}-q_K}$, $\mathcal{A}$ chooses to be challenged on the pair (ID_i, ID_j) with $i \neq j$. Hence the probability that $\mathcal{A}$'s response is helpful to $\mathcal{C}$ is $\frac{1}{q_{H_1}}$. Note that if $\mathcal{A}$ has submitted a key extraction query on ID_j, then $\mathcal{C}$ fails because he is unable to answer the question. On the other hand, if $\mathcal{A}$ does not choose (ID_i, ID_j) as target identities, $\mathcal{C}$ fails too.

Then $\mathcal{A}$ outputs two plaintexts m_0 and m_1. $\mathcal{C}$ chooses $b \in_R \{0, 1\}$ and signcrypts m_b. To do so, he sets $U^* = aP$, obtains $k^* = H_2(h)$(where h is $\mathcal{C}$ candidate for the DBDH problem) from the H_2 simulation algorithm, and computes $c_b = m \oplus k^*$. Then $\mathcal{C}$ chooses $V^* \in_R G_1$ and sends the ciphertext $\sigma^* = (c_b, U^*, V^*)$ to $\mathcal{A}$.

$\mathcal{A}$ then performs a second series of queries which is treated in the same way as the first one. At the end of the simulation, he produces a bit b' for which he believes the relation $\sigma^* = \mathbf{Signcrypt}(m_{b'}, S_{ID_i}, ID_j)$ holds. At this moment, if $b = b'$, $\mathcal{C}$ outputs $h = \hat{e}(U^*, S_{ID_j}) = \hat{e}(aP, cbP) = \hat{e}(P, P)^{abc}$ as a solution of the DBDH problem, otherwise $\mathcal{C}$ stops and outputs "failure".

Taking into account all the probabilities that $\mathcal{C}$ will not fail its simulation, the probability that $\mathcal{A}$ chooses to be challenged on the pair (ID_i, ID_j), and also the probability that $\mathcal{A}$ wins the IND-IDSCMP-CCA2 game, the value of $Adv(\mathcal{C})$ is calculated as follows.

$$\begin{aligned} Adv(\mathcal{C}) &= (\frac{(\epsilon+1)}{2}(1 - \frac{q_U}{2^k}) - \frac{1}{2})(\frac{1}{q_{H_1}}) \\ &= \frac{\epsilon(2^k - q_U) - q_U}{q_{H_1} 2^{k+1}} \end{aligned}$$

The bound on $\mathcal{C}$'s computation time derives from the fact that every signcryption query requires at most $q_{H_3} + q_S$ pairing operations and every unsigncryption query requires at most 4 pairing operations. □

Theorem 2 (Unforgeability). *The proposed scheme is secure in the sense of unforgeability.*

Proof. The unforgeability against adaptive chosen messages attacks derives from the security of Hess's ID-based signature scheme [13] under the Computational Diffie-Hellman assumption. One can show that an attacker that is able to forge a signcrypted message must be able to forge a signature for the following scheme which is a variant of Hess's signature.

Setup and **Extract** are the same as above. The others are described as follows.

Sign: To sign a message m, the signer follows the steps below.

1. Choose $x \in_R Z_q^*$ and compute $U = xP$.
2. Compute $r = H_3(m, U)$.
3. Compute $V = xP_{pub}^1 + rS_{ID_A}$.

The signature on message m is $\sigma = (U, V)$.

Verify: When receiving $\sigma = (U, V)$, the verifier follows the steps below.

1. Compute $r = H_3(m, U)$.
2. Accept the signature if and only if the following equation holds:

$$\hat{e}(P, V) = \hat{e}(U, P_{pub}^1)\hat{e}(P_{pub}^1, Q_{ID_A})^r.$$

□

Theorem 3 (Public verifiability). *The proposed scheme provides the public verifiability.*

Proof. When necessary, Bob may forward (m, U, V) to others, who can be convinced that it came originally from Alice by computing $r = H_3(m, U)$ and verifying

$$\hat{e}(P, V) = \hat{e}(U, P_{pub}^1)\hat{e}(P_{pub}^1, Q_{ID_A})^r.$$

Therefor, our scheme provides the public verifiability. □

6 Conclusions

We have proposed an ID-based signcryption scheme based on the bilinear pairings. Our scheme can work in multiple PKGs environment. We proved that our scheme satisfies the confidentiality, the unforgeability, and the public verifiability. As compared with previously proposed schemes based on a single PKG, our scheme is more suitable for multi-domain ad hoc networks.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China under contract no. 60473029.

References

1. J.H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Advances in Cryptology-EUROCRYPT 2002*, LNCS 2332, pp. 83–107, Springer-Verlag, 2002.
2. J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. In *Public Key Cryptography-PKC 2002*, LNCS 2274, pp. 80–98, Springer-Verlag, 2002.
3. F. Bao and R.H. Deng. A signcryption scheme with signature directly verifiable by public key. In *Public Key Cryptography-PKC'98*, LNCS 1431, pp. 55–59, Springer-Verlag, 1998.
4. P.S.L.M. Barreto, B. Libert, N. McCullagh, and J.J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Advances in Cryptology-ASIACRYPT 2005*, LNCS 3788, pp. 515–532, Springer-Verlag, 2005.
5. M. Bohio and A. Miri. An authenticated broadcasting scheme for wireless ad hoc network. In *2nd Annual Conference on Communication Networks and Services Research-CNSR'04*, pp. 69–74, Fredericton, Canada, 2004.
6. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology-CRYPTO 2001*, LNCS 2139, pp. 213–229, Springer-Verlag, 2001.
7. X. Boyen. Multipurpose identity-based signcryption: a swiss army knife for identity-based cryptography. In *Advances in Cryptology-CRYPTO 2003*, LNCS 2729, pp. 383–399, Springer-Verlag, 2003.
8. L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairings. In *16th IEEE Computer Security Foundations Workshop-CSFW'03*, pp. 219–233, Pacific Grove, USA, 2003.
9. L. Chen and J. Malone-Lee. Improved identity-based signcryption. In *Public Key Cryptography-PKC 2005*, LNCS 3386, pp. 362–379, Springer-Verlag, 2005.
10. S.S.M. Chow, S.M. Yiu, L.C.K. Hui, and K.P. Chow. Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity. In *Information Security and Cryptology-ICISC 2003*, LNCS 2971, pp. 352–369, Springer-Verlag, 2004.
11. H. Deng and D.P. Agrawal. TIDS: threshold and identity-based security scheme for wireless ad hoc networks. *Ad Hoc Networks*, Vol. 2, No. 3, pp. 291–307, 2004.
12. C. Gamage, J. Leiwo, and Y. Zheng. Encrypted message authentication by firewalls. In *Public Key Cryptography-PKC'99*, LNCS 1560, pp. 69–81, Springer-Verlag, 1999.
13. F. Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography-SAC 2002*, LNCS 2595, pp. 310–324, Springer-Verlag, 2003.
14. H.Y. Jung, D.H. Lee, J.I. Lim, and K.S. Chang. Signcryption schemes with forward secrecy. In *Information Security Application-WISA 2001*, pp. 463–475, Seoul, Korea, 2001.
15. D. Kidston and J. Robinson. Distributed network management for coalition deployments. In *21st Century Military Communications Conference-MILCOM 2000*, Vol. 1, pp. 460–464, Los Angeles, USA, 2000.
16. S. Kim, H. Lee, and H. Oh. Enhanced ID-based authenticated key agreement protocols for a multiple independent PKG environment. In *Information and Communications Security-ICICS 2005*, LNCS 3783, pp. 323–335, Springer-Verlag, 2005.
17. H. Lee, D. Kim, S. Kim, and H. Oh. Identity-based key agreement protocols in a multiple PKG environment. In *Computational Science and Its Applications-ICCSA 2005*, LNCS 3483, pp. 877–886, Springer-Verlag, 2005.

18. G. Li and W. Han. A new scheme for key management in ad hoc networks. In *4th International Conference on Networking-ICN 2005*, LNCS 3421, pp. 242–249, Springer-Verlag, 2005.
19. B. Libert and J.J. Quisquater. A new identity based signcryption schemes from pairings. In *2003 IEEE information theory workshop*, pp. 155–158, Paris, France, 2003.
20. J. Malone-Lee. Identity based signcryption. *Cryptology ePrint Archive*, Report 2002/098, 2002. Available from: http://eprint.iacr.org/2002/098.
21. J. Malone-Lee and W. Mao. Two birds one stone: signcryption using RSA. In *Topics in Cryptology-CT-RSA 2003*, LNCS 2612, pp. 211–226, Springer-Verlag, 2003.
22. Y. Mu and V. Varadharajan. Distributed signcryption. In *Progress in Cryptology-INDOCRYPT 2000*, LNCS 1977, pp. 155–164, Springer-Verlag, 2000.
23. B.N. Park, J. Myung, and W. Lee. ISSRP: a secure routing protocol using identity-based signcryption scheme in ad-hoc networks. In *Parallel and Distributed Computing: Applications and Technologies-PDCAT 2004*, LNCS 3320, pp. 711–714, Springer-Verlag, 2004.
24. B.N. Park and W. Lee. ISMANET: a secure routing protocol using identity-based signcryption scheme for mobile ad-hoc networks. *IEICE Transactions on Communications*, Vol. E88-B, No. 6, pp. 2548–2556, 2005.
25. M. Seo and K. Kim. Electronic funds transfer protocol using domain-verifiable signcryption scheme. In *Information Security and Cryptology-ICISC'99*, LNCS 1787, pp. 269–277, Springer-Verlag, 1999.
26. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology-CRYPTO'84*, LNCS 196, pp. 47–53, Springer-Verlag, 1984.
27. D.H. Yum and P.J. Lee. New signcryption schemes based on KCDSA. In *Information Security and Cryptology-ICISC 2001*, LNCS 2288, pp. 305–317, Springer-Verlag, 2002.
28. Y. Zheng. Digital signcryption or how to achieve cost (signature & encryption) $\ll$ cost (signature) + cost(encryption). In *Advances in Cryptology-CRYPTO'97*, LNCS 1294, pp. 165–179, Springer-Verlag, 1997.
29. Y. Zheng and H. Imai. How to construct efficient signcryption schemes on elliptic curves. *Information Processing Letters*, Vol. 68, No.5, pp. 227–233, 1998.

Efficient Self-healing Key Distribution with Revocation for Wireless Sensor Networks Using One Way Key Chains

Ratna Dutta[1], Ee-Chien Chang[2], and Sourav Mukhopadhyay[2]

[1] Computing Division
Systems and Security Department
Institute for Infocomm Research
21, Heng Mui Keng Terrace, Singapore 119613
dratna@i2r.a-star.edu.sg
[2] School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543
{changec,sourav}@comp.nus.edu.sg

Abstract. Security of group communication for large mobile wireless sensor network hinges on efficient key distribution and key management mechanism. As the wireless medium is characterized by its lossy nature, reliable communication cannot be assumed in the key distribution schemes. Therefore, self-healing is a good property for key distribution in wireless applications. The main idea of self-healing key distribution scheme is that even if during a certain session some broadcast messages are lost due to network faults, the users are capable of recovering lost session keys on their own, without requesting additional transmission from the group manager. The only requirement for a user to recover the lost session keys, is its membership in the group both before and after the sessions in which the broadcast packets containing the keys are sent. Self-healing approach of key distribution is stateless in the sense that a user who has been off-line for some period is able to recover the lost session keys immediately after coming back on-line. In this paper, we propose two constructions for scalable self-healing key distribution with t revocation capability. The novelty of our constructions are that we apply a different and more efficient self-healing mechanism compared to the ones in the literature using one-way key chain. The main improvements that our proposed schemes achieve over previous approaches are

(a) communication bandwidth reduces from $O((tj + j - t - 1)\log q)$ to $O((t + 1)\log q)$, and
(b) computation costs for our first and second constructions reduce from $O(2tj + j)$ to $O(2t + 1)$ and $O(2(t^2 + t))$ respectively,

where m is the maximum number of sessions, j is the current session number, t is the maximum number of compromised group members that may collude and q is a large prime number. We achieve this result without any increase in the storage complexity. The schemes are scalable to very large groups in highly mobile, volatile and hostile network. We prove in an appropriate security framework that our constructions are computationally secure and achieve both forward secrecy and backward secrecy.

Keywords: sensor network, session key distribution, self-healing, revocation, computational security.

1 Introduction

Secure group communication relies on secure and robust distribution of group keys. A single symmetric key known only to the group members can effectively protect a multicast group. However, only legitimate users should have access to the group communication in order to achieve privacy. Thus the group key (session key) must be updated each time when new users join or old users leave the group and securely redistributed to the existing members of the group. This is referred to as group rekeying. The newly joint users should not be able to derive the previous group keys, even if they are able to derive future group keys with subsequently distributed keying information. Similarly, the revoked users should not be able to derive the future session keys, even if they are able to compute the previous session keys with previously distributed keying information. If a group is rekeyed on each membership change, the frequency of rekeying becomes the primary bottleneck as the size of the group grows and/or the rate of membership change increases. Therefore, scalable group rekeying is an important and challenging problem to be addressed in order to support secure multicast communication for dynamic groups, where typical systems are large: tens of millions of users. How to distribute and update session key efficiently over an unreliable channel is an interesting research topic.

Self-Healing Key Distribution: In this paper, we address *self-healing key distribution scheme with revocation* [22] that deals with the problem of distributing session keys for secure communication to a dynamic group of users over an unreliable, lossy network in a manner that is resistant to packet lost and collusion attacks. The main concept of self-healing key distribution schemes is that users, in a large and dynamic group communication over an unreliable network, can recover lost session keys on their own, even if lost some previous key distribution messages, without requesting additional transmissions from the group manager. This reduces network traffic and risk of user exposure through traffic analysis and also decreases the work load on the group manager. The key idea of self-healing key distribution schemes is to broadcast information that is useful only for trusted members. Combined with its pre-distributed secrets, this broadcast information enables a trusted member to reconstruct a shared key. On the contrary, a revoked member is unable to infer useful information from the broadcast. The only requirement that a user must satisfy to recover the lost keys through self-healing, is its membership in the group both before and after the sessions in which the broadcast packet containing the key is sent. A user who has been off-line for some period is able to recover the lost session keys immediately after coming back on-line. Thus self-healing approach of key distribution is stateless. The scheme is said to have t-revocation capability if the key distribution mechanism cannot be broken by any coalition of up to t users.

Our Contribution: This paper focuses on designing computationally secure and efficient key distribution schemes with self-healing property and revocation capability for large and dynamic groups over insecure wireless networks. We propose two new constructions for self-healing key distributions adopting a new self-healing technique. The novelty of our self-healing approach is that it uses one-way key chain that is more efficient compared to the self-healing techniques used in the previous schemes [4, 12, 14, 22]. This yields an improved secret-sharing based self-healing key distribution scheme (Construction 2) compared to the secret-sharing based scheme in [4]. We obtain further improvement (Construction 1) using polynomial based revocation which is more efficient compared to all the previous scheme. The main attraction of this paper is that our constructions have significant improvements in terms of both communication and computation overhead without any increase in the storage complexity. Table 1 summarizes the comparison of our schemes with the previous approaches (m being the maximum number of sessions, j stands for the current session number and q is a prime large enough to accommodate a cryptographic key).

Table 1. Comparison among different self-healing key distribution schemes in j-th session

Schemes	Storage Overhead	Communication Overhead	Computation Overhead
Construction 3 of [22]	$(m-j+1)^2 \log q$	$(mt^2+2mt+m+t)\log q$	$2mt^2+3mt-t$
Scheme 3 of [14]	$2(m-j+1)\log q$	$[(m+j+1)t+(m+1)]\log q$	$mt+t+2tj+j$
Scheme 2 of [4]	$(m-j+1)\log q$	$(2tj+j)\log q$	$2j(t^2+t)$
Construction 1 of [12]	$(m-j+1)\log q$	$(tj+j-t-1)\log q$	$2tj+j$
Our Construction 1	$(m-j+1)\log q$	$(t+1)\log q$	$2t+1$
Our Construction 2	$(m-j+1)\log q$	$(t+1)\log q$	$2(t^2+t)$

We emphasize that each user in both the proposed constructions requires $(m-j+1)\log q$ memory and size of the broadcast message at the j-th session is $(t+1)\log q$ with computation costs $2t+1$ and $2(t^2+t)$ respectively. Our key distribution schemes are scalable to very large groups in highly mobile, volatile and hostile wireless network as the communication and computation overhead does not depend on the size of the group, instead they depend on the number of compromised group members that may collude together. We have shown in an appropriate security model that our proposed constructions are computationally secure and achieve both forward secrecy and backward secrecy.

Related Works: Broadcast encryption is a closely related area which has received much attention from both the network and cryptography community. Efficient key distribution and key management mechanisms are at the core of this. The area of broadcast encryption was formally defined by Fiat and Naor [10] after the work of Berkovits [1] and has been extensively studied since then. A number of approaches have been proposed and has grown up in different directions: rekeying schemes for dynamic groups, broadcast schemes with tracing capability, users revocation from a predefined subset of users *etc.* A few of them are [2, 3, 5, 6, 7, 8, 10, 11, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27]. However, the underlying networks are assumed to be reliable in all the above works. Self-healing key distribution with revocation was first introduced by Staddon *et al.* in [22]. They provide formal definitions and security notions that were later generalized by Liu *et al.* [14] and Blundo *et al.* [4]. The constructions given in [22] suffers from high storage and communication overhead. Liu *et al.* [14] introduced a novel personal key distribution scheme and combining it with the self-healing technique in [22], they proposed a new construction that improves the storage and communication overhead greatly. Blundo *et al.* [4] showed an attack to the first construction in [22] and developed a new self-healing technique different from [22] under a slightly modified framework. More recently, Hong *et al.* [12] proposed self-healing key distribution constructions having less storage and communication complexity. Recently, Dutta and Mukhopadhyay [9] proposed a new storage efficient self-healing key distribution scheme.

Applications: The spectrum of applicability of self-healing key distribution is quite large. Self-healing key distribution is a potential candidate to establish session keys for secure communication to large and dynamic groups in highly mobile, volatile and hostile wireless network, where frequent membership changes may be necessary and ability to revoke users during certain exchanges is desirable. In such situations the session keys need to be used for a short time-period or need to be updated frequently. Mobile wireless ad hoc networks have wide applications in military operations, rescue missions and scientific explorations, where there are usually no network infrastructure support and the adversary may intercept, modify, and/or partially interrupt the communication. In such applications, security becomes a critical concern. The traditional approaches for key distribution and group re-keying used for reliable network, are not suitable for large and dynamic wireless networks because of the lossy nature of wireless medium. Therefore, self-healing is a good property for key distribution in wireless mobile and ad hoc networks, where the nodes/devices are powered by batteries and have the unique feature of moving in and out of range frequently. Hence expensive computations like the ones required by public key cryptography are not suitable for such networks. For example, military networks consist of mobile devices carried by soldiers, automatic weapons, sensing devices *etc.* and there could be a need in a battle field for a rapid revocation of devices caught by the enemy. Also there might be situations where some users are not constantly on-line or experience burst packet losses. It can rejoin the group once the power is on again. All these aspects can take great advantage from self-healing key distribution schemes with revocation capability. Self-healing key distribution schemes have also found applicable in broadcast communication over low-cost channels, pay-per-view

TV, information service delivering sensitive content/information to authorized recipients and several other Internet-related settings.

Organization: The rest of the paper is organized as follows. Section 2 presents notations to be used in the paper and our security model. Section 3 describes the details of our constructions. We provide a proof of security of our proposed schemes in Section 4. Section 5 focuses on the performance analysis of the schemes and their comparison with the previous works. Finally, we conclude in Section 6.

2 Preliminaries

This section briefly define our security model for self-healing key distribution.

Table 2. Notations

$\mathcal{U}$	: set of all users in the networks
U_i	: i-th user
GM	: group manager
n	: total number of users in the network
m	: total number of sessions
t	: the maximum number of compromised user
F_q	: a field of order q
S_i	: personal secret of user U_i
SK_j	: session key generated by the GM in session j
$\mathcal{B}_j$	: broadcast message by the GM during session j
$Z_{i,j}$	: the information learned by U_i through $\mathcal{B}_j$ and S_i
R_j	: the set of all revoked users in session j
$\mathcal{H}$	: a cryptographically secure one-way function
S^F	: forward key seed generated by the GM
S^B	: backward key seed generated by the GM
K_i^F	: i-th forward key in the forward key chain
K_i^B	: i-th backward key in the backward key chain

2.1 Our Security Model

We now state the following definitions that are aimed to computational security for session key distribution adopting the security model of [14, 22].

Definition 2.1 *(Session Key Distribution with privacy [22]) Let* $t, i \in \{1, \ldots, n\}$ *and* $j \in \{1, \ldots, m\}$.

1. *$\mathcal{D}$ is a session key distribution with privacy if*
   - *(a) for any user U_i, the session key SK_j is efficiently determined from $\mathcal{B}_j$ and S_i.*
   - *(b) for any set $R \subseteq \mathcal{U}$, $|R| \leq t$, and $U_i \notin R$, it is computationally infeasible for users in R to determine the personal key S_i.*

(c) *what users $U_1, \ldots, U_n$ learn from $\mathcal{B}_j$ cannot be determined from broadcasts or personal keys alone.* i.e. *if we consider separately either the set of m broadcasts $\{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$ or the set of n personal keys $\{S_1, \ldots, S_n\}$, then it is computationally infeasible to compute session key SK_j (or other useful information) from either set.*

2. *$\mathcal{D}$ has t-revocation capability if given any $R \subseteq \mathcal{U}$, where $|R| \leq t$, the group manager GM can generate a broadcast $\mathcal{B}_j$, such that for all $U_i \notin R$, U_i can efficiently recover the session key SK_j, but the revoked users cannot.* i.e. *it is computationally infeasible to compute SK_j from $\mathcal{B}_j$ and $\{S_l\}_{U_l \in R}$.*
3. *$\mathcal{D}$ is self-healing if the following is true for any j, $1 \leq j_1 < j < j_2 \leq m$: For any user U_i who is a member in sessions j_1 and j_2, the key SK_j is efficiently determined by the set $\{Z_{i,j_1}, Z_{i,j_2}\}$.*

Definition 2.2 *(t-wise forward and backward secrecy [14]) Let $t, i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.*

1. *A key distribution scheme $\mathcal{D}$ guarantees t-wise forward secrecy if for any set $R \subseteq \mathcal{U}$, where $|R| \leq t$, and all $U_l \in R$ are revoked before session j, it is computationally infeasible for the members in R together to get any information about SK_j, even with the knowledge of group keys $\mathsf{SK}_1, \ldots, \mathsf{SK}_{j-1}$ before session j.*
2. *A session key distribution $\mathcal{D}$ guarantees t-wise backward secrecy if for any set $J \subseteq \mathcal{U}$, where $|J| \leq t$, and all $U_l \in J$ join after session j, it is computationally infeasible for the members in J together to get any information about K_j, even with the knowledge of group keys $\mathsf{SK}_{j+1}, \ldots, \mathsf{SK}_m$ after session j.*

3 Our Constructions

In this section, we present two constructions for self-healing key distribution with revocation capability. We use revocation polynomial in our first construction and apply secret sharing in our second construction. Unlike previous approaches, we adopt a different technique to perform self-healing which is more efficient from both communication and computation point of view compared to the previous techniques.

We consider a setting in which there is a group manager (GM) and n users $\mathcal{U} = \{U_1, \ldots, U_n\}$. All of our operations take place in a finite field, F_q, where q is a large prime number ($q > n$). In our setting, we never allow a revoked user to rejoin the group in a later session. Let $\mathcal{H} : F_q \longrightarrow F_q$ be a cryptographically secure one-way function.

3.1 Construction 1: Revocation Using Polynomial

– *Setup*: The group manager randomly picks two initial key seeds, the forward key seed $S^F \in F_q$ and the backward key seed $S^B \in F_q$. It repeatedly applies (in the pre-processing time) the one-way function $\mathcal{H}$ on S^B and computes the one-way key chain of length m:

$$K_i^B = \mathcal{H}(K_{i-1}^B) = \mathcal{H}^{i-1}(S^B)$$

for $1 \leq i \leq m$. The j-th session key is computed as

$$\mathsf{SK}_j = K_j^F + K_{m-j+1}^B,$$

where $K_j^F = \mathcal{H}^{j-1}(S^F)$. The group manager chooses independently and uniformly at random m t-degree polynomials $f_1(x), \ldots, f_m(x) \in F_q[x]$, $t < m, n$. Each user U_i, for $1 \leq i \leq n$, receives its personal secret keys corresponding to the m sessions $S_i = \{f_1(i), \ldots, f_m(i)\}$ and the forward key seed S_F from the group manager via the secure communication channel between them.

– *Broadcast*: Let $R_j = \{U_{l_1}, \ldots, U_{l_{w_j}}\}$ be the set of all revoked users for sessions in and before j such that $|R_j| = w_j \leq t$. In the j-th session the group manager locates the backward key K_{m-j+1}^B in the backward key chain and computes the polynomials

$$r_j(x) = (x - l_1) \cdots (x - l_{w_j}),$$

$$h_j(x) = K_{m-j+1}^B r_j(x) + f_j(x).$$

The polynomial $r_j(x)$ is called the revocation polynomial in session j and the polynomial $f_j(x)$ plays the role of masking polynomial in session j. The group manager broadcasts the following message $\mathcal{B}_j$:

$$\mathcal{B}_j = R_j \cup \{h_j(x)\}.$$

– *Session Key Recovery*: When a non-revoked user U_i receives the j-th session key distribution message $\mathcal{B}_j$, it evaluates the polynomial $r_j(x)$ at point i and recovers

$$K_{m-j+1}^B = \frac{h_j(i) - f_j(i)}{r_j(i)}.$$

Finally, U_i computes the j-th forward key $K_j^F = \mathcal{H}^{j-1}(S^F)$ and evaluates the current session key

$$\mathsf{SK}_j = K_j^F + K_{m-j+1}^B.$$

– *Add Group Members*: When the group manager adds a new group member starting from session j, it picks an unused identity $v \in F_q$, computes the personal secret keys corresponding to the current and future sessions $S_v = \{f_j(v), f_{j+1}(v), \ldots, f_m(v)\}$ and gives $\{v, S_v, K_j^F\}$ to this new group member via the secure communication channel between them.

– *Re-initialization:* The system fails when all m sessions are exhausted, or when number of revoked users becomes more than t. At this phase, re-initialization is required and a new setup is executed.

Complexity:

Storage overhead: Storage complexity of personal key for each user is $m \log q$ bits. The group members that join later need to store less data. Foe example, the personal key for a user joining in j-th session occupies $(m - j + 1) \log q$ bits memory space.

Communication overhead: Communication bandwidth for key management is $(t+1)\log q$ bits. Here we ignore the communication overhead for the set of identities of revoked users, as these identities of revoked users can be picked from a small finite field [12].

3.2 Construction 2: Revocation Using Secret Sharing

– *Setup*: Let t be a positive integer. The group manager GM chooses independently and uniformly at random m polynomials $f_1(x), \ldots, f_m(x) \in F_q[x]$, each of degree t. The group manager randomly picks two initial key seeds, the forward key seed $S^F \in F_q$ and the backward key seed $S^B \in F_q$. It repeatedly applies (in the pre-processing time) the one-way function $\mathcal{H}$ on S^B and computes the one-way backward key chain of length m:

$$K_i^B = \mathcal{H}(K_{i-1}^B) = \mathcal{H}^{i-1}(S^B) \text{ for } 1 \leq i \leq m.$$

The j-th session key is computed as

$$\mathsf{SK}_j = K_j^F + K_{m-j+1}^B,$$

where $K_j^F = \mathcal{H}^{j-1}(S^F)$.

Each user U_i, for $1 \leq i \leq n$, receives its personal secret keys corresponding to the m sessions $S_i = \{f_1(i), \ldots, f_m(i)\}$ and the forward key seed S_F from the group manager via the secure communication channel between them.

– *Broadcast*: Let R_j be the set of all revoked users for sessions in and before j such that $|R_j| \leq t$ and G_j be the set of all non-revoked users in session j. In the j-th session the GM first chooses a set of indices (different from 0) $W_j = \{x_{1,j}, \ldots, x_{t,j}\}$ such that $I_{R_j} \subseteq W_j$, but $W_j \cap I_{G_j} = \emptyset$, where I_{R_j} represents the indices of the users in R_j, I_{G_j} denotes the set of indices of users in G_j and $\emptyset$ is the empty set. The GM then computes $Z_j = K_{m-j+1}^B + f_j(0)$ and broadcasts the following message $\mathcal{B}_j$:

$$\mathcal{B}_j = \{x_{1,j}, \ldots, x_{t,j}; f_j(x_{1,j}), \ldots, f_j(x_{t,j}); Z_j\}.$$

– *Session Key Recovery and Message Recovery*: When a non-revoked user U_i receives the j-th session key distribution message $\mathcal{B}_j$, it interpolates $\{(x_{l,j}, f_j(x_{l,j})\}_{l=1,\ldots,t}$ and $(i, f_j(i))$ to recover $f_j(0)$ by Lagrange's interpolation formula as follows:

$$f_j(0) = \sum_{l=0}^{t} \Lambda_l f_j(x_{l,j}),$$

where

$$\Lambda_l = \prod_{\substack{k=0 \\ k\neq l}}^{t} \frac{-x_{k,j}}{x_{l,j} - x_{k,j}}$$

with $x_{0,j} = i$. Then U_i recovers the key K_{m-j+1}^B as

$$K_{m-j+1}^B = Z_j - f_j(0).$$

Finally, U_i computes the j-th forward key $K_j^F = \mathcal{H}^{j-1}(S^F)$ and evaluates the current session key

$$\mathsf{SK}_j = K_j^F + K_{m-j+1}^B.$$

- *Add Group Members*: When the group manager adds a new group member starting from session j, it picks an unused identity $v \in F_q$, computes the personal secret keys corresponding to the current and future sessions $S_v = \{f_j(v), f_{j+1}(v), \dots, f_m(v)\}$ and gives $\{v, S_v, K_j^F\}$ to this new group member via the secure communication channel between them.
- *Re-initialization:* The system fails when all m sessions are exhausted, or when number of revoked users becomes more than t. At this phase, re-initialization is required and a new setup is executed.

Complexity:

Storage overhead: Storage complexity of personal key for each user is $m \log q$ bits. The group members that join later need to store less data. Foe example, the personal key for a user joining in j-th session occupies $(m - j + 1) \log q$ bits memory space.

Communication overhead: Communication bandwidth for key management is $(t + 1) \log q$ bits. Here we ignore the communication overhead for the broadcast of points $x_{l,j}$ for $l = 1, \dots, t$, as these identities can be picked from a small finite field.

3.3 Self-healing

We now explain our self-healing mechanism in the above constructions: Let U_i be a group member that receives session key distribution messages $\mathcal{B}_{j_1}$ and $\mathcal{B}_{j_2}$ in sessions j_1 and j_2 respectively, where $1 \le j_1 \le j_2$, but not the session key distribution message $\mathcal{B}_j$ for session j, where $j_1 < j < j_2$. User U_i can still recover all the lost session keys K_j for $j_1 < j < j_2$ as follows:

(a) U_i recovers from the broadcast message $\mathcal{B}_{j_2}$ in session j_2, the backward key $K_{m-j_2+1}^B$ and repeatedly apply the one-way function $\mathcal{H}$ on this and computes the backward keys K_{m-j+1}^B for all j, $j_1 \le j < j_2$.

(b) U_i computes the forward keys K_j^F for all j, $j_1 \le j \le j_2$ by repeatedly applying $\mathcal{H}$ on the forward seed S^F or on the forward key $K_{j_1}^F$ of the j_1-th session.

(c) U_i then recovers all the session keys $\mathsf{SK}_j = K_j^F + K_{m-j+1}^B$, for $j_1 \le j \le j_2$.

Note that a user revoked in session j cannot compute the backward keys $K_{m-j_1+1}^B$ for $j_1 > j$, although it can compute the forward keys $K_{j_1}^F$. As a result, revoked users cannot compute the subsequent session keys SK_{j_1} for $j_1 > j$, as desired.

Similarly, a user U_i joined in session j cannot compute the forward keys $K_{j_2}^F$ for $j_2 < j$ as U_i knows only the j-th forward key K_j^F, not the initial forward seed value S^F, although it can compute the backward keys $K^B_{m-j_2+1}$ for $j_2 < j$. This forbids U_i to compute the previous session keys as desired.

4 Security Analysis

In this section we show that our Constructions realizes self-healing key distribution schemes with revocation capability. More precisely, we can prove Theorem 4.1 and Theorem 4.2 which state the security result of Construction 1 and Construction 2 respectively in our security model described in Section 2.1

Theorem 4.1 *Construction 1 is secure, self-healing session key distribution scheme with privacy, t-revocation capability with respect to Definition 2.1 and achieve t-wise forward and backward secrecy with respect to Definition 2.2.*

Proof: Our goal is security against coalition of size at least t. We will show that the Construction 1 is computationally secure with respect to revoked users assuming the difficulty of inverting one-way function, *i.e.* for any session j it is computationally infeasible for any set of revoked users of size at most t before and on session j to compute with non-negligible probability the session key SK_j, given the View consisting of personal keys of revoked users, broadcast messages before, on and after session j and session keys of revoked users before session j.

Consider a coalition of t revoked users, say $U_1, U_2, \dots U_t$, who are revoked on or before the j-th session. The revoked users are not entitled to know the j-th session key SK_j. We can model this coalition of t users as a polynomial-time algorithm $\mathcal{A}'$ that takes View as input and outputs its guess for SK_j.We say that $\mathcal{A}'$ is successful in breaking the construction if it has a non-negligible advantage in determining the session key SK_j. Then using $\mathcal{A}'$, we can construct a polynomial-time algorithm $\mathcal{A}$ for inverting one-way function $\mathcal{H}$ and have the following claim:

Claim: $\mathcal{A}$ inverts one-way function $\mathcal{H}$ with non-negligible probability if $\mathcal{A}'$ is successful.

Proof: Given any instance $y = \mathcal{H}(x)$ of one-way function $\mathcal{H}$, $\mathcal{A}$ first generates an instance View for $\mathcal{A}'$ as follows: $\mathcal{A}$ randomly selects a forward key seed $S^F \in F_q$ and constructs the following backward key chain by repeatedly applying $\mathcal{H}$ on y:

$$K^B_1 = y, K^B_2 = \mathcal{H}(y), K^B_3 = \mathcal{H}^2(y), \dots, K^B_j = \mathcal{H}^{j-1}(y), \dots, K^B_m = \mathcal{H}^{m-1}(y).$$

$\mathcal{A}$ computes the j-th forward key $K^F_j = \mathcal{H}^{j-1}(S^F)$ and sets the j-th session key

$$\mathsf{SK}_j = K^F_j + K^B_{m-j+1}.$$

$\mathcal{A}$ chooses at random m polynomials $f_1(x), \dots, f_m(x) \in F_q[x]$, each of degree $t < n$. Each user U_i, for $1 \le i \le n$, receives its personal secret keys corresponding to the m sessions $S_i = \{f_1(i), \dots, f_m(i)\}$ and the forward key seed S_F from $\mathcal{A}$ via the secure communication channel between them. For $1 \le j \le m$, $\mathcal{A}$ computes broadcast message $\mathcal{B}_j$ as:

$$\mathcal{B}_j = R_j \cup \{h_j(x)\}$$

where $R_j = \{U_1, \ldots, U_t\}$ is the set of revoked users for sessions in and before j such that $R_{j-1} \subseteq R_j$ for $2 \leq j \leq m$, and

$$h_j(x) = K^B_{m-j+1} r_j(x) + f_j(x), \text{ with } r_j(x) = (x-1)\cdots(x-t).$$

Then $\mathcal{A}$ sets View as

$$\mathsf{View} = \left\{ \begin{array}{l} f_j(1), \ldots, f_j(t) \text{ for } j = 1, \ldots, m, \\ \mathcal{B}_j \text{ for } j = 1, \ldots, m, \\ S^F, \\ \mathsf{SK}_1, \ldots, \mathsf{SK}_{j-1} \end{array} \right\}$$

$\mathcal{A}$ gives View to $\mathcal{A}'$, which in turn selects $X \in F_q$ randomly, sets the j-th session key to be $\mathsf{SK}'_j = K^F_j + X$ and returns SK'_j to $\mathcal{A}$. $\mathcal{A}$ checks whether $\mathsf{SK}'_j = \mathsf{SK}_j$. If not, $\mathcal{A}$ chooses a random $x' \in F_q$ and outputs x'.

$\mathcal{A}'$ can compute the j-th forward key $K^F_j = \mathcal{H}(S^F)$ as it knows S^F from View for $j = 1, \ldots, m$. Note that from View, $\mathcal{A}'$ knows at most t points on the t-degree polynomial $f_j(x)$ and at most $j-1$ session keys $\mathsf{SK}_1, \ldots, \mathsf{SK}_{j-1}$. Consequently $\mathcal{A}'$ has knowledge of at most $j-1$ backward keys $K^B_m, \ldots, K^B_{m-j+2}$. Observe that $\mathsf{SK}'_j = \mathsf{SK}_j$ provided $\mathcal{A}'$ knows the backward key K^B_{m-j+1}. This occurs if either of the following two holds:

(a) $\mathcal{A}'$ is able to compute the t-degree polynomial $f_j(x)$ from View and consequently can recover the backward key K^B_{m-j+1} as follows:

$$K^B_{m-j+1} = \frac{h_j(i) - f_j(i)}{r_j(i)},$$

where $i \neq 1, \ldots, t$. Note that $r_j(i) = 0$ for $i = 1, \ldots, t$.

From View, $\mathcal{A}'$ knows only t-points on the t-degree polynomial $f_j(x)$ and will not be able to compute $f_j(x)$. Consequently, $\mathcal{A}'$ will not be able to recover K_{m-j+1} from $\mathcal{B}_j$ as described in (a) above.

(b) $\mathcal{A}'$ is able to choose $X \in F_q$ so that the following relations hold:

$$K^B_m = \mathcal{H}^{j-1}(X), K^B_{m-1} = \mathcal{H}^{j-2}(X), \ldots, K^B_{m-j+2} = \mathcal{H}(X)$$

This occurs with a non-negligible probability only if $\mathcal{A}$ is able to invert the one-way function $\mathcal{H}$. In that case, $\mathcal{A}$ returns $x = \mathcal{H}^{-1}(y)$.

The above arguments show that if $\mathcal{A}'$ is successful in breaking the security of Construction 1, then $\mathcal{A}$ is able to invert the one-way function. □
(of claim)

Hence Construction 1 is computationally secure under the hardness of inverting one-way function. We will now show that Construction 1 satisfies all the conditions required by Definition 2.1.

1) (a) Session key efficiently recovered by a non-revoked user U_i is described in the third step of our Construction 1.

(b) For any set $R \subseteq \mathcal{U}$, $|R| \leq t$, and any non-revoked user $U_i \notin R$, we show that the coalition R knows nothing about the personal secret S_i of U_i. For any session j, U_i's personal secret $S_i = f_j(i)$ is a point over a t-degree polynomial $f_j(x)$. Since the coalition R gets at most t points over the t-degree polynomial $f_j(x)$, it is computationally infeasible for coalition R to learn $f_j(i)$ for $U_i \notin R$.
(c) The j-th session key $\mathsf{SK}_j = K_j^F + K_{m-j+1}^B$, where $K_j^F = \mathcal{H}(K_{j-1}^F) = \mathcal{H}^{j-1}(S^F)$, $K_j^B = \mathcal{H}(K_{j-1}^B) = \mathcal{H}^{j-1}(S^B)$, S^F is the forward seed value given to all initial group members and S^B is the secret backward seed value. Thus $\mathsf{SK_j}$ is independent of the personal secret $S_i = f_j(i)$ for $i = 1, \ldots, n$. So the personal secret keys alone do not give any information about any session key. Since the initial backward seed S^B is chosen randomly, the backward key K_{m-j+1}^B and consequently the session key SK_j is random as long as S^B, $K_1^B, K_2^B, \ldots, K_{m-j+2}^B$ are not get revealed. This in turn implies that the broadcast messages alone cannot leak any information about the session keys. So it is computationally infeasible to determine $Z_{i,j}$ from only personal key S_i or broadcast message $\mathcal{B}_j$.

2) (t-revocation property) Let R be a collection of t-revoked users collude in session j. It is impossible for coalition R to learn the j-th session key SK_j because knowledge of SK_j implies the knowledge of either the backward key K_{m-j+1}^B or the knowledge of the personal secret $f_j(i)$ of user $U_i \notin R$. The coalition R knows the points $\{f_j(i) : U_i \in R\}$. The size of the coalition R is at most t. Consequently, the colluding users only have at most t-points on the polynomial $f_j(x)$. But degree of the polynomial $f_j(x)$ is t. Hence the coalition R cannot recover $f_j(x)$, which in turn makes K_{m-j+1}^B appears random to R. Therefore, SK_j is completely safe to R from computation point of view.
3) (Self-healing property) From the third step of our Construction 1, any user U_i that is a member in sessions j_1 and j_2 ($1 \leq j_1 < j_2$), can recover the backward key $K_{m-j_2+1}^B$ and hence can obtain the sequence of backward keys $K_{m-j_1}^B, \ldots, K_{m-j_2+2}^B$ by repeatedly applying $\mathcal{H}$ on $K_{m-j_2+1}^B$. User U_i also holds the forward key $K_{j_1}^F = \mathcal{H}^{j_1-1}(S^F)$ of the j_1-th session and hence can obtain the sequence of forward keys $K_{j_1+1}^F, \ldots, K_{j_2-1}^F$ by repeatedly applying $\mathcal{H}$ on $K_{j_1}^F$. Hence, as shown in Section 3.3, user U_i can efficiently recover all missed session keys.

We will show the Construction 1 satisfies all the conditions required by Definition 2.2.

1) (t-wise forward secrecy) Let $R \subseteq \mathcal{U}$, where $|R| \leq t$ and all user $U_l \in R$ are revoked before the current session j. The coalition R can not get any information about the current session key SK_j even with the knowledge of group keys before session j. This is because of the fact that in order to know SK_j, $U_l \in R$ needs to know at least $t+1$ points on the polynomial $f_j(x)$. Since

size of the coalition R is at most t, the coalition R has at most t personal secrets $f_j(i)$, *i.e.* gets t points on the polynomial $f_j(x)$. But at least $t+1$ points are needed on the polynomial $f_j(x)$ to recover the current session key SK_j for any user $U_l \in R$. Besides, because of the one-way property of $\mathcal{H}$, it is computationally infeasible to compute $K^B_{j_1}$ from $K^B_{j_2}$ for $j_1 < j_2$. The users in R might know the sequence of backward keys $K^B_m, \ldots, K^B_{m-j+2}$, but cannot compute K^B_{m-j+1} and consequently SK_j from this sequence. Hence the Construction 1 is t-wise forward secure.

2) (t-wise backward secrecy) Let $J \subseteq \mathcal{U}$, where $|J| \leq t$ and all user $U_l \in J$ join after the current session j. The coalition J can not get any information about any previous session key SK_{j_1} for $j_1 \leq j$ even with the knowledge of group keys after session j. This is because of the fact that in order to know SK_{j_1}, $U_l \in J$ requires the knowledge of j_1-th forward key $K^F_{j_1} = \mathcal{H}(K^F_{j_1-1}) = \mathcal{H}^{j_1-1}(S^F)$. Now when a new member U_v joins the group starting from session $j+1$, the GM gives $(j+1)$-th forward key K^F_{j+1} instead of the initial forward key seed S^F, together with the values $f_{j+1}(v), \ldots, f_m(v)$. Note that $K^F_{j+1} = \mathcal{H}(K^F_j)$. Hence it is computationally infeasible for the newly joint member to trace back for previous forward keys $K^F_{j_1}$ for $j_1 \leq j$ because of the one-way property of the function $\mathcal{H}$. Consequently, our protocol is t-wise backward secure. In fact, this backward secrecy is independent of t. □

A similar result holds for our Construction 2 and we can prove Theorem 4.2 stated below following the same line of proving Theorem 4.1.

Theorem 4.2 *Construction 2 is secure, self-healing session key distribution scheme with privacy, t-revocation capability with respect to Definition 2.1 and achieves t-wise forward and backward secrecy with respect to Definition 2.2.*

5 Performance Analysis

Comparison of storage overhead, communication complexity and computation cost of each user (not the GM) in our constructions with the existing self-healing session key distribution schemes is provided in Table 1 (see Introduction). It is demonstrated in Table 1 that our proposed constructions are more efficient than the previous schemes. In particular, our Construction 1 is the most efficient key distribution scheme with self-healing and revocation property among all the previous approaches. In one hand our constructions reduce the communication complexity (bandwidth) to $O(t)$, whereas optimal communication complexity achieved by the previous schemes is $O(tj)$ at the j-th session. Achieving less computation cost is on the other side of the coin. For a user U_i at the j-th session, the computation cost is incurred by recovering all previous session keys upto the j-th session (worst case) by self-healing mechanism. The backward key used at the j-th session in our Construction 1 is $K^B_{m-j+1} = \frac{h_j(i)-f_j(i)}{r_j(i)}$.

Consequently, user U_i needs to computes two points $h_j(i), r_j(i)$ on the polynomials $h_j(x)$ and $r_j(x)$ which require at most $2t$ multiplication operations. Since division can be regarded as multiplication, total number of multiplication operations required to get K^B_{m-j+1} is $2t + 1$. After obtaining K^B_{m-j+1}, user U_i can easily compute $K^B_{m-j+2}, K^B_{m-j+3}, \ldots, K^B_{m-1}, K^B_m$ by applying the one-way function $\mathcal{H}$ each time. Then U_i is able to compute all previous session keys $\mathsf{SK}_{j_1} = K^F_{j_1} + K^B_{m-j_1+1}$ for all $1 \leq j_1 \leq j$. Thus the computation cost for each user is $2t+1$, whereas the computation complexity of the revocation polynomial based scheme in [12] is $j(2t + 1)$. Similarly, for Construction 2 the computation complexity is $2\{(t + 1)^2 - (t + 1)\} = 2(t^2 + t)$ which is the number of multiplication operations needed to recover a t degree polynomial by using Lagrange formulation. Thus the communication complexity and computation cost in our constructions do not increase as the number of session grows. These are the most prominent improvement of our schemes over the previous works. The storage overhead of each user for personal key in both our self-healing key distribution schemes is $O((m - j + 1) \log q)$, which is same as that of [4, 12].

Remark and Future Work: Our security model excludes the following property of self-healing key distribution unlike the security model provided by [14, 22]: Let $1 \leq j_1 < j < j_2 \leq m$. For any disjoint subsets $L_1, L_2 \subset \mathcal{U}$, where $|L_1 \cup L_2| \leq t$, no information about the session key SK_j, $j_1 < j < j_2$ can be obtained by the coalition $L_1 \cup L_2$, where the set L_1 is a coalition of users removed before session j_1 and the set L_2 is a coalition of users joined from session j_2. Our protocol does not satisfy this property as illustrated by the following simple example: Let $L_1 = \{U_3\}, L_2 = \{U_6\}$ and $j_1 = 2, j_2 = 5$. The above property states that U_3 and U_6 jointly should not be able to know SK_j, $j = 3, 4$. But U_3 knows K^F_2 and U_6 knows K^B_{m-5+1}. Consequently, U_2 can compute K^F_3, K^F_4 and U_6 can compute K^B_{m-4+1}, K^B_{m-3+1}. Hence, U_3 and U_6 together can compute $\mathsf{SK}_j = K^F_j + K^B_{m-j+1}, j = 3, 4$. As a future work we are interested to incorporate this property in our scheme.

6 Conclusion

In this paper, we develop and analyze two efficient computationally secure self-healing key distribution schemes with revocation capability, enabling a very large and dynamic group of users to establish a common key for secure communication over an insecure wireless network. We introduce a novel self-healing mechanism for session key-recovery on possible packet lost in the lossy environment using one-way key chain. Our proposed key distribution mechanism significantly improves the communication and computation costs over the previous approaches without any increase in the storage complexity. The schemes are properly analyzed in an appropriate security model to prove that they are computationally secure and achieve both forward secrecy and backward secrecy.

References

[1] S. Berkovit. *How to Broadcast a Secret.* Advances in Cryptology, Eurocrypt'91, LNCS 547, pp. 536-541, Springer-Verlag, 1991.

[2] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung. *Perfectly-Secure Key Distribution for Dynamic Conferences.* In Crypto'92, LNCS 740, pp. 471-486, Springer-Verlag, 1993.

[3] C. Blundo, L. F. Mattos, D. Stinson. *Trade-offs between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution.* Advances in Cryptology, Crypto'96, LNCS 1109, pp. 387-400, Springer-Verlag, 1996.

[4] C. Blundo, P. D'Arco, A. Santis, M. Listo. *Design of Self-healing Key Distribution Schemes.* Design Codes and Cryptology, N. 32, pp. 15-44, 2004.

[5] C. Blundo, P. D'Arco, A. Santis, M. Listo. *Definitions and Bounds for Self-healing Key Distribution.* Fuzzy Sets and Systems, 31st International Colloquium on Automata, Languages and Programming ICALP 04, LNCS 3142, pp. 234-245, Springer-Verlag, 2004.

[6] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas. *Multicast Security: A Taxonomy and Some Efficient Constructions.* In IEEE INFOCOMM'99, 1999.

[7] R. Canetti, T. Malkin, K. Nissim. *Efficient Communication-Storage Tradeoffs for Multicast Encryption.* Advances of Cryptology - Eurocrypt'99, LNCS 1592, pp. 459-474, Springer-Verlag, 1999.

[8] T. M. Cover, J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[9] R. Dutta, S. Mukhopadhyay. *Improved Self-Healing Key Distribution with Revocation in Wireless Sensor Network.* In the proceeding of the IEEE Wireless Communications and Networking Conference (WCNC 2007), will be held in Hong Kong, China, 2007 (to appear).

[10] A. Fiat, M. Naor. *Broadcast Encryption.* In Crypto'93, LNCS 773, pp. 480-491, Springer-Verlag, 1994.

[11] L. Gong. *New Protocols for Third-Party-Based Authentication and Secure Broadcast.* In Proceedings of ACM CCS, 1994.

[12] D. Hong, J. Kang. *An Efficient Key Distribution Scheme with Self-healing Property.* IEEE Communication Letters'05, Vol. 9, pp. 759-761, 2005.

[13] M. Just, E. Kranakis, D. Krizanc, P. van Oorschot. *On Key Distribution via True Broadcasting.* In Proceedings of ACM CCS, 1994.

[14] D. Liu, P. Ning, K. Sun. *Efficient Self-healing Key Distribution with Revocation Capability.* Proceedings of the 10th ACM CCS'03, pp. 27-31, 2003.

[15] D. McGrew, A. Sherman. *Key Establishment in large dynamic groups using one-way function trees.* TIS Report No. 0755, 1998.

[16] S. More, M. Malkin, J. Staddon. *Sliding-window Self-healing Key Distribution with Revocation.* ACM Workshop on Survivable and Self-regenerative Systems'03, pp. 82-90, 2003.

[17] D. Naor, M. Naor, J. Lotspiech. *Revocation and Tracing Schemes for Stateless Users.* Advances of Cryptology - Crypto'01, LNCS 2139, pp. 41-62, Springer-Verlag, 2001.

[18] A. Perrig, D. Song, J. D. Tygar. *ELK, a New Protocol for Efficient Large-Group Key Distribution.* Proceedings of IEEE Symposium on Security and Privacy'01, pp. 247-262, 2001.

[19] G. Saez. *On Threshold Self-healing Key Distribution Schemes.* Cryptography and Coding'04, LNCS 3796, pp. 340-354, Springer-Verlag, 2004.

[20] R. Safavi-Naini, H. Wang. *New Constructions of Secure Multicast Re-Keying Schemes using Perfect Hash Families.* In Proceedings of ACM CCS'00, pp. 228-234, 2000.

[21] S. Setia, S. Koussih, S. Jajodia. *Korons: A Scalable Group Re-Keying Approach for Secure Multicast.* In Proceedings of IEEE Symp. on Security and Privacy.

[22] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, D. Dean. *Self-healing key distribution with Revocation.* Proceedings of IEEE Symposium on Security and Privacy'02, pp. 224-240, 2002.

[23] D. R. Stinson. *On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption.* Designs, Codes and Cryptology, vol. 12, pp.215-243, 1997.

[24] D. R. Stinson, T. van Trung. *Some New Results on Key Distribution Patterns and Broadcast Encryption.* Designs, Codes and Cryptography, vol. 14, pp. 261-279, 1998.

[25] C. Wong, M. Gouda, S. Lam. *Secure Group Communications using Key Graphs.* In Proceedings of ACM SIGCOMM'98, pp. 68-79, 1998.

[26] S. Xu. *On the security of group communication schemes.* In the Journal of Computer Security, Volume 15, Number 1, 2007, pp. 129 - 169, 2007.

[27] Y. R. Yang, X. S. Li, X. B. Zhang, S. S. Lam. *Reliable Group Re-Keying: A Performance Analysis.* In ACM SIGCOMM'01, pp. 27-38, 2001.

BAP: Broadcast Authentication Using Cryptographic Puzzles*

Patrick Schaller, Srdjan Čapkun, and David Basin

Computer Science Department, ETH Zurich
ETH Zentrum, CH-8092 Zurich, Switzerland
{patrick.schaller,srdjan.capkun,david.basin}@inf.ethz.ch

Abstract. We present two broadcast authentication protocols based on delayed key disclosure. Our protocols rely on symmetric-key cryptographic primitives and use cryptographic puzzles to provide efficient broadcast authentication in different application scenarios, including those with resource-constrained wireless devices such as sensor nodes. The strong points of the protocols proposed are that one protocol allows instantaneous message origin authentication, whereas the other has low communication overhead. In addition to formalizing and analyzing these specific protocols, we carry out a general analysis of broadcast authentication protocols based on delayed key disclosure. This analysis uncovers fundamental limitations of this class of protocols in terms of the required accuracy of message propagation time estimations, if the protocols are to guarantee security and run efficiently.

1 Introduction

Recent research in broadcast authentication for wireless networks [26,25,31,22,21] addresses the question of how to develop efficient mechanisms and protocols for broadcast message authentication in networks of low-cost and resource-constrained wireless devices (e.g., sensor networks). The main challenge here concerns efficiency: reducing the cost of message generation by the sender and the verification of message authenticity by the receiver. The approaches proposed include the use of symmetric-key primitives and delayed key disclosure [26,27,16,8], one-time signatures [25,6], and solutions based on devices' awareness of presence in the vicinity of the sender [31].

In this work, we propose two new broadcast authentication protocols based on delayed key disclosure. Our protocols are based on symmetric-key cryptographic primitives and rely on cryptographic puzzles to provide efficient broadcast authentication in a wide range of application scenarios, including those with resource-constrained wireless devices such as sensor nodes. The first protocol (BAP-1) achieves instantaneous message-origin authentication upon message

* The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

reception. Our second protocol (BAP-2) achieves broadcast authentication using a single transmission per authenticated message.

Similar to previously proposed broadcast authentication protocols based on delayed key disclosure [26], we also use authenticated keys derived from one-way (hash) chain elements. However, instead of relying on delayed key transmission (and consequently, delayed message verification), we use cryptographic puzzles to hide the key up to the point in time when the key can be safely disclosed. In our first protocol (BAP-1), the key is sent in a puzzle before the message, thereby allowing instantaneous message verification. This allows the message to be verified upon reception, assuming that the puzzle is solved by the time the entire message is received. Our second protocol (BAP-2) achieves broadcast authentication with delayed key disclosure by transmitting a single message containing the original message, its message authentication code (MAC), and the key. BAP-2 therefore reduces the communication overhead in terms of the number of messages needed for message authentication. We provide a detailed security analysis of both protocols and use this analysis to highlight applications where each of these protocols is suitable.

In addition to proposing and analyzing our protocols, we carry out a general analysis of broadcast authentication protocols based on delayed key disclosure. This analysis uncovers fundamental limitations of this class of protocols in terms of the required accuracy of message propagation time estimations and of time synchronization, if the protocol is to guarantee security and run efficiently. More specifically, our analysis shows that, if a protocol is to work both securely and efficiently, the message propagation times in the network must be known in advance. This requirement limits the applicability of this class of protocols. However, the design of these protocols makes them well suited for networks with known (or predictable) topologies and for delay-tolerant networks of low-cost and resource-scarce devices.

In summary, we make the following contributions in this work. First, we propose two new protocols for broadcast authentication based on delayed key disclosure. These proposed protocols represent two new points in the security-performance subspace of this class of protocols. Second, we analyze the class of broadcast authentication protocols based on delayed key disclosure, where we highlight the importance of the accurate estimation of message propagation times for the performance of protocols in this class.

The rest of the paper is organized as follows. In Section 2, we state the problem and describe our system and attacker model. In Section 3, we describe our broadcast authentication protocols. In Section 4, we present our analysis of broadcast authentication protocols based on delayed key disclosure. In Section 5, we survey related work and we conclude the paper in Section 6.

2 Problem Statement and Background

The objective of *broadcast authentication* is to guarantee message-origin authentication and hence also the integrity of messages transmitted by a sender to the receivers. That is, all receivers in the network can verify that each broadcasted message has been generated by the claimed source and that it has not been modified in transmission. This problem is generally solved using asymmetric cryptographic primitives such as digital signatures [28,11], where the sender signs a message with his private key and broadcasts the signature along with the message. Any receiver holding the sender's public key can verify the correctness of the signature and validate if the message was indeed generated by the claimed sender.

Achieving efficient broadcast authentication using symmetric-key primitives is more challenging. One naive solution is that the sender appends message authentication codes (MACs) to the messages, generated with the keys that the sender shares with the receivers (one MAC per receiver). This solution clearly does not scale and adds substantial overhead to the network, especially in the case of multi-hop wireless networks. Another solution is that the sender shares a single key with all receivers, in which case a single MAC is sufficient to authenticate the sender. The downside of this solution is that a single compromised node is sufficient to compromise the whole scheme.

In this work, our goal is to provide a scalable and efficient broadcast authentication protocol for resource-constrained devices using symmetric-key primitives. We now provide a definition of broadcast authentication, incorporating both message-origin authentication and a parameterized notion of recentness.

Definition 1

i) *A broadcast protocol guarantees (message-origin) authentication iff whenever a node B receives a message m and concludes that it was sent by node A, then m was indeed sent by A.*
ii) *A broadcast protocol guarantees T-recentness iff whenever a node B receives a message m and concludes that it was sent within T time units before its reception, then m was indeed sent within this time interval.*
iii) *A broadcast protocol guarantees T-authentication iff it guarantees both authentication and T-recentness.*

2.1 System and Attacker Model

We now describe the class of systems we consider. A system consists of a collection of nodes connected via (e.g., wireless) communication links. The nodes can be connected directly or can communicate over multiple hops. We neither impose restrictions on the network topology nor do we assume that the network nodes are aware of the network topology or of their respective locations. The network is operated by an authority. This authority can be on-line, meaning that the authority operates on-line servers (that can be reached by single-hop or multi-hop communication), or off-line, meaning that the services of the authority cannot be reached over the network. We assume that all network nodes can establish pairwise secret keys. This can be achieved by manually pre-loading all keys onto the nodes in a network setup phase, using probabilistic key pre-distribution protocols [12,5], or through an on-line key distribution center [17]. We also assume that every network node holds authentic (public) commitments of hash chains generated by other network nodes. These commitments are distributed by the network authority prior to network operation. We discuss this further in Section 2.2. Finally, any network node is a potential broadcast message source and all other nodes are designated message receivers.

We adopt the following attacker model. We assume that the attacker Mallory (M) controls the communication channel in the sense that he can insert, eavesdrop, delay, schedule, modify, or block transmitted messages. Additionally the attacker can send messages ahead of time, if he can predict them. We assume that the attacker is not a part of the network controlled by the authority and cannot gain access to network keys or disclose any messages exchanged between the nodes or between the nodes and the authority in the setup phase. We also assume that network nodes can be compromised by the attacker. However, we will assume that the sender participating in the broadcast authentication protocol is not compromised, as then broadcast authentication would be meaningless. Finally, we require M to be *computationally bounded*; specifically we assume that we can create cryptographic puzzles that are time-consuming for the attacker to solve (see Section 3.3).

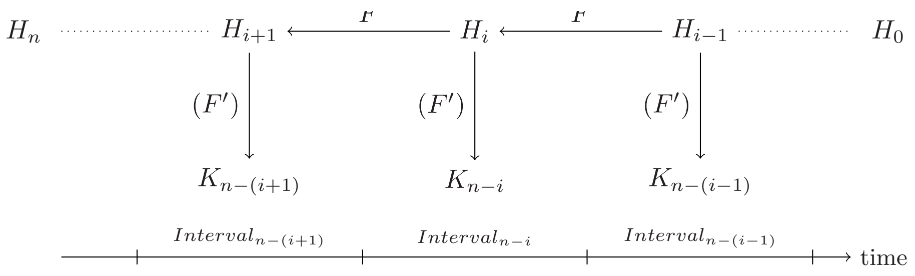
Before presenting our solutions to the broadcast authentication problem, we briefly describe authentication based on one-way chains using delayed key disclosure.

2.2 Authentication Using One-Way-Chains and Delayed Key Disclosure

The use of hash-chains for authentication was first introduced by Lamport in [19]. Hauser et al. used hash-chains in [16] to authenticate routing updates in routing protocols by assigning the elements of a hash-chain to points in time. Cheung [8] added the notion of late key disclosure, which was then used by Perrig et al. for broadcast authentication in TESLA [26].

The basic idea is as follows: The sender (whose messages should be authenticated) creates a hash-chain by selecting a random element H_0 as the root and by iteratively applying to it a one-way (pre-image-resistant) function F. This produces the sequence H_0, H_1, ..., H_n, where $H_i = F^i(H_0)$, for $1 \leq i \leq n$. As F is one-way, a receiving node possessing H_i cannot feasibly compute the predecessor H_{i-1}; only the owner of the root can do so, by computing forward from H_0. However, given a string s, any node possessing H_i can easily check if $s = H_{i-1}$ by checking if $F(s) = H_i$. The sender then commits to the hash chain by distributing H_n in an authentic way to each receiver. Moreover, if required, the receiver synchronizes his clock with the sender's at this point.

We associate hash values with keys in the following way. For each H_i, we apply another one-way function F' to derive a key K_{n-i}, for the corresponding time interval $n-i$. F' is used to avoid using the string H_i for two different purposes: as a hash value in the chain and as a key. In the following, we will use the chain element and the corresponding key interchangeably, since this does not affect our observations.



To authenticate a messages m, the sender assigns the message to a time interval. Then, to send m in the $(n-i)$th time interval, the sender appends to m a keyed MAC, $MAC_{K_{n-i}}(m)$, as well as the chain element for the preceding time interval, H_{i+1}, in clear text. This hash value opens the commitment to H_{i+1}, and hence the receiver can determine the key $K_{n-(i+1)}$ and thereby authenticate the previous message.

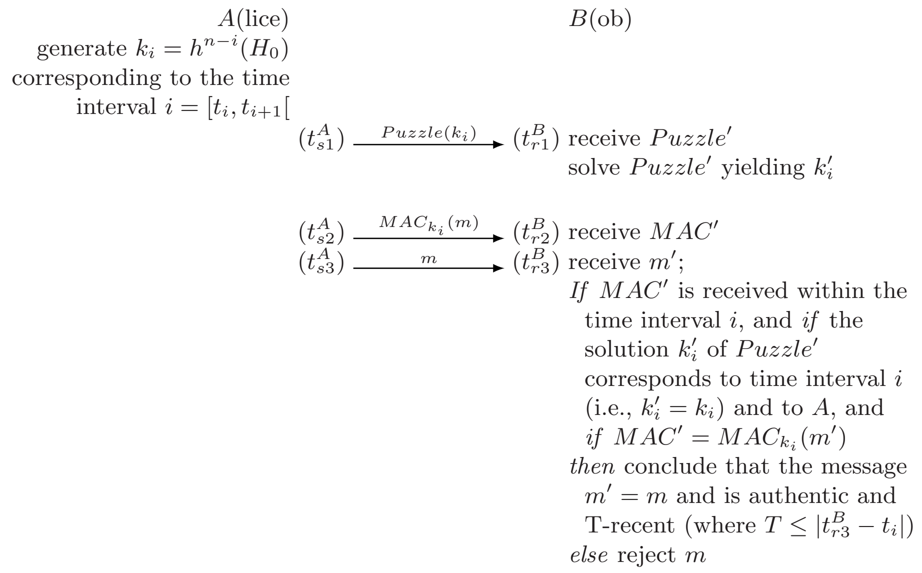
3 Broadcast Authentication Using Cryptographic Puzzles

In this section, we present our broadcast authentication protocols. We begin by stating our assumptions. We assume that the sender and the receivers have synchronized clocks. Recently, several proposals for (secure) time synchronization in wireless networks have emerged that successfully address this problem [13,23,30]. In wired networks, we assume that the nodes are securely synchronized using online synchronization servers or precise local clocks. In some application scenarios, we can also rely on nodes synchronizing their clocks using GPS [15] receivers. In Section 4, we will analyze the implications of time synchronization in more detail. We further assume that a broadcasting node (A) has generated a one-way (hash) chain and distributed its corresponding commitment to the designated receivers (B) in an authentic manner. As described in Section 2.2, the elements of this chain are used by the sender to derive message authentication keys, whereas the chain commitment value is used by the receivers to verify the authenticity of the used keys. Finally, we assume that the sender can create, and receivers can solve (within some given time), cryptographic puzzles. In Section 3.3, we discuss different puzzle schemes and their implications for the proposed broadcast authentication protocols.

We now present two broadcast authentication protocols based on cryptographic puzzles and delayed key disclosure, which we call BAP-1 and BAP-2.

3.1 BAP-1

BAP-1 is designed to achieve instantaneous message verification upon message receipt. The protocol is shown on Figure 1. In this protocol, the message sender first chooses the cryptographic key k_i, which corresponds to the time interval $i = [t_i, t_{i+1}[$ (where $[t_i, t_{i+1}[$ denotes the set $\{t \in \mathbb{R} | t_i \leq t < t_{i+1}\}$), according



A(lice)

generate $k_i = h^{n-i}(H_0)$ corresponding to the time interval $i = [t_i, t_{i+1}[$

B(ob)

$(t^A_{s1}) \xrightarrow{Puzzle(k_i)} (t^B_{r1})$ receive $Puzzle'$
solve $Puzzle'$ yielding k'_i

$(t^A_{s2}) \xrightarrow{MAC_{k_i}(m)} (t^B_{r2})$ receive MAC'

$(t^A_{s3}) \xrightarrow{m} (t^B_{r3})$ receive m';

If MAC' is received within the time interval i, and *if* the solution k'_i of $Puzzle'$ corresponds to time interval i (i.e., $k'_i = k_i$) and to A, and *if* $MAC' = MAC_{k_i}(m')$

then conclude that the message $m' = m$ and is authentic and T-recent (where $T \leq |t^B_{r3} - t_i|$)

else reject m

Fig. 1. BAP-1 protocol. The protocol achieves broadcast authentication through delayed key release based on cryptographic puzzles. Instant message authentication is achieved if the receiver solves the puzzle, and therefore obtains the key, before receiving the message. All messages received by B are marked with $'$ to denote that they might have been modified in transit by an attacker.

to the scheme described in Section 2.2. The sender then encapsulates k_i within a cryptographic puzzle $Puzzle(k_i)$, and broadcasts the puzzle at time t^A_{s1}. The puzzle serves to hide the key for a given time, which depends on the puzzle complexity and on the solver's processing speed. Immediately after the last bits of the puzzle have been sent (at t^A_{s2}), the sender starts transmitting the message authentication code $MAC_{k_i}(m)$, computed over the broadcast message m, using the key k_i contained in the puzzle. Finally, when the last bits of $MAC_{k_i}(m)$ are sent (at t^A_{s3}), the sender transmits the broadcast message m.

From the receiver's side, the protocol proceeds as follows: At time t^B_{r1}, the receiver B receives the puzzle $Puzzle'$ and starts solving it to retrieve the key k'_i. Here, all messages received by B are marked with $'$ to denote that they may have been modified in transit by the adversary. Concurrent to solving the puzzle, B receives MAC' and subsequently the message m'. In order to verify the authenticity of the message immediately upon its receipt, the receiver must solve the puzzle before receiving the last bits of the message (i.e., prior to t^B_{r3}). This case is optimal in terms of verification speed and we study it in more detail in Section 3.1. After the receiver solves the puzzle, he then verifies (i) if MAC' was received within the time interval i, (ii) if the key k'_i is indeed authentic and corresponds to the current time slot i and to the claimed sender A (i.e., if k'_i can be bound to the public commitment H_n previously distributed

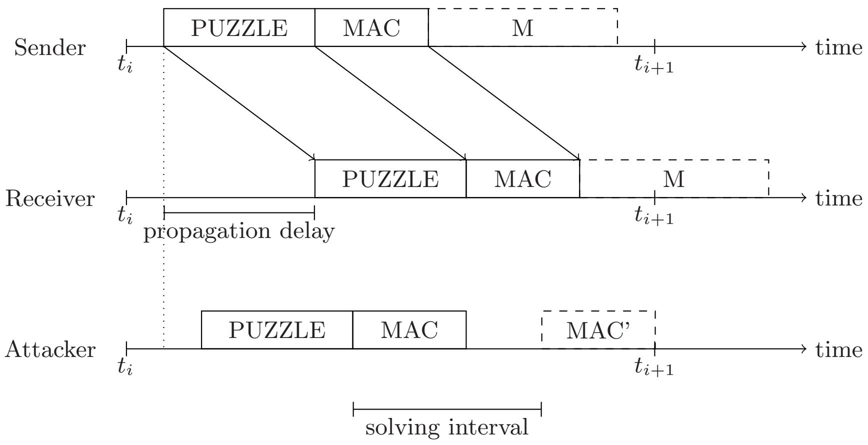


Fig. 2. The diagram shows the sender and a receiver of a broadcast message. At the bottom, the attacker receives the message earlier than the receiver. In order to create a valid, spoofed message, he must not only solve the puzzle and create a valid MAC (MAC'), he must also have it delivered to the receiver inside the validity window of the key.

by the sender and to its intended time release slot, see Section 2.2), and (iii) if the message authentication code $MAC_{k_i}(m')$ computed with the derived key over the received message equals the received authentication code MAC'. If all verifications succeed, then the receiver concludes that the message $m' = m$ is both authentic (i.e., generated by the claimed source A) and T-recent (i.e., has been sent by A within T time units before reception, where $T \leq |t_{r3}^{B} - t_i|$). Hence, the receiver concludes that the message is T-authentic according to Definition 1.

Security and Performance Analysis. From the previous protocol description, we can derive the main protocol performance condition: the BAP-1 protocol achieves optimal performance in terms of message verification time, if the receiver can solve the puzzle by the time that he receives the last bit of the message. If this condition is met, the receiver can verify the message immediately upon reception. The fulfillment of this condition depends on the receiver's processing speed (i.e., the speed that it can solve puzzles), the MAC and the message propagation delays (depending on the speed of the underlying communication channel and the network topology), and on the transmission time (depending on the bit rate and the size of the MAC and the message). Note that for the message verification to succeed, the MAC needs to reach the receiver within the key's validity window.

The main security condition of this protocol is that the attacker M is not able to solve the puzzle before the validity of the key expires (each key is valid for only a certain time interval, as described in Section 2.2). This prevents the attacker from being able to create a valid MAC for his own messages. Besides solving the puzzle and jamming the original MAC, the attacker has to create his own MAC and deliver it before the validity window ends. The relationship between the message propagation time, key validity window, and the minimum time within which the attacker needs to solve the puzzle is shown on Figure 2. The sender can enhance the security level of the protocol by creating puzzles that are harder (and thereby require more time) to solve or by tightening the key validity window. For a more detailed discussion on cryptographic puzzles and how they are constructed, see Section 3.3. Note that the receiver has much more time to solve the puzzle than the attacker. Namely, to achieve instant verification, the receiver needs to solve the puzzle before the message has been received, which will typically be after the validity period of the key has expired (depending on the message size). Alternatively, the receiver can continue solving the puzzle after he receives the message, in which case the message will be verified with a delay. We want to point out that a puzzle in combination with the hash-chain and the corresponding disclosure schedule protects the MAC (and therefore the integrity and authenticity of the corresponding message) during transportation up to the point of reception. Therefore the required restrictions on the computational resources of the adversary do not depend on the computational power of the honest nodes involved, but mainly depend on the transmission time given by the communication medium and the network topology.

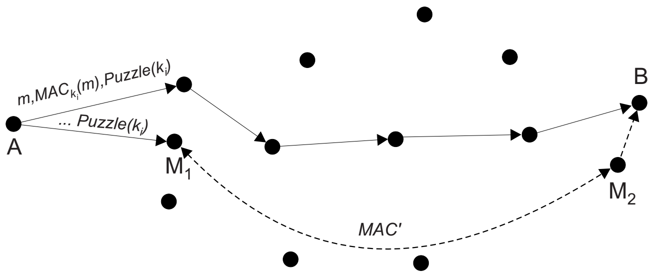


Fig. 3. This figure shows a scenario in which the attacker controls two devices (M_1 and M_2), M_1 located close to the sender (A), whereas M_2 is located close to the receiver (B). Furthermore the attacker nodes are connected by a fast link (dashed line).

From Figure 2, we can draw conclusions about how the attacker's physical distance from the sender and the receiver affects his ability to successfully break the scheme. Namely, if the attacker is located close to the sender, he will quickly obtain the puzzle (i.e., the propagation delay of the puzzle from the sender to the attacker will be short), and therefore can start solving it earlier. However, after solving the puzzle, the attacker still needs to forge a new message, create a new MAC', and send MAC' to the attacked receiver. Therefore being close to the sender (instead of to the receiver) means that the propagation time of MAC' from the attacker to the receiver will be long, thus reducing the attacker's ability to solve the puzzle and send MAC' before the key becomes invalid. This is illustrated in Figure 3, where the attacker M_1 eavesdrops on the puzzle sent by the sender and, after solving it, forges MAC', and sends it to the receiver (using multi-hop communication). Equally, if the attacker is placed close to the receiver, then he will get the puzzle at the same time as the receiver, but will have almost no time to solve it before the first bits of $MAC_{k_i}(m)$ are received by the receiver or before the key's validity expires. From these considerations we conclude that the optimal placement for the attacker is the one that minimizes the sum of the puzzle propagation delay from the sender to the attacker and the MAC' propagation delay from the attacker to the receiver. This means that the attacker is best located along the fastest (e.g., shortest) communication path between the sender and the receiver.

Note that, if the attacker controls two devices, M_1 and M_2, one located close to the sender and the other close to the receiver, connected with a fast (e.g., wired) link, then the attacker can shorten the communication time of MAC' to the receiver, and therefore gain time to solve the puzzle and obtain the key. This scenario is illustrated in Figure 3. However, if the broadcast source has a direct link to all receivers, the attacker cannot increase his chances of success, as he cannot speed up the propagation of the signal between the sender and the receiver.

These arguments show that the key validity interval and the puzzle hardness need to be appropriately set to reflect the attacker's expected strength and his possible locations. In the following analysis, we consider the worst-case scenario, in which the attacker controls two devices placed in the vicinity of the sender and the receiver, connected via a fast wired link (i.e., effectively forming a worm-hole between two locations). We start by stating the conditions for a message to be successfully authenticated by an honest receiver:

- The key k_i included in the puzzle must be an element of the hash chain.
- If the puzzle has been received at t^B_{r1}, then t^B_{r1} must be in the time interval associated to the key k_i in the puzzle.
- The arrival time t^B_{r2} of $MAC_{k_i}(m)$ must be before $t^B_{r1} + \delta_M$, the point of time, when Bob assumes that Mallory would have solved the puzzle and therefore is able to create his own message $\tilde{m}$ with a valid MAC $MAC_{k_i}(\tilde{m})$. Note that this is where the assumption about the computational power of Mallory comes into play.
- The message m is successfully authenticated by $MAC_{k_i}(m)$.

Therefore we get the following conditions on the parameters for the protocol to achieve the desired security properties:

- $t^B_{r1} \in [t_i, t_{i+1}[$, i.e., the puzzle has been received in the validity period of the corresponding key (element of the hash-chain).
- $t^M_{r1} + \delta_M \notin [t_i, t_{i+1}[$, i.e., the intruder is not able to solve the puzzle in the validity interval i of the associated key.
- $t^M_{r1} + \delta_M \geq t^B_{r2}$, i.e., the intruder cannot solve the puzzle before Bob receives $MAC_{k_i}(m)$ and therefore cannot create a valid MAC for his own message.

- In order for the protocol to achieve instantaneous message authentication, we need $t_{r1}^B + \delta_B \leq t_{r3}^B$, i.e., Bob can solve the puzzle before he receives the message authenticated by the key in the puzzle.

The above analysis shows that secure broadcast authentication can be achieved if $t_{r1}^M + \delta_M \notin [t_i, t_{i+1}[$ and $t_{r1}^M + \delta_M \geq t_{r2}^B$ hold. Furthermore, the protocol achieves instantaneous message authentication if $t_{r1}^B + \delta_B \leq t_{r3}^B$ is fulfilled.

In terms of our definition of broadcast authentication, the protocol authenticates the message m, since no node other than the owner of the one-way chain knows k_i before t_{i+1} and therefore only the owner can create a valid MAC for the message. Similarly m is T-recent since it is verified by the corresponding MAC using the key of the time interval $i = [t_i, t_{i+1}[$. Therefore the protocol achieves T-recentness, where $T \leq |t_{r3}^B - t_i|$, in case of the successful authentication of the message.

This analysis also shows that the ability of the receiver to verify a given message depends on the assumption that the sender can estimate message propagation delays. If the sender can reach all the receivers through a direct link (e.g., a sender is a node with a high-power radio), then estimating propagation delays is not difficult, as it only depends on the link communication speed, which is predictable. However, if the sender broadcasts in a multi-hop network, the estimation of propagation delays will depend on network topology and is more challenging. As we show in Section 4, incorrect propagation delay estimates affect the performance (but not the security) of all protocols that are based on delayed key disclosure by resulting in valid messages being rejected by a subset of network nodes.

Note that the given analysis makes worst-case assumptions in terms of the attacker's abilities, i.e., the attacker receives each message instantaneously from the sender and similarly can deliver messages to the intended receiver without any delay. In reality, it will be more difficult to successfully attack the protocol. For example, if we assume that the puzzle and the MAC are concatenated together within the message, then the attacker has to jam the entire message, solve the puzzle, and create a valid MAC for his own message. Finally the new MAC has to be concatenated with the puzzle and must be delivered to the receiver before the validity interval of the key (inside the puzzle) ends. In a scenario where we have optimal conditions in terms of propagation delay (to the receivers), synchronized clocks, and predefined message-sizes, we can choose a sufficiently small validity interval for keys that minimizes the possibility of a successful attack.

3.2 BAP-2

BAP-2 is based on an approach similar to BAP-1 in that late key disclosure is achieved using cryptographic puzzles. The main difference is that, in BAP-2, not only the key, but also the message and its MAC is encapsulated within a puzzle. This collapses three messages into one and also reduces the time that the attacker has to solve the puzzle in order to break the scheme. The BAP-2 protocol is shown on Figure 4. In this protocol, the sender generates

A(lice)
generate $k_i = h^{n-i}(H_0)$
corresponding to the
time interval i

B(ob)

$$(t_s^A) \xrightarrow{Puzzle(m, MAC_{k_i}(m), k_i)} (t_r^B)$$

receive $Puzzle'$;
solve $Puzzle'$
yielding m', MAC', k_i';
If $Puzzle'$ is received
within time interval i,
and *if* k_i' corresponds
to time interval i
(i.e., $k_i' = k_i$) and
to node A, and *if*
$MAC' = MAC_{k_i'}(m')$
then conclude that
the message $m' = m$
and is authentic
and T-recent
(where $T \leq |i|$)
else reject m

Fig. 4. BAP-2 protocol. The protocol achieves broadcast authentication through delayed key release based on cryptographic puzzles. Message authentication is achieved if the receiver receives the puzzle before the attacker has solved it. All messages received by B are marked with $'$ to denote that they might have been modified in transit by the adversary.

the key $k_i = h^{n-i}(H_0)$ for time interval i. Hence the sender encapsulates the message m, its message authentication code $MAC_{k_i}(m)$, and the key k_i in a puzzle $Puzzle(m, MAC_{k_i}(m), k_i)$. After receiving the puzzle $Puzzle'$, the receiver solves it and then verifies (i) that the $Puzzle'$ was received during the time interval i, (ii) that the key k_i' (derived from $Puzzle'$) is indeed authentic and that it corresponds to the current time slot i and to the claimed sender A, and (iii) that the message authentication code MAC' derived from the puzzle corresponds to $MAC_{k_i'}(m')$ computed with the derived key k_i' over the derived message m'. If and only if all three verifications succeed, the receiver concludes that the message $m' = m$ is both authentic (i.e., generated by the claimed source A) and T-recent (where $T \leq |i|$). Consequently, BAP-2 reaches T-authentication according to Definition 1, where $T \leq |i|$.

One advantage of BAP-2 over BAP-1 is that the attacker has less time to solve the puzzle. Namely, as soon as the first bits of the puzzle are received by the receiver, the attacker looses the possibility to forge the message. Therefore, the key validity time intervals can be shortened in BAP-2 with respect to the intervals in BAP-1, assuming the same message size, key size, and propagation delays. One drawback of this solution is the loss of instantaneous message verification and the inability to prepare the puzzles beforehand (unless the messages are largely predictable or drawn from a small, well-defined set).

The security analysis of BAP-2 closely resembles that of BAP-1 and we therefore omit further details. Similar to BAP-1, we require that the attacker cannot generate a valid message prior to solving the puzzle and cannot solve the puzzle before the validity of the key expires.

3.3 Cryptographic Puzzles

In this section, we discuss possible realizations of the cryptographic puzzles used in our protocols. Cryptographic puzzles were first suggested by Merkle [9] and led to the invention of public-key cryptography. In [29], Rivest et al. present a construction of time-lock puzzles based on repeated squaring. The main contribution of time-lock puzzles is that they are non-parallelizable as solving them requires iterated application of an inherently sequential set of operations. However, solving this kind of puzzle requires the use of modular arithmetic and is therefore prohibitively expensive in networks composed of resource-constrained devices. Juels and Brainard [18] propose client puzzles based on one-way hash functions with partially disclosed hash input values. Their client puzzles use light-weight cryptographic primitives, but as they rely on exhaustive search, they are parallelizable. The main advantage of both time-lock and client puzzles is that they are simple to construct, but take significant time to solve. If the time required for puzzle construction is not prohibitive (e.g., can be performed during idle time), another puzzle construction scheme can be used. Namely, puzzles can be constructed by iterating a strong encryption function (such as AES [24]) a predefined number of times over a protected message, while either partially or entirely disclosing the used encryption keys. To solve this puzzle, a receiver needs to decrypt the ciphertext the same number of times as it was encrypted. Puzzles constructed based on iterated encryption are therefore non-parallelizable.

We now summarize desirable properties for puzzle schemes usable for broadcast authentication. (i) Puzzle generation should be computationally inexpensive for the sender. (ii) A puzzle should be solvable by a receiver within a given, finite time interval. In particular, a sender should be able, with the same puzzle scheme, to generate puzzles which a receiver can solve in short time as well as puzzles for which a receiver needs more time to solve. (iii) Solving a puzzle should not be parallelizable. This prevents a puzzle being solved faster by several colluding devices.

Although none of the described puzzle construction schemes satisfies all these requirements, both hash-based client puzzles and puzzles based on iterative encryption can be used for broadcast authentication in networks of resource scarce devices (e.g., sensor networks). Hash-based client puzzles neither incur heavy cost in terms of storage nor in terms of puzzle generation; these puzzles are therefore well suited for scenarios in which *any* sensor node is a potential broadcasting node. A drawback of client puzzles is that they are parallelizable and therefore introduce a security risk in the broadcast authentication scheme if the attacker's strength (in terms of the number and type of devices that it holds) is underestimated. Puzzles based on iterated encryption functions are costly to generate, but are non-parallelizable; these puzzles can therefore be used in scenarios in which a subset of selected network nodes (i.e., network sinks, base stations, or cluster heads) broadcast messages to the network of resource-constrained (sensor) nodes. Given that these selected nodes have higher computational and storage capabilities than "regular" sensor nodes, they can compute and store puzzles for all the keys prior to their use in broadcast communication. These puzzles can therefore be computed for all keys in parallel with the creation of hash chains from which the keys are derived.

4 Analysis of Broadcast Authentication Protocols Using Delayed Key Disclosure

In this section, we analyze common properties shared by all broadcast authentication protocols based on one-way chains and delayed key disclosure. The common elements found in all these schemes are *validity windows* and a *disclosure schedule* for the keys, i.e., for the elements of the one-way chain (see Section 2.2 for further explanations). We will focus our analysis on the impact of clock synchronization and propagation delay on the security and performance of these protocols.

In the next subsection, we model the commonalities of this protocol class as a protocol pattern that can be found in any of these schemes. Therefore security properties related to this skeleton are relevant for this entire class of protocols. Examples of protocols in this class are the BAP-1 and BAP-2 protocols just presented, Cheung's authentication scheme [8], and TESLA, including all its variants [27].

4.1 The Protocol Pattern

As explained in Section 2.2, the key idea of using one-way chains for authentication is to associate elements of the chain in reverse order to time intervals. Therefore there are two important notions related to an element of the one-way chain: (i) the validity window of the element, i.e., the interval in which it should be used, and (ii) the point in time when the key is published, in order to prove the correctness of a MAC (built with a key derived from the element) and to prove the key's membership in the one-way chain.

In the protocol pattern below, we consider that the sender holds a one-way chain consisting of elements $(H_k)_{0\leq k\leq n}$. To prove the origin of a message to a receiver, the sender appends the message authentication code $MAC(m_i, K_i)$ to a message m_i sent in time interval $[t_i, t_{i+1}[$, where K_i is derived from H_i. $[t_i, t_{i+1}[$ denotes the validity window of the element H_i, i.e., K_i is only accepted in this interval as a key. Finally the chain element H_i is released (published) at $t_i + d$, i.e., d time units after the start of K_i's validity window.

Note that these parameters may vary in different implementations. For example, in TESLA [26] the chain element H_i is released two validity windows after it has been used to create a key for a MAC. Therefore $d_{TESLA} = 2 \cdot |t_{i+1} - t_i|$. In our protocols, we do not send the chain element after the message, but implement the delayed key disclosure by hiding the key in a puzzle. Therefore d_{BAP}

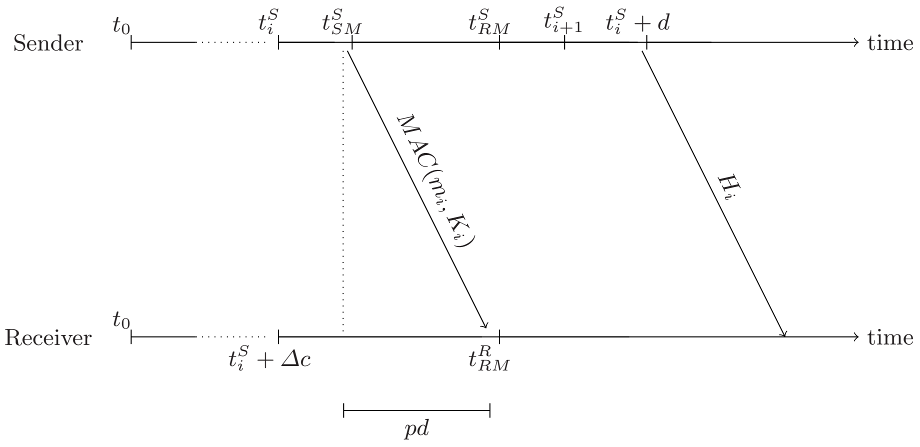


Fig. 5. Common Pattern for Delayed Key Disclosure Protocols

is determined by the difficulty of the puzzle and the assumptions made about the attacker's computational power.

In Figure 5, the superscripts S (Sender) and R (Receiver) indicate the clock taken as a reference. Furthermore we make the following definitions and assumptions:

t_0:	sender and receiver synchronize their clocks.
t_i^S, t_{i+1}^S:	start and end of the validity interval for key K_i.
t_{SM}^S:	$MAC(m_i, K_i)$ is sent.
t_{RM}^R:	the receiver has received $MAC(m_i, K_i)$.
d:	key disclosure delay, i.e. H_i is disclosed at $t_i^S + d$.
pd:	the propagation delay of the message.
Δc:	clock difference between sender and receiver.

Concerning the synchronization of the receiver's and sender's clock, we assume that at time t_i^S on the sender's clock, the receiver's clock shows $t_i^S + \Delta c$. For our observations, it is suffices to assume that Δc is constant since we consider here a relatively short time interval.

In order for the protocol to guarantee T-authentication, the following conditions must be fulfilled:

i) The MAC for message m_i is computed using the key K_i, which is valid in the interval when the puzzle and the MAC are sent.
ii) In order for the receiver to accept the message as being authentic, MAC (m_i, K_i) (but not necessarily the message m_i) must be received within the validity interval of K_i.

From the sender's point of view, the receiver receives the message at $t_{RM}^S = t_{SM}^S + pd$. At this point in time, the receiver's clock shows $t_{RM}^R = t_{RM}^S + \Delta c + pd$. In order for the receiver to accept the message as valid, the inequality $t_{RM}^R < t_{i+1}^S$

must be fulfilled. Note at this point, that the boundaries (t_i and t_{i+1}) of the validity interval are given in the protocol specification and therefore $t_i^S = t_i$. Given $t_{RM}^R = t_{RM}^S + \Delta c + pd$, we have the condition $t_{RM}^S + \Delta c + pd < t_{i+1}$ for the receiver to accept the message as being sent by the holder of the one-way chain. Since $t_{RM}^S \geq t_i$, we conclude $t_i + \Delta c + pd < t_{i+1}$. Note that $t_{RM}^S = t_i$ is the earliest point in time when the key K_i would be used to authenticate a message. Therefore for the message to be successfully authenticated and therefore to be accepted by the receiver, we have the condition that:

$$\Delta c + pd < t_{i+1} - t_i \tag{1}$$

From the security standpoint, the time when the key is disclosed is particularly critical. Under the assumption of perfect cryptography, this is the only possibility for the attacker to acquire the key and therefore to create a valid message.

To analyze the performance and security properties of the schemes, we will now consider the possible values of the parameters involved, case by case.

4.2 Impact of the Accuracy of Time Synchronization

As we mentioned in the description, we assume that sender and receiver have synchronized their clocks at t_0. At t_i we assume the clocks to show a difference of Δc time units, due to clock drift. We therefore have the following two possibilities:

$\Delta c < 0$: The receiver's clock is slower than the sender's clock. By (1), even with a larger propagation delay pd, the message would still be accepted by the receiver. Therefore the receiver would accept messages authenticated by keys that are, according to the sender's clock, no longer valid. From the security point of view, the critical point is reached if $|\Delta c| > d - (t_{i+1} - t_i)$. This opens the possibility of an attack against the scheme if the intruder is able to jam the original message and waits (or solves the puzzle) for the key, which he could then use to create a valid MAC for his own message.

$\Delta c > 0$: The receiver's clock is faster than the sender's clock. In this case, the validity window on the client side shrinks and therefore no attacks are possible, except breaking the one-way chain (or the puzzle). On the other hand, a message could be invalidated by the receiver, although it would still be valid from the sender's perspective.

4.3 Impact of the Propagation Delay

In this section, we analyze the impact of propagation delay on the performance of such a scheme. For simplicity, in this analysis, we assume perfect synchronization (i.e., $\Delta c = 0$). Therefore the inequality reduces to $pd < t_{i+1} - t_i$, as a necessary condition for the receiver to accept messages. If he receives a message after a key is no longer valid, he will reject the message. This implies that, under the assumption of perfectly synchronized clocks, this class of protocols can only be used in an environment where the propagation delays of all nodes involved are known in advance, since the validity window and the disclosure delay have then to be chosen appropriately. In multi-hop networks, propagation delays are difficult to estimate in advance [13]. Since the disclosure delay has then to be set according to the maximum propagation delay, this would delay the time of verification drastically.

4.4 Differences to Schemes Using Public Key Cryptography

As indicated in Section 2, the broadcast authentication problem is easily solved using asymmetric cryptography, but the computational cost is high. Hence, in order for a protocol to meet the requirements of resource-constraint devices, schemes using symmetric cryptography were introduced. But clearly, some form of asymmetry is needed in the solutions. The direct use of symmetric cryptography fails as the ability to verify the authenticity of a message is equivalent to the ability to authenticate an arbitrary message. One-way chains introduce the required asymmetry by letting the holder of the root element commit to a set of keys (the chain), while only publishing a single element (the last element of the chain). By using the elements of the chain as keys in keyed MAC functions, we transfer the asymmetry of knowing a certain key (chain element) to the ability to create a valid message authentication code. Since the key is needed to prove the validity of the MAC, the key must be disclosed. Upon disclosure, the asymmetry of the MAC is lost since, from this point on, anybody can create MACs using this key.

The asymmetry of the scheme is preserved by using a predefined disclosure schedule: if a MAC is created before the disclosure of the key, only the holder of the root of the one-way chain could have created it. Therefore we gain an additional property, namely evidence of when the message has been sent by the sender given by the time interval in which the key is intended to being used (before disclosure). This gives us *T-recentness*, where T is defined by the disclosure schedule and the arrival time of the message. These considerations also help illustrate the tradeoff in schemes using this technique. For efficiency reasons, the key should be disclosed as early as possible for the receiver to be able to verify instantaneously the authenticity of the corresponding message. But for security reasons, the key should not be disclosed too early, so that an attacker cannot create a valid MAC by using a disclosed key.

5 Related Work

Efficient broadcast (and multicast) authentication in wireless networks is an active field of research. In recent years, a number of proposals emerged that address this problem. To our knowledge, the first authentication scheme based on delayed key disclosure was introduced by Cheung [8] in the context of secure link state updates in routing protocols for wired networks. This technique was later used by Perrig et al. in [26], who propose TESLA, a broadcast authentication protocol based on delayed key disclosure. This approach relies on explicit key disclosure after the message (i.e., message and MAC). In [27], Perrig and Tygar present a suite of broadcast authentication protocols, including a Tesla variant named TIK, that achieves instant message verification with a single message release. A number of extensions of TESLA are presented in [20,22,21].

In TESLA and its variants, the time of message verification is delayed until the point in time when the key is disclosed by the message source. In comparison, our schemes achieve similar properties by sending the authentication key hidden in a puzzle with, or prior to, the message. Therefore our schemes reduce the number of messages needed to guarantee authenticity. Since our protocols are based on cryptographic puzzles, the security and also the performance of BAP-1 and BAP-2 depend on the computational power of the attacker and the honest nodes involved.

In the case of instantaneous message authentication, all schemes based on delayed key disclosure exhibit a trade-off between security and performance. As we have pointed out in the Section 4, the smaller the validity window of a key is chosen, the smaller the propagation must be. Since the message should be authenticated right after reception, the disclosure delay (and therefore also the validity window) of the key must be chosen small in order to guarantee authentication. As a consequence, the allowed propagation delay is equally small. Therefore TIK (the TESLA variant guaranteeing instantaneous message authentication) and BAP-1 suffer from the same restrictions on the allowed propagation delay. BAP-1 achieves broadcast authentication under the assumption that the puzzle has been solved up to the point where the message is received, whereas in TIK, the key bits need to be sent right after the message.

Besides the approaches based on delayed key disclosure, alternatives based on one-time signatures have been proposed in the context of broadcast authentication [27,7]. Other approaches include broadcast authentication based on receiver proximity awareness [31]. Moreover, there are similar contributions in the field of multicast authentication that are more related to wired networks, e.g., [3], [14], [4].

Information-theoretically secure broadcast authentication mechanisms were proposed by a number of researchers [1], [10]. These protocols typically have a high overhead with many receivers and do not scale in large (sensor) networks. Canetti et al. present a broadcast authentication protocol in which a message is authenticated with k different MAC's [4] and in which no coalition of w (corrupted) receivers can forge a packet for a specific receiver.

Boneh, Durfee, and Franklin show in [2] that a compact collusion-resistant broadcast authentication protocol cannot be built without relying on digital signatures or on time synchronization.

6 Conclusion

In this paper, we have introduced two broadcast authentication protocols based on symmetric-key primitives and delayed key disclosure. We showed that these protocols achieve T-authentication, a form of authentication that includes a notion of recentness. By identifying the core components of broadcast authentication protocols using delayed key disclosure, we were able to uncover fundamental limitations of this class of protocols. Our results show that there is a trade-off between security and performance for protocols of this class. Furthermore, for performance and security reasons, the propagation delay inside a network must be known beforehand in order to choose the protocol parameters correctly. Since these delays are difficult to predict in multi-hop wireless networks, this class of protocols is not well suited for such networks. However, these protocols are well suited for wireless broadcasts where the receivers are in the direct range of the transmitter.

Acknowledgment. The authors would like to thank Ueli Maurer for a useful discussion on cryptographic puzzles.

References

1. Shimshon Berkovits. How to broadcast a secret. In *EUROCRYPT*, 1991.
2. Dan Boneh, Glenn Durfee, and Matt Franklin. Lower bounds for multicast message authentication. *Lecture Notes in Computer Science*, 2001.
3. Ran Canetti, Pau-Chen Cheng, Frederique Giraud, Dimitrios Pendarakis, Josyula R. Rao, and Pankaj Rohatgi. An IPSec-based host architecture for secure internet multicast. In *Internet Society Symposium on Network and Distributed Systems Security*, 2000.
4. Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, 1999.
5. Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, 2003.
6. Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh. An efficient broadcast authentication scheme. In *ACM Symposium on Information, Computer and Communications Security*. ACM, 2006.
7. Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006.
8. S. Cheung. An efficient message authentication scheme for link state routing. In *ACSAC*, 1997.
9. Ralph C.Merkle. Secure communications over insecure channels. *Communications of the ACM*, 1978.
10. Yvo Desmedt, Yair Frankel, and Moti Yung. Multi-receiver/multi-sender network security: efficient authenticated multicast/feedback. In *IEEE INFOCOM '92: Proceedings of the eleventh annual joint conference of the IEEE computer and communications societies on One world through communications*, 1992.
11. Wenliang Du, Ronghua Wang, and Peng Ning. An efficient scheme for authenticating public keys in sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005.
12. Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, 2002.
13. Saurabh Ganeriwal, Srdjan Capkun, Chih-Chieh Han, and Mani B. Srivastava. Secure time synchronization service for sensor networks. In *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, 2005.

14. Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. *Lecture Notes in Computer Science*, 1997.
15. Ivan Getting. The Global Positioning System. *IEEE Spectrum*, December 1993.
16. Ralf Hauser, Antoni Przygienda, and Gene Tsudik. Reducing the cost of security in link state routing. In *Internet Society Symposium on Network and Distributed Systems Security*, 1997.
17. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne:: a secure on-demand routing protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, 2002.
18. Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. *Proccedings of NDSS '99 (Network and Distributed Security Systems)*, 1999.
19. Leslie Lamport. Password authentication within insecure communication. *Communications of the ACM*, 1981.
20. Donggang Liu and Peng Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. Technical report, Raleigh, NC, USA, 2002.
21. Donggang Liu and Peng Ning. Multilevel μ-tesla: Broadcast authentication for distributed sensor networks. *Trans. on Embedded Computing Sys.*, 3(4), 2004.
22. Donggang Liu, Peng Ning, Sencun Zhu, and Sushil Jajodia. Practical broadcast authentication in sensor networks. In *MOBIQUITOUS '05: Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 118–132, Washington, DC, USA, 2005. IEEE Computer Society.
23. Michael Manzo, Tanya Roosta, and Shankar Sastry. Time synchronization attacks in sensor networks. In *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, 2005.
24. National Institute of Stanards and Technology. Announcing the advanced encryption standard (aes). *Federal Information, Processing Standards Publication 197*, 2001.
25. Adrian Perrig. The biba one-time signature and broadcast authentication protocol. In *ACM Conference on Computer and Communications Security*, pages 28–37, 2001.
26. Adrian Perrig, Ran Canetti, Doug Tygar, and Dawn Song. The tesla broadcast authentication protocol. *RSA Cryptobytes*, 2002.
27. Adrian Perrig and Doug Tygar. *Secure Broadcast Communication in Wired and Wireless Networks.* Kluwer Academic Publishers, 2003.
28. Ronald L. Rivest, Adi Shamir, and Leonard M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. Technical Report MIT/LCS/TM-82, 1977.
29. Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. (MIT/LCS/TR-684):21, 1996.
30. Kun Sun, Peng Ning, and Cliff Wang;. Secure and resilient clock synchronization in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 2006.
31. Mario Čagalj, Srdjan Čapkun, RamKumar Rengaswamy, Ilias Tsigkogiannis, M. Srivastava, and Jean-Pierre Hubaux. Integrity (I) codes: Message Integrity Protection and Authentication Over Insecure Channels. In *IEEE Symposium on Security and Privacy*, 2006.

Compressed XTR

Masaaki Shirase[1], Dong-Guk Han[2,⋆], Yasushi Hibino[3],
Ho Won Kim[2], and Tsuyoshi Takagi[1]

[1] Future University-Hakodate, Japan
shirasem@yahoo.co.jp, takagi@fun.ac.jp
[2] Electronics and Telecommunications Research Institute(ETRI), Korea
{christa,khw}@etri.re.kr
[3] Japan Advanced Institute of Science and Technology(JAIST), Japan
hibino@jaist.ac.jp

Abstract. XTR public key system was introduced at Crypto 2000, which is based on a method to present elements of a subgroup of a multiplicative group of a finite field. Its application in cryptographic protocols leads to substantial savings both in communication and computational overhead without compromising security. It was shown how the use of finite extension fields and subgroups can be combined in such a way that the number of bits to be exchanged is reduced by a factor 3.

In this paper we show how to more compress the communication overhead. The compressed XTR leads to a factor 6 reduction in the representation size compared to the traditional representation and achieves as twice compactness as XTR. The computational overhead of it is a little worse than that of XTR, however the compressed XTR requires only about additional 6% computational effort. If finding 4-th roots of unity is pre-computed, then the computational overhead is only 1% compared to that of original XTR. Furthermore, the required size of public key data of it reduces about 26% from that of XTR.

1 Introduction

In the classical Diffie-Hellman (DH) key exchange scheme, two system parameters are fixed: a large prime number q and a generator g of the multiplicative group of the basic prime field $\mathbb{F}_q$. In the basic DH scheme the two parties each send a random power of g to the other party. Assuming both parties know q and g, each party transmits about $\log_2(q)$ bits to the other party.

In [4], ElGamal suggested that finite extension fields can be used instead of prime fields, but no direct computational or communication advantages where implied. In [10], Schnorr proposed a variant of the classical Diffie-Hellman scheme, in which g does not generate the whole multiplicative group of the prime field $\mathbb{F}_q$, but only a small subgroup of which the order contain relatively small compared to q. This considerably reduces the computational cost of the DH scheme, but has no effect on the number of bits to be exchanged.

⋆ Corresponding author.

After that, it has tried to make use of traces to represent and calculate powers of elements of a subgroup of a finite field to achieve efficient and compact subgroup representation. The LUC cryptosystem uses the trace over $\mathbb{F}_q$ to represent elements of the order $q+1$ subgroup of $\mathbb{F}_{q^2}^*$ [11]. Compared to the traditional representation LUC leads to a factor 2 reduction in the representation size. The variant described in [5] uses the subgroup of order q^2+q+1 of $\mathbb{F}_{q^3}^*$ instead, but as a result sizes are reduced by only a factor 1.5. In [2], Brouwer et al. introduced for the first time how the use of finite extension fields and subgroups can be combined in such a way that the number of bits to be exchanged is reduced by a factor 3. More specifically, it was shown that elements of an order p subgroup of $\mathbb{F}_{q^6}^*$ can be represented using $2\log_2(q)$ bits if p divides q^2-q+1. Despite its communication efficiency, the method of it is rather troublesome and computationally not particularly efficient.

In 2000 Lenstra-Verheul introduced XTR [7], a cryptosystem using the trace over $\mathbb{F}_{q^2}$ to represent elements of the order q^2-q+1 subgroup of $\mathbb{F}_{q^6}^*$, there by achieving a factor 3 size reduction. Also, the resulting calculations are appreciably faster than using the standard representation. XTR of security equivalent to 1024-bit RSA achieves speed comparable to cryptosystems based on random elliptic curves over random prime fields (ECC) of equivalent security. The corresponding XTR public keys are only about twice as large as ECC keys, assuming global system parameters - without the last requirement the sizes of XTR and ECC public keys are bout the same. Furthermore, parameter initialization from scratch for XTR takes a negligible amount of computing time, unlike RSA and ECC. Combined with its very easy programmability, this makes XTR an excellent public key system for a very wide variety of environments, ranging from smart cards to web servers.

In this paper we present a greatly improved version of XTR that leads to a factor 6 reduction in the representation size compared to the traditional representation. That is to say, we achieve a factor 2 reduction compared to the original XTR. We show that if the characteristic of q is three, i.e. $q=3^{2k-1}$ for some integer k, then we can use the trace over $\mathbb{F}_q$ to represent elements of the order $q-\sqrt{3q}+1$ subgroup of $\mathbb{F}_{q^6}^*$. Also, the resulting calculations such as exponentiations are as faster as that of XTR. Given $Tr_{(q^6,q)}(g)$ and n, $Tr_{(q^6,q)}(g^n)$ takes about 1381 multiplications in $\mathbb{F}_q$, which is only about 6% increase compared to the cost of computation of $Tr_{(q^6,q^2)}(h^n)$ for given $Tr_{(q^6,q^2)}(h)$ and n, where the size of n is 160 bits. If q is fixed, then finding square root of -1 can be pre-computed. In this case, the computational overhead is only 1% compared to that of original XTR. Furthermore, the required size of public key data of it reduces about 26% from that of the original XTR.

In Section 2 we describe XTR, and in Section 3 we introduce XTR over characteristic three, which achieves a factor 2 reduction in the representation size compared to XTR. Section 4 shows efficient calculations of XTR exponentiation over characteristic three. Applications and comparisons to the original XTR are given in Section 5.

2 XTR

2.1 Description of XTR

XTR uses a subgroup of prime order p of the order $q^2 - q + 1$ subgroup of $\mathbb{F}_{q^6}^*$. The latter group is referred to as the *XTR supergroup* denoted as G_{q^2-q+1} and the order p subgroup G_p is referred to as the *XTR group*. The XTR supergroup G_{q^2-q+1} is not contained in any proper subfield of $\mathbb{F}_{q^6}$ due to the following fact.

Fact 1. [8] *Let p be a prime factor of $\Phi_m(q)$, where m-th cyclotomic polynomial for a positive integer m not divisible by q. Then the subgroup G_p of $\mathbb{F}_{q^m}^*$ is not contained in any proper subfield of $\mathbb{F}_{q^m}$.*

Combined with the choice of p it follows that computing discrete logarithms in G_p is as hard, in general, as it is in $\mathbb{F}_{q^6}^*$ [7].

Before describing XTR more detail, we introduce two definitions about optimal normal basis.

Definition 1. *Type I Optimal Normal Basis (Type-I ONB)*
If $m+1$ is a prime and q is a generator of $\mathbb{F}_{m+1}^*$, then the set $\{\omega^m, \omega^{m-1}, \cdots, \omega^2, \omega\}$ forms an optimal normal basis of type I in $\mathbb{F}_{q^m}$ and called Type-I ONB. Here, ω is the primitive $(m+1)$-th root of unity.

Definition 2. *Type II Optimal Normal Basis (Type-II ONB)*
If $2m + 1$ is a prime and either of the following two conditions holds,

- *q is a primitive root module $2m + 1$,*
- *q is a quadratic residue module $2m + 1$ and $p \not\equiv 1 \bmod (2m + 1)$,*

then the set $\{\beta^m, \beta^{m-1}, \cdots, \beta^2, \beta\}$ forms an optimal normal basis of type II in $\mathbb{F}_{q^m}$ and called Type-II ONB. Here, $\beta = \gamma+\gamma^{-1}$ and γ is the primitive $(2m+1)$-th root of unity.

XTR uses $\mathbb{F}_{q^2}$ arithmetic to achieve $\mathbb{F}_{q^6}$ security, without requiring explicit construction of $\mathbb{F}_{q^6}$. Let q be a prime that is 2 mod 3. It follows that $(X^3 - 1)/(X - 1) = X^2 + X + 1$ is irreducible over $\mathbb{F}_q$ and the zeros α and α^q of it form an Type-I ONB for $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$. In XTR elements of G_p are represented by their trace over $\mathbb{F}_{q^2}$. For $h \in \mathbb{F}_{q^6}^*$ the trace $Tr_{(q^6,q^2)}(h)$ over $\mathbb{F}_{q^2}$ is defined as the sum of the conjugates over $\mathbb{F}_{q^2}$ of h, i.e. $Tr_{(q^6,q^2)}(h) = h + h^{q^2} + h^{q^4} \in \mathbb{F}_{q^2}$. Let p and q be primes with p dividing $q^2 - q + 1$. Also let h be a generate of G_p and let $c = Tr_{(q^6,q^2)}(h)$. Suggested lengths to provide adequate levels of security are $\log_2(q) \approx 170$ and $\log_2(p) \approx 160$.

c_n denotes $Tr_{(q^6,q^2)}(h^n) \in \mathbb{F}_{q^2}$, for some q and h of order p dividing $q^2 - q + 1$ as above. Efficient computation of c_n given q, p and c depends on the recurrence relation

$$c_{u+v} = c_u c_v - c_v^q c_{u-v} + c_{u-2v}, \tag{1}$$

for $u, v \in Z$. It simplifies for $u = v$ to

$$c_{2u} = c_u^2 - 2c_u^q. \tag{2}$$

In [7], Lenstra and Verheul proved that computing c_{u+v} and c_{2u} take four and two multiplications in $\mathbb{F}_q$ respectively, when c_u, c_v, c_{u-v}, and c_{u-2v} are given.

2.2 XTR Exponentiation

In XTR, an algorithm for computing $Tr_{(q^6,q^2)}(h^n)$ given $Tr_{(q^6,q^2)}(h)$ and a scalar $n \in Z$ is needed like the algorithm for computing h^n in public key system based on discrete logarithm problem. By using two formula (1),(2) above, we define the following two functions called as XTR addition and XTR doubling respectively;

$$A[u, v, w, z] = u \cdot v - v^q \cdot w + z,$$
$$D[u] = u^2 - 2u^q.$$

XTR Exponentiation ([7], Algorithm 2.3.7)

```
INPUT: $c$ and $n$ where $n > 2$
OUTPUT: $c_n$
 1. Compute initial values:
    1.1. $C_3 \leftarrow c$, $C_0 \leftarrow D[C_3]$, $C_1 \leftarrow A[C_0, C_3, C_3, 3]$, and $C_2 \leftarrow D[C_0]$
    1.2. If $n$ is even, $n$ replace $n - 1$.
         Let $n = 2m + 1$ and $m = \sum_{j=0}^{l} m_j 2^j$ with $m_j \in \{0, 1\}$ and $m_l = 1$.
 2. for $j = l - 1$ down to 0
    2.1. $T_1 \leftarrow D[C_{m_j}]$
    2.2. $T_2 \leftarrow D[C_{1+m_j}]$
    2.3. if $(m_j = 0)$ then $T_3 \leftarrow A[C_0, C_1, C_3^q, C_2^q]$
         if $(m_j = 1)$ then $T_3 \leftarrow A[C_2, C_1, C_3, C_0^q]$
    2.4. $C_0 \leftarrow T_1$
    2.5. $C_1 \leftarrow T_3$
    2.6. $C_2 \leftarrow T_2$
 3. If $n$ is odd then return $C_1$
    else return $C_2$
```

Theorem 1. ([7], *Theorem* 2.3.8) *Let c and a positive integer n be given. Computing the sum c_n of the n^{th} powers of the roots takes $8\log_2(n)$ multiplications in $\mathbb{F}_q$.*

Thus, given the representation $Tr_{(q^6,q^2)}(h) \in \mathbb{F}_{q^2}$ of the conjugates of h, the representation $Tr_{(q^6,q^2)}(h^n) \in \mathbb{F}_{q^2}$ of the conjugates of the n^{th} power of h can be computed at the cost of $8\log_2(n)$ multiplications in $\mathbb{F}_q$, for any integer n.

Denote the above XTR exponentiation with input c and n outputs c_n as

$$\mathsf{XTR\_Exp}[c, n] = c_n.$$

2.3 XTR-DH Key Agreement

XTR can be used in any cryptosystem that relies on the discrete logarithm problem. This section contains a description of an application of XTR that provides confidentiality service, for example Diffie-Hellman key agreement.

Public Parameters : q, p, $c = Tr_{(q^6,q^2)}(h)$

If Alice and Bob want to agree on a secret key K they do the following.

1. Alice selects at random $a \in Z_p$, uses $\mathsf{XTR\_Exp}[c,a] = c_a \in \mathbb{F}_{q^2}$, and sends c_a to Bob.
2. Bob receives c_a from Alice, selects at random $b \in Z_p$, uses $\mathsf{XTR\_Exp}[c,b] = c_b \in \mathbb{F}_{q^2}$, and sends c_b to Alice.
3. Alice receives c_b from Bob, computes $\mathsf{XTR\_Exp}[c_b,a] = c_{ba}$, and determines K based on $c_{ba} := Tr_{(q^6,q^2)}(h^{ba})$.
4. Bob uses $\mathsf{XTR\_Exp}[c_a,b] = c_{ab}$, and determines K based on $c_{ab} := Tr_{(q^6,q^2)}(h^{ab})$.

3 XTR over Characteristic Three

The original XTR uses the trace over $\mathbb{F}_{q^2}$ to represent elements of the order $q^2 - q + 1$ subgroup of $\mathbb{F}_{q^6}^*$, thereby achieving a factor 3 size reduction. This section shows that if q is 3 to the odd power then elements in G_{q^2-q+1} can be represented as elements in $\mathbb{F}_q$ using the trace over $\mathbb{F}_q$. It achieves a factor 6 size reduction, which is the half size reduction compared to the original XTR representation.

3.1 New XTR Group

We assume that $q = 3^t$ for any odd integer t, say $t = 2k - 1$. Then, $\sqrt{3q} = 3^k$ is an integer and $q^2 - q + 1$ is factorized as

$$q^2 - q + 1 = (q + \sqrt{3q} + 1)(q - \sqrt{3q} + 1).$$

In this section, we define a new XTR group $G_p =< g >$ which is a subgroup of $G_{q-\sqrt{3q}+1}$, namely, XTR uses a subgroup of prime order p of the order $q - \sqrt{3q} + 1$ subgroup of $\mathbb{F}_{q^6}^*$. The order p subgroup $< g >$ generated by g is referred as the *New XTR group*. Since p does not divide any $q^s - 1$ for $s = 1, 2, 3$, the new XTR group G_p generated by g cannot be embedded in the multiplicative group of any true subfield of $\mathbb{F}_{q^6}$. Combined with the choice of p it follows that computing discrete logarithms in G_p is as hard, in general, as it is in $\mathbb{F}_{q^6}^*$ (cf. [7], Section 5).

$$G_p =< g > \quad \lhd \quad G_{q-\sqrt{3q}+1} \quad \lhd \quad G_{q^2-q+1} \quad \lhd \quad G_{q^3-1} \tag{3}$$

Here, $A \lhd B$ denotes A is a subgroup of B.

From $q = 3^t$ and t is odd it follows q is a generator of $\mathbb{F}_5^*$, so that $\{\omega + \omega^{-1}, \omega^2 + \omega^{-2}\}$ form an Type-II ONB for $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$, where ω is a root of the polynomial $(X^5 - 1)/(X - 1) = X^4 + X^3 + X^2 + X + 1$. For the simplicity, we denote $x = x_1 \cdot (\omega + \omega^{-1}) + x_2 \cdot (\omega^2 + \omega^{-2}) \in \mathbb{F}_{q^2}$ as (x_1, x_2).

Lemma 1. *Let $x, y, z \in \mathbb{F}_{q^2}$ with $q = 3^t$ and t is odd.*

i. Computing x^q is for free.
ii. Computing x^2 takes two multiplications in $\mathbb{F}_q$.
iii. Computing $x * z - y * z^q$ takes four multiplications in $\mathbb{F}_q$.

Proof. Let $x = (x_1, x_2)$, $y = (y_1, y_2)$ and $z = (z_1, z_2) \in \mathbb{F}_{q^2}$ for $x_i,\ y_i,\ z_i \in \mathbb{F}_q$, $i \in \{1, 2\}$. From $(\omega+\omega^{-1})^q = \omega^2+\omega^{-2}$ and $(\omega^2+\omega^{-2})^q = \omega+\omega^{-1}$, $x^q = (x_2, x_1)$. It follows that q^{th} powering in $\mathbb{F}_{q^2}$ does not require arithmetic operations and can thus be considered to be for free.

From $x^2 = \big((x_1 + x_2)(x_1 - x_2) + 2x_1x_2, 2(x_1 + x_2)(x_1 - x_2) + 2x_1x_2\big)$, x^2 is obtained from two multiplications in $\mathbb{F}_q$.

Finally, to compute $x * z - y * z^q$ four multiplications in $\mathbb{F}_q$ suffice, because it is easily verified that

$$\begin{aligned} x * z - y * z^q = &\big((x_2 - 2x_1 + y_2 - y_1) * z_1 + (x_1 - x_2 + 2y_1 - y_2) * z_2\big) * (\omega + \omega^{-1}) \\ &+ \big((x_2 - x_1 + 2y_2 - y_1) * z_1 + (x_1 - 2x_2 + y_1 - y_2) * z_2\big) * (\omega^2 + \omega^{-2}). \end{aligned}$$

□

3.2 Compression and Restoration

For some q and g of order p dividing $q - \sqrt{3q} + 1$, define d and e as the trace $Tr_{(q^6,q^2)}(g)$ over $\mathbb{F}_{q^2}$ and the trace $Tr_{(q^6,q)}(g)$ over $\mathbb{F}_q$, respectively. We use the shorthand $d_n = Tr_{(q^6,q^2)}(g^n)$ and $e_n = Tr_{(q^6,q)}(g^n)$, i.e. e_n and d_n are the sum of the conjugates over $\mathbb{F}_{q^2}$ and $\mathbb{F}_q$ of g^n respectively. Immediately, $d = d_1$ and $e = e_1$.

$$\begin{aligned} d_n &= Tr_{(q^6,q^2)}(g^n) = g^n + g^{nq^2} + g^{nq^4} \in \mathbb{F}_{q^2} \\ e_n &= Tr_{(q^6,q)}(g^n) = g^n + g^{nq} + g^{nq^2} + g^{nq^3} + g^{nq^4} + g^{nq^5} \in \mathbb{F}_q. \end{aligned}$$

Compression. From the definition of d_n and e_n, e_n can be easily derived from d_n due to the following equation

$$e_n = d_n + d_n^q. \tag{4}$$

For any $d_n = x(\omega + \omega^{-1}) + y(\omega^2 + \omega^{-2}) \in \mathbb{F}_{q^2}$, we have that $e_n = (x + y) * (\omega + \omega^{-1}) + (x + y) * (\omega^2 + \omega^{-2}) \in \mathbb{F}_q$ because of Lemma 1-*i*. Note that as $d_n \in \mathbb{F}_{q^2}$ and $d_n \notin \mathbb{F}_q$, $x \neq y$.

Define a compression function with input an element of $\mathbb{F}_{q^2}$ represented by two elements of $\mathbb{F}_q$, say (x, y), outputs an element of $\mathbb{F}_q$.

$$\mathsf{Compression}[x, y] = x + y$$

Restoration. Contrary to the compression from d_n to e_n, this section explains how to get d_n from e_n, called it as restoration in this paper.

Lemma 2. *The roots of* $X^2 - e_nX + e_n^{\sqrt{3q}} \in \mathbb{F}_q[x]$ *are* d_n *and* d_n^q.

Proof. It is sufficient to prove $d_n * d_n^q = e_n^{\sqrt{3q}}$ because $d_n + d_n^q = e_n$ from equation (4). For simplicity, we prove $d_n * d_n^q = e_n^{\sqrt{3q}}$ when $n = 1$.

$$\begin{aligned}
d{*}d^q &= (g + g^{q^2} + g^{q^4}) \cdot (g^q + g^{q^3} + g^{q^5}) \\
&= g^{1+q} + g^{1+q^3} + g^{1+q^5} + g^{q^2+q} + g^{q^2+q^3} + g^{q^2+q^5} + g^{q^4+q} + g^{q^4+q^3} + g^{q^4+q^5} \\
&\overset{\checkmark}{=} g^{\sqrt{3q}} + (g^q)^{\sqrt{3q}} + (g^{q^2})^{\sqrt{3q}} + (g^{q^3})^{\sqrt{3q}} + (g^{q^4})^{\sqrt{3q}} + (g^{q^5})^{\sqrt{3q}} + 3 \\
&= (g + g^q + g^{q^2} + g^{q^3} + g^{q^4} + g^{q^5})^{\sqrt{3q}} \\
&= e^{\sqrt{3q}}.
\end{aligned}$$

The third equality is derived from the series of subgroups in equation (3), that is to say, $g^{q+1} = g^{\sqrt{3q}}$ (from $<g> \lhd G_{q-\sqrt{3q}+1}$) and $g^{q^3+1} = 1$ (from $<g> \lhd G_{q^3-1}$). □

From Lemma 2, we can find two roots d_n and d_n^q by solving the quadratic formula, which are

$$\{d_n,\ d_n^q\} = \frac{e_n \pm \sqrt{e_n^2 - 4e_n^{\sqrt{3q}}}}{2}. \qquad (5)$$

Let $e_n = z \in \mathbb{F}_q$ and the roots of the quadratic equation be $\{(x, y), (y, x)\}$, where $x, y \in \mathbb{F}_q$ and $x \neq y$. Actually, $z = x + y$. Define a restoration function with input e_n, outputs $\{d_n, d_n^q\} \in \mathbb{F}_{q^2}$.

$$\mathsf{Restoration}[e_n] = \{d_n, d_n^q\}.$$

4 Efficient Method of Restoration - Finding d_n and d_n^q from e_n

As we have looked around at the previous section, we need to solve the quadratic formula described in Lemma 2 to extract d_n and d_n^q from e_n. In other words, we have to compute the square root extraction $\sqrt{e_n^2 - 4e_n^{\sqrt{3q}}}$.

In a finite field $\mathbb{F}_{r^s}$ where $r \equiv 3 \ (mod\ 4)$ and odd s, the best algorithm known [3,9] to compute a squire root executes $O(s \log_2 r)$ multiplications in $\mathbb{F}_{r^s}$. By that method, a solution of $X^2 = A$ is given by $X = A^{\frac{r^s+1}{4}}$, assume that A is a quadratic residue. Recently, Barreto et al. (c.f. [1], Section 4) presented an improvement to it. The complexity is reduced to $O(\log_2 s + \log_2 r)$ multiplications in $\mathbb{F}_{r^s}$. If the characteristic r is fixed and small compared to s, the complexity is simply $O(\log_2 s)$.

4.1 Square Root Extraction

Let $R = e_n^2 - 4e_n^{\sqrt{3q}} \in \mathbb{F}_q$. Here, $q = 3^{2k-1}$ for any inter k. As d_n or $d_n^q \notin \mathbb{F}_q$ $\sqrt{R}$ is not an element of $\mathbb{F}_q$. Thus, we can not utilize Barreto et al.'s method directly to compute square root of R even if $q \equiv 3 \ (mod\ 4)$.

Fact 2. -1 *has a square root in* $\mathbb{F}_q$ *if and only if* $q \equiv 1$ $(mod\ 4)$.

As $q \equiv 3$ $(mod\ 4)$, $\sqrt{-1} \notin \mathbb{F}_q$, but in $\mathbb{F}_{q^2}$.

Lemma 3. $\sqrt{-R} \in \mathbb{F}_q$, *where* $-R = 2e_n^2 + e_n^{\sqrt{3q}}$.

Proof. Let $\mathbb{F}_q^* =< g_1 >$. $\sqrt{g_1^n} = g_1^{n/2} \in \mathbb{F}_q$ if n is even and $\sqrt{g_1^n}$ is not in $\mathbb{F}_q$ if n is odd for $g_1^n \in \mathbb{F}_q^*$. From $(g_1^{(q-1)/2})^2 = 1$ and g_1 is a generator of $\mathbb{F}_q$, $g_1^{(q-1)/2} = -1$. We confirm easily that $(q-1)/2$ is odd if $q = 3^{2k-1}$. Then we see that $R = g_1^{n_1}$ for some odd n_1 since $\sqrt{R} \notin \mathbb{F}_q$. Hence $-R = R \cdot (-1) = g_1^{n_1+(q-1)/2}$ and $n_1 + (q-1)/2$ is even. Therefore, $\sqrt{-R} \in \mathbb{F}_q$. □

From Lemma 3, one of $\sqrt{-R}$ is $(-R)^{\frac{q+1}{4}}$ and it is efficiently computed by using the idea of Barreto et al. [1]. The basic idea is as follows.

They noticed that, if $q = 3^{2k-1}$ for some k:

$$\frac{q+1}{4} = \frac{3^{2k-1}+1}{4} = 6 \cdot \sum_{i=0}^{k-2} (3^2)^i + 1,$$

so that

$$\big(-R\big)^{(q+1)/4} = \Big[\big((-R)^2\big)^{\sum_{i=0}^{k-2}(3^2)^i}\Big]^3 \cdot (-R).$$

The quantity $\big((-R)^2\big)^{\sum_{i=0}^{k-2}(3^2)^i}$ is efficiently computed in an analogues fashion to Itoh-Teechai-Tsujii inversion [6], based on the Frobenius map in characteristic three. Let $A \in \mathbb{F}_q$. Then, one can compute $A^{\sum_{i=0}^{k-2}(3^2)^i}$ with no more than $\lfloor \log_2(k-1) \rfloor + HW(k-1) - 1$ multiplications in $\mathbb{F}_q$. Here, $\lfloor \cdot \rfloor$and $HW(\cdot)$ denote the maximum integer less than its operand and the Hamming weight of its operand respectively. Thus, we need at most $\lfloor \log_2(k-1) \rfloor + HW(k-1) + 1$ multiplications in $\mathbb{F}_q$ to compute $(-R)^{(q+1)/4}$ in total.

Next we must find $\sqrt{-1} \in \mathbb{F}_{q^2}$ to compute $\sqrt{R} = \sqrt{-R} \cdot \sqrt{-1}$. 4-th roots of unity in $\mathbb{F}_{q^2}$ are ± 1, $\pm\sqrt{-1}$. We select an element x in $\mathbb{F}_{q^2}^*$ at random then $x^{(q^2-1)/4}$ becomes any of $\pm 1, \pm\sqrt{-1}$. If $x^{(q^2-1)/4}$ is not ± 1 then it is one of square root of -1. 4-th roots of unity are also efficiently computed just by small modification of Barreto et al.'s idea [1].

$$\frac{q^2-1}{4} = \frac{3^{4k-2}-1}{4} = 2 \cdot \sum_{i=0}^{2k-2} (3^2)^i,$$

so that

$$\big(x\big)^{(q^2-1)/4} = \big(x^2\big)^{\sum_{i=0}^{2k-2}(3^2)^i}.$$

As $x \in \mathbb{F}_{q^2}$, to find a 4-th root of unity it takes on average $2 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big)$ multiplications in $\mathbb{F}_{q^2}$ using the Frobenius map in characteristic three. Thus, $6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big)$ multiplications in $\mathbb{F}_q$ as one multiplication in $\mathbb{F}_{q^2}$ takes three[1] multiplications in $\mathbb{F}_q$. Note that if q is fixed, then finding square root of -1 can be pre-computed.

4.2 Computation of d_n and d_n^q

Thanks to the equation (5) and the results of the previous section, for given $e_n \in \mathbb{F}_q$

$$\begin{aligned} \{d_n, d_n^q\} &= \frac{e_n \pm \sqrt{R}}{2} \\ &= 2 \cdot (e_n \pm \sqrt{-R} \cdot \sqrt{-1}) \\ &= 2e_n \pm (2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \cdot \sqrt{-1}. \end{aligned} \tag{6}$$

Table 1 shows the number of multiplications in $\mathbb{F}_q$ required to compute equation (6), where as customary we do not count the cost of additions and subtractions in $\mathbb{F}_q$.

Table 1. The number of multiplications in $\mathbb{F}_q$ for computation of d_n and d_n^q, where $q = 3^t$ and $t = 2k - 1$ for some integer k

Operation	# of multiplications in $\mathbb{F}_q$
e_n^2	1
$e_n^{\sqrt{3q}}$	free if $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$
$(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}}$	$\lfloor \log_2(k-1) \rfloor + HW(k-1) + 1$
$\sqrt{-1}$	$6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big)$
$(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \cdot \sqrt{-1}$	2
$2e_n \pm (2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \cdot \sqrt{-1}$	$6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) +$ $\lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$

For efficient computation of $\sqrt{3q}$-th power of e_n ($\in \mathbb{F}_q$), i.e. $e_n^{\sqrt{3q}}$, we should select q such that $\mathbb{F}_q$ has optimal normal basis (ONB) over $\mathbb{F}_3$. As $q = 3^{2k-1}$, $\sqrt{3q} = 3^k$. Thus, $\sqrt{3q}$-th power is performed by shift of coefficients when $\mathbb{F}_q$ has ONB over $\mathbb{F}_3$. However, $\mathbb{F}_q$ never has Type-I ONB over $\mathbb{F}_3$ since $2k$ is not prime. Therefore, we should check whether $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$ or not for given k. For example, we may select $k = 55, 70, 82, 89$, and 94, which satisfy that $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$.

Note that a multiplication $(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} * \sqrt{-1}$ takes two multiplications in $\mathbb{F}_q$ because $(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \in \mathbb{F}_q$ and $\sqrt{-1} \in \mathbb{F}_{q^2}$.

[1] Multiplication in $\mathbb{F}_{q^2}$ can be done using four multiplications in $\mathbb{F}_q$. These straightforward results can simply be improved to three multiplications by using a Karatsuba-like approach: to compute $(x_1, x_2) * (y_1, y_2)$ one computes $x_1 * y_1$, $x_2 * y_2$, and $(x_1 + x_2) * (y_1 + y_2)$, then $(x_1, x_2) * (y_1, y_2)$ becomes $\big((x_1 + x_2) * (y_1 + y_2) + x_2 * y_2, (x_1 + x_2) * (y_1 + y_2) + x_1 * y_1\big)$.

Theorem 2. *Given e_n for any integer n, computing d_n and d_n^q take about* $6 \cdot \big(\lfloor \log_2(2k-1)\rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1)\rfloor + HW(k-1) + 4$ *multiplications in* $\mathbb{F}_q$ *under assumption that* $\mathbb{F}_q$ *has Type-II ONB over* $\mathbb{F}_3$*.*

5 Compressed XTR Exponentiation

In this section it is shown how e_n can be computed based on e_1 and an arbitrary integer n.

Restoration - compute d_1 and d_1^q from Restoration$[e_1]$. Between $\{d_1, d_1^q\}$ choose one of them at random, denoted d'.

XTR exponentiation - compute d'_n from XTR_Exp$[d', n]$ described in section 2.2.

Compression - compute Compression$[d'_n] = d'_n + (d'_n)^q$. Actually, Compression$[d'_n] = e_n$.

At the compression step, we can easily check $d'_n + (d'_n)^q = e_n$. d' is one of $\{d_1, d_1^q\}$. If $d' = d_1$ then it is trivial because of the definition of e_n. Otherwise, i.e. $d' = d_1^q$ then $d'_n + (d'_n)^q = d_n^q + d_n$ because $d_n \in \mathbb{F}_{q^2}$, which concludes the justification of the compression step.

Denote the above XTR exponentiation over characteristic three with input e_1 and n outputs e_n as

$$\mathsf{XTR\_Exp}_3[e_1, n] = e_n.$$

Theorem 3. *Let e_1 and a positive integer $n \in Z_p$ be given. Assume that $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$. Then, computing e_n takes about* $8\log_2(n) + 6 \cdot \big(\lfloor \log_2(2k-1)\rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1)\rfloor + HW(k-1) + 4$ *multiplications in* $\mathbb{F}_q$*.*

Proof. Immediate from Theorem 2, XTR Exponentiation algorithm ([7], Algorithm 2.3.7), and Lemma 1.

5.1 Application to XTR-DH

In this section we describe XTR version Diffie-Hellman key agreement over characteristic three.

Public Parameters : $q(= 3^{2k-1})$, p, $Tr_{(q^6,q)}(g) := e$

Suppose that Alice and Bob who both have access to the XTR public key data, want to agree on a shared secret key K. This can be done using the following XTR version.

1. Alice selects at random $a \in Z_p$, uses $\mathsf{XTR\_Exp}_3[e, a] = e_a \in \mathbb{F}_q$, and sends e_a to Bob.
2. Bob receives e_a from Alice, selects at random $b \in Z_p$, uses $\mathsf{XTR\_Exp}_3[e, b] = e_b \in \mathbb{F}_q$, and sends e_b to Alice.
3. Alice receives e_b from Bob, computes $\mathsf{XTR\_Exp}_3[e_b, a] = e_{ba}$, and determines K based on $e_{ba} = Tr_{(q^6,q)}(g^{ba})$.
4. Bob uses $\mathsf{XTR\_Exp}_3[e_a, b] = e_{ab}$, and determines K based on $e_{ab} = Tr_{(q^6,q)}(g^{ab})$.

5.2 Comparison to Original XTR

In this section, we compare XTR over characteristic three to the original XTR. Let $\mathbb{XTR}$ and $\mathbb{XTR}_3$ denote the original XTR [7] and XTR over characteristic three respectively.

XTR group G_p

$\mathbb{XTR}$ - XTR group $G_p =<h>$ is a subgroup of G_{q^2-q+1}, where $h \in \mathbb{F}^*_{q^6}$.
- p and q are prime, and $q \equiv 2(\ mod\ 3)$.

$\mathbb{XTR}_3$ - XTR group $G_p =<g>$ is a subgroup of $G_{q-\sqrt{3q}+1}$, where $g \in \mathbb{F}^*_{q^6}$.
- p is prime and $q = 3^{2k-1}$.

Note that suggested lengths to provide adequate levels of security are $\log_2(q) \approx 170$ and $\log_2(p) \approx 160$.

XTR Exponentiation

$\mathbb{XTR}$ - For given $Tr_{(q^6,q^2)}(h)$ and $n \in Z_p$ computing $Tr_{(q^6,q^2)}(h^n)$ takes $8\log_2(n)$ multiplications in $\mathbb{F}_q$.
- $\mathbb{F}_{q^2}$ has Type-I ONB over $\mathbb{F}_q$.

$\mathbb{XTR}_3$ - For given $Tr_{(q^6,q)}(g)$ and $n \in Z_p$ computing $Tr_{(q^6,q)}(g^n)$ takes $8\log_2(n) + 6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$ multiplications in $\mathbb{F}_q$.
- $\mathbb{F}_{q^2}$ has Type-II ONB over $\mathbb{F}_q$.
- $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$.

Denote by P and Q the sizes of the prime p and q to be generated, respectively. To achieve security at least equivalent to 1024-bit RSA, $6Q$ should be set to about 1024, i.e., $Q \approx 170$, and P can for instance be set at 160. In $\mathbb{XTR}_3$, $k = 55$ satisfies that $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$, and also the size of Q ,where $k = 55$, is about 170. In the case when $k = 55$, the result of $6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$ is at most 101. In general, the size of n is about 160-bit. Under these conditions, $Tr_{(q^6,q)}(g^n)$ takes about 1359 multiplications in $\mathbb{F}_q$, which is only about 6% increase compared to the cost of computation of $Tr_{(q^6,q^2)}(h^n)$. If q is fixed, then finding square root of -1 can be pre-computed. Then, the computational overhead is only 1% compared to that of original XTR.

Communication Overhead. The communication overhead of XTR-DH in $\mathbb{XTR}_3$ is about *half* of XTR-DH proposed in [7] and *one six* of traditional implementations of the Diffie-Hellman protocol that are based on subgroups of multiplicative groups of finite fields, and that achieves the same level of security.

Size of Public Key Parameter. In $\mathbb{XTR}$, the public key data are q, p, and $Tr_{(q^6,q^2)}(h)$. Thus, the total length is $3Q + P$. However, in the case of $\mathbb{XTR}_3$, the public key data are $q(= 3^{2k-1})$, p, and $Tr_{(q^6,q)}(g)$, and the total length of it is $2Q + P$. If we select $Q \approx 170$ and $P \approx 160$ then the required length of public key data of $\mathbb{XTR}_3$ is reduced about 26% from that of $\mathbb{XTR}$.

References

1. P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," *Advances in Cryptology-CRYPTO 2002*, LNCS 2442, Springer, pp.354-369, 2002.
2. A.E. Brouwer, R. Pellikaan, E.R. Verheul, "Doing more with fewer bits," *Asiacrytp 1999*, LNCS 1716, pp.321-332, 1999.
3. H. Cohen, "A Course in Computational Algebraic Number Theory," Springer, 1993.
4. T.ElGamal, "A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, 31(4), pp.469-472, 1985.
5. G.Gong, L.Harn, "Public key cryptosystems based on cubic finite field extensions," *IEEE Transactions on Information Theory*, 45 (7), pp.2601-2605, 1999.
6. T. Itoh, O. Teechai and S. Tsujii, "A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases," *Information and Computation*, 78, pp.171-177, 1988.
7. A. K. Lenstra and E. R. Verheul, "The XTR Public Key System," *Advances in Cryptology-CRYPTO 2000*, LNCS 1800, Springer, pp.1-20, 2000.
8. A. K. Lenstra, "Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields," *ACISP 1997*, LNCS 1270, Springer, pp.127-138, 1997.
9. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1997.
10. C.P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, 4, pp.161-174, 1991.
11. P.Smith, C.Skinner, "A Public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms," *Asiacrytp 1994*, LNCS 917, pp.357-364, 1995.

Sliding Window Method for NTRU[*]

Mun-Kyu Lee[1,**], Jung Woo Kim[2], Jeong Eun Song[1], and Kunsoo Park[2]

[1] School of Computer Science and Engineering,
Inha University, Incheon 402-751, Korea
mklee@inha.ac.kr
[2] School of Computer Science and Engineering,
Seoul National University, Seoul 151-742, Korea

Abstract. The NTRU cryptosystem is a ring-based public key system using hard problems over lattices. There has been an extensive research on efficient implementation of NTRU operations, including recent results such as Bailey et al.'s software implementation over a resource-constrained device and Gaubatz et al.'s hardware implementation using only 3,000 gates. In this paper, we present a new algorithm to improve further the performance of NTRU. We speed up the encryption and decryption operations of NTRU up to 32% using some temporary memory, and if we can use precomputation, then the speed-up becomes up to 37%. Our method is based on the observation that specific sub-operations are repeated frequently in the underlying polynomial operations of NTRU.

1 Introduction

The NTRU cryptosystem [1] is a public key cryptosystem over polynomial rings, whose security is based on hard problems over lattices. After the introduction of NTRU encryption, a digital signature scheme using NTRU lattices, which is called NTRUsign [2], was also proposed. Since Coppersmith and Shamir [3] presented an attack against NTRU using lattice basis reduction algorithms, there have been various attempts to break NTRUencrypt [4] and NTRUsign [5,6,7]. However, none of these attacks revealed any significant weakness in the lattice problems used for NTRU [8].

On the other hand, there has been an extensive research on the efficient implementation of NTRU. Hoffstein and Silverman [9,10] proposed to use special forms of polynomials to reduce the amount of computation in NTRU while preserving its security, and Bailey et al. [11] showed that NTRU can be efficiently implemented over resource-constrained devices. Recently, Gaubatz, Kaps and Sunar [12] presented a hardware implementation of NTRU using no more than 3,000 gates, showing it is possible to use public key cryptography on sensor nodes. Now NTRU is being considered for the IEEE P1363.1 standard [13].

[*] This work was supported by grant No.R01-2006-000-10957-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

[**] Corresponding author.

In this paper, we present a method to improve further the performance of NTRU. We speed up the encryption and decryption operations of NTRU by 25 to 37%, based on the observation that specific patterns are repeated frequently in a convolution operation, which is a dominant polynomial operation of NTRU. Our contributions are as follows:

– We propose an efficient method to find such patterns, which resembles the sliding window method for exponentiation. Hence we name our method a *sliding window method for NTRU*. We show that our method is optimal in the sense that it can find the maximum number of these patterns within a given window size.
– We propose a new convolution algorithm that improves on the algorithm given in [11]. According to our experiments, the new algorithm accelerates the encryption and decryption operations of NTRU up to 32% using some temporary memory. If we can use off-line precomputation, then the speed-up becomes up to 37%.

2 Preliminaries

2.1 Convolution

Let Z be the set of integers. The polynomial ring over Z, denoted by $Z[X]$, is the set of all polynomials with coefficients in Z. We work in the quotient ring $R = Z[X]/(X^N - 1)$. An element $a \in R$ can be written as a polynomial or a vector,

$$a(X) = \sum_{i=0}^{N-1} a_i X^i = [a_0, a_1, \ldots, a_{N-1}].$$

Then multiplication of $a \in R$ and $b \in R$ can be represented as the convolution product c, which is given by $c(X) = a(X) * b(X)$ with

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} + \sum_{i=k+1}^{N-1} a_i b_{N+k-i} = \sum_{i+j \equiv k \pmod N} a_i b_j,$$

since $X^N \equiv 1 \bmod (X^N - 1)$.

In principle, this operation requires N^2 integer multiplications. However, for a typical product used by NTRU, either a or b has small coefficients, so the computation of $a * b$ can be done very fast.

2.2 The NTRU Public-Key Cryptosystem

In this section, we briefly review the NTRU cryptosystem. While there are several variants of NTRU, an improved version given in [9,11] can be described as follows:

– NTRU has three public parameters (N, p, q), where $gcd(p, q) = 1$ and $p \ll q$.
– Coefficients of polynomials are reduced mod p or q.
– The inverse of polynomial f mod q, denoted by f^{-1} mod q, is defined as the polynomial satisfying $f * f^{-1} \equiv 1 \bmod q$.

The working draft of IEEE P1363.1 standard [13] presents a few typical parameter sets for NTRU, one of which is $(N, p, q) = (251, 2, 197)$.

Key Generation. Randomly choose polynomials $F, g \in R$ with small coefficients. Then compute $f := 1 + pF$ and $h := pf^{-1} * g \bmod q$, where mod q means that every coefficient in a polynomial is reduced mod q. The private key is the polynomial f and the public key is the polynomial h.

Encryption. Let m be the polynomial representing a message. Then randomly choose a polynomial r of degree $N - 1$ with small coefficients, and compute the ciphertext $e := r * h + m \bmod q$.

Decryption. In order to decrypt e, first compute $a := e * f \bmod q$, choosing the coefficients of a to satisfy $A \le a_i < A + q$. The value of A is fixed and is determined by a simple formula depending on the other parameters. Then recover the plaintext m as $m := a \bmod p$.

Why Decryption Works. The polynomial a satisfies

$$\begin{aligned} a &\equiv e * f \bmod q \\ &\equiv (r * h + m) * f \bmod q && (\text{since } e \equiv r * h + m) \\ &\equiv pr * g + m * f \bmod q && (\text{since } h * f \equiv pg * f^{-1} * f \equiv pg) \end{aligned}$$

Consider the last polynomial $pr * g + m * f$. By an appropriate choice of parameters, one can adjust its coefficients to lie in an interval of length less than q. Hence we can recover

$$a = pr * g + m * f = pr * g + m * (1 + pF)$$

exactly, not merely modulo q. In other words, $m \equiv a \bmod p$.

2.3 Fast Convolution

The most time consuming part of NTRU encryption is computation of the convolution product $r(X) * h(X) \bmod q$. Similarly, the most time consuming part of NTRU decryption is computation of $e(X) * f(X) \bmod q$, and thus $e(X) * F(X) \bmod q$, since $e(X) * f(X) \equiv e(X) + pe(X) * F(X)$.

Note that while the coefficients in polynomials $h(X)$ and $e(X)$ are almost randomly distributed modulo q, we can control the forms of $r(X)$ and $F(X)$. Thus, $r(X)$ and $F(X)$ are usually selected to have binary coefficients, i.e., 0 or 1, so that coefficients may be computed without any multiplication. For example, if $r(X)$ is a binary polynomial with Hamming weight $HW(r)$, i.e., with $HW(r)$ ones, computation of the product $r(X) * h(X) \bmod q$ requires approximately $HW(r) \times N$ operations, where each operation is an addition plus a reduction modulo q. Therefore, if we can use $r(X)$ and $F(X)$ with low Hamming weights, the encryption and decryption procedures become very efficient. In [13], appropriate values for $HW(r)$ and $HW(F)$ are given according to the choice of public parameters (N, p, q).[1]

[1] Note that too small values of $HW(r)$ and $HW(F)$ may compromise security, since the sizes of spaces for r and f become very small.

Algorithm 1. Fast Convolution Algorithm (reproduced from [11])

```
Input: $b$ an array of $d$ locations for '1' representing the polynomial $a(X)$; $c(X)$ the
    polynomial; $N$ the number of coefficients in $a(X)$, $c(X)$.
Output: $t$ the array where $t(X) = a(X) * c(X)$.
 1: for $0 \le j < 2N$ do
 2:     $t_j \leftarrow 0$
 3: end for
 4: for $0 \le j < d$ do
 5:     for $0 \le k < N$ do
 6:         $t_{k+b[j]} \leftarrow t_{k+b[j]} + c_k$
 7:     end for
 8: end for
 9: for $0 \le j < N$ do
10:     $t_j \leftarrow (t_j + t_{j+N}) \bmod q$
11: end for
```

Bailey et al. [11] presents an efficient convolution algorithm under the assumption that one of the two input polynomials has binary coefficients. Algorithm 1 shows its simplified form, where $c(X) \in R$ is a general polynomial and $a(X) \in R$ is a binary polynomial with $HW(a) = d$. That is, $a(X)$ represents $r(X)$ and $F(X)$ in NTRU.

In Algorithm 1, line 6 repeats dN times and requires two additions each time, i.e., one for the addition of c_k and the other for the computation of index $k+b[j]$. On the other hand, line 10 repeats N times and it requires two additions and one modular reduction each time. Therefore the total number of operations of Algorithm 1 is exactly $2(d+1)N$ additions and N modular reductions.

3 Sliding Window Method for NTRU

The speed of a convolution operation, and thus the performance of NTRU encryption and decryption, can be improved significantly if some memory is available. In this section, we show the motivation for our work, and give an improved convolution algorithm, which we call the *sliding window method for NTRU*. The new algorithm is based on the observation that for a binary polynomial $a(X) \in R$ which is produced by randomly selecting $HW(a)$ ones out of N possible positions, the distribution of ones has some desirable properties.

3.1 Basic Idea

We begin by examining the structure of a convolution operation. Note that multiplication of $a \in R$ and $c \in R$ can be represented as the convolution product $t \in R$:

$$t_k = \sum_{i=0}^{k} a_i c_{k-i} + \sum_{i=k+1}^{N-1} a_i c_{N+k-i} = \sum_{i+j \equiv k \pmod N} a_i c_j .$$

Because $t \in R$ can also be written as a vector, this operation can be rewritten as the following matrix form:

$$t(X) = a(X) * c(X) = \begin{pmatrix} a_0 & a_{N-1} & a_{N-2} & \cdots & a_2 & a_1 \\ a_1 & a_0 & a_{N-1} & \cdots & a_3 & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_4 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N-1} & a_{N-2} & a_{N-3} & \cdots & a_1 & a_0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{pmatrix}.$$

We can observe that each row in the above $N \times N$ matrix is produced by rotating the previous row to right by one position.

Now we give a small example with a binary polynomial $a(X) = X + X^2 + X^5 + X^6 + X^8 + X^9$ with $N = 10, d = 6$. For simplicity, we will write a binary polynomial as a bit string throughout this section. Thus $a(X)$ will be written as 0110011011. Then $a(X) * c(X)$ can be written as

$$t(X) = a(X) * c(X) = \begin{pmatrix} 0&1&1&0&1&1&0&0&1&1 \\ 1&0&1&1&0&1&1&0&0&1 \\ 1&1&0&1&1&0&1&1&0&0 \\ 0&1&1&0&1&1&0&1&1&0 \\ 0&0&1&1&0&1&1&0&1&1 \\ 1&0&0&1&1&0&1&1&0&1 \\ 1&1&0&0&1&1&0&1&1&0 \\ 0&1&1&0&0&1&1&0&1&1 \\ 1&0&1&1&0&0&1&1&0&1 \\ 1&1&0&1&1&0&0&1&1&0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \end{pmatrix}.$$

Hence t_0 will be computed as $t_0 = c_1 + c_2 + c_4 + c_5 + c_8 + c_9$, which requires six additions.[2] Among these additions, now we concentrate on the term $c_1 + c_2$. We can see that this term also occurs in the computation of t_3 and t_7. This is because the pattern '11' is repeated three times in the binary representation of $a(X)$. To be more precise, $c_1 + c_2$ in the computation of t_0, t_3, t_7 corresponds to $\underline{11_3}$, $\underline{11_1}$, $\underline{11_2}$ in $a(X) = 0\underline{11_1}00\underline{11_2}0\underline{11_3}$, respectively. Since the term $c_1 + c_2$ occurs three times, we can compute this term only once, store it in a look-up table, and reuse it when it is required, which can reduce the number of additions by two. This reduction can also be applied to other terms related to the pattern '11'. For example, the term $c_2 + c_3$ occurs in the computation of t_1, t_4 and t_8, the term $c_3 + c_4$ in t_2, t_5 and t_9, the term $c_4 + c_5$ in t_3, t_6 and t_0, and so on. Thus the overall savings by the pattern '11' becomes $2 \times N = 20$.

Note that the above idea can be applied to other patterns such as '101', '1001', '111', etc., if only we can find these patterns in the binary representation of the polynomial so that these patterns may not share '1's. The following lemma shows a general rule for the relation between pattern occurrences and the amount of computation.

[2] For the sake of convenience in explanation, we do not consider the cost for index computation and reduction mod q here.

Lemma 1. *If a pattern containing n '1's, occurs m times in the binary representation of a polynomial with N coefficients, then we can reduce the number of integer additions by $N(m-1)(n-1)$ at the cost of memory to store N intermediate integers.*

Proof. It is straightforward since the number of integer additions related to such a pattern is reduced to $N(m+n-1)$ from Nmn. □

Therefore, patterns $p_1, p_2, \ldots, p_l$ can be used to reduce the number of integer additions by

$$N \sum_{i=1}^{l} (m_i - 1)(n_i - 1), \tag{1}$$

where p_i contains n_i '1's and it occurs m_i times. (Note that either a pattern with a single '1' or a pattern that occurs just once does not introduce any speed-up.) Thus a method to maximize (1) is crucial for fast NTRU computation. For example, for a string 01101101100, using a pattern '11' such that $0\underline{11}0\underline{11}0\underline{11}00$ will provide more saving than using '101' such that $01\underline{101}\underline{101}100$.

3.2 Finding Patterns

In this subsection, we present an efficient pattern-finding algorithm and analyze its performance. Our algorithm is based on the following facts, which will be justified throughout this subsection:

– It is sufficient to consider only the patterns that have a few (or no) zeros between two ones, i.e., '11', '101', '1001', and so on.
– There is an efficient greedy method to find such patterns, i.e., we just scan the given bit string once, marking the positions of pattern occurrences. Actually, we can show this approach is optimal.

Now we examine the first claim. Lemma 2 gives a clue to the question, "which pattern do we have to try to find?" First, the *distance* between two bit positions is defined as the difference of their indices. For example, in a string 1001, the distance between two '1's is 3.

Lemma 2. *Consider a task that chooses a bit according to a distribution where the probability that 1 is selected is p. We repeat this task independently to choose coefficients of a binary polynomial. Let Z be the distance between two neighboring occurrences of 1's. Then $Pr[Z > d] = (1-p)^d$.*

Proof. First, fix a specific nonzero position. Since we assume the independence between coefficients, we can see that $Pr[Z = t] = (1-p)^{t-1}p$. Therefore, we obtain

$$Pr[Z > d] = \sum_{i=d}^{\infty} \{(1-p)^i p\} = \frac{(1-p)^d p}{1-(1-p)} = (1-p)^d.$$ □

According to [13], binary polynomials $F(X)$ and $r(X)$ are randomly selected such that dF and dr coefficients are equal to 1, respectively, and the remaining coefficients equal to 0. While this situation is not exactly the same as the assumption in Lemma 2, the approximation $p \approx dF/N$ or $p \approx dr/N$ shows a similar behavior to a real distribution, as shown in Table 1. Although the values given in this table are experimental results according to the method of [13], they are almost the same as the values estimated from $Pr[Z = d] = (1-p)^{d-1}p$, where $p = dF/N = dr/N$.

Table 1. Distribution of distances d between two neighboring 1's in $F(X)$ and $r(X)$

parameter set	$(N, dF = dr)$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
ees251ep6	$(251, 48)$	0.191	0.156	0.126	0.101	0.083	0.066
ees347ep2	$(347, 66)$	0.191	0.155	0.125	0.102	0.082	0.067
ees397ep1	$(397, 74)$	0.186	0.152	0.124	0.101	0.082	0.067
ees491ep1	$(491, 91)$	0.186	0.152	0.123	0.101	0.082	0.067
ees587ep1	$(587, 108)$	0.184	0.151	0.123	0.101	0.082	0.067
ees787ep1	$(787, 140)$	0.177	0.147	0.120	0.099	0.081	0.067

Lemma 2 and Table 1 show that for practical parameter sets given in [13], the distance between two neighboring 1's is less than or equal to 5 with probability about 2/3. Thus we can expect that patterns such as '11', '101', '1001', '10001' and '100001' should cover a fairly large portion of $F(X)$ and $r(X)$, and provide considerable speed-up. We define these patterns as *simple patterns* with length 2 through 6, respectively.[3]

Now what we have to do is to develop an efficient method that finds the patterns '11', '101', '1001', and so on. We define the window size w, and find only the simple patterns that have up to $w - 2$ zeros between two ones. Hence there could be *separated* ones that cannot be paired with neighboring ones. For notational convenience, let p_0 be a separated '1', and let $p_1 =$ '11', $p_2 =$ '101', $p_3 =$ '1001', and so on.

We use a greedy algorithm that scans the input bit string from right to left just once. Algorithm 2 shows this algorithm. The reason why we start from right can be found in the behavior of Algorithm 1 that we use as a basis for our new convolution algorithm. That is, if we examine the construction of a specific t_j, we can see that the coefficients c_k accumulated to t_j are scanned from higher degrees to lower degrees, except one wrap-around at c_{N-1}.

We call our method a *sliding window method for NTRU*, since its bit-scanning behavior resembles that of the well-known sliding window method for exponentiation. The only differences are that there should be only up to two '1's in a single window, and the scanning is done in the opposite direction. It is easy to see that Algorithm 2 requires exactly N bit comparisons regardless of w, and the arrays $b_0, b_1, \dots, b_{w-1}$ cover all positions of '1's in x.

[3] We need not consider patterns containing more than two '1's, since these patterns occur too rare. For example, for a fixed position of '1', the probability that the next two bits are all '1's, i.e., the probability that it makes a pattern '111' is $p^2 \approx 0.037$ for $N = 251, d = 48$.

Algorithm 2. Finding Simple Patterns with Window Size $\leq w$

Input: x a binary string; N length of x; w window size.
Output: b_0 an array representing the positions for separated '1's; $b_1 \dots b_{w-1}$ arrays representing the positions for $p_1, \dots, p_{w-1}$, respectively.

```
 1: $i \leftarrow N - 1$
 2: while $i \geq 0$ do
 3:   if $x_i = 1$ then
 4:     if $p_j = x_{i-j} \dots x_i$ for some $j$ in $\{1, 2, \dots, w-1\}$, then
 5:       append $i$ to $b_j$
 6:       $i \leftarrow i - (j + 1)$
 7:     else
 8:       append $i$ to $b_0$
 9:       $i \leftarrow i - w$
10:     end if
11:   else
12:     $i \leftarrow i - 1$
13:   end if
14: end while
```

The following example illustrates our method for $w = 4, N = 28$:

$$\underline{1}00001\underline{1}\ \underline{1001}000\underline{1001}0\underline{101}0\underline{11}000\underline{1}. \tag{2}$$

The arrays b_0 through b_3 will be as follows:

$$b_0 = [27, 5, 0], b_1 = [23], b_2 = [20], b_3 = [16, 9]. \tag{3}$$

Theorem 1. *Algorithm 2 is an optimal algorithm to find the maximum number of simple patterns with length $\leq w$ in a bit string.*

Proof. Note that if there is an interval containing $w - 1$ or more consecutive zeros, then there cannot be any simple pattern with length $\leq w$ that overlaps this interval. Therefore these zero intervals partition the input string x into many segments, and the distance of two neighboring ones that belong to a same segment should always be less than $w - 1$. Now we only have to show that within a single segment, the greedy algorithm is optimal, which is straightforward since we can find k simple patterns in a segment with $2k$ or $2k + 1$ ones. □

We remark that although Algorithm 2 is an optimal algorithm for a linear bit string, we may find one more simple pattern by merging the first and last '1's in the string if we can deal with a circular string.

3.3 New Convolution Algorithm

If the positions for simple patterns are given by Algorithm 2, then Algorithm 3 can be used to accelerate a convolution operation. Algorithm 3 can be viewed as an improved version of Algorithm 1, and it is a sliding window method using

Algorithm 3. Sliding Window Method for Fast Convolution

```
Input: $b_0, \dots, b_{w-1}$, where $b_i$ is an array of $d_i$ positions of $p_i$ from the polynomial
    $a(X)$; $c(X)$ the polynomial; $N$ the number of coefficients in $a(X), c(X)$; $w$ window
    size.
Output: $t$ the array where $t(X) = a(X) * c(X)$.
 1: for $0 \le j < w - 1$ do
 2:   $c_{j+N} \leftarrow c_j$
 3: end for
 4: for $1 \le i \le w - 1$ do
 5:   for $0 \le j < N$ do
 6:     $T_i[j] \leftarrow c_j + c_{j+i}$
 7:   end for
 8: end for
 9: for $0 \le j < 2N$ do
10:   $t_j \leftarrow 0$
11: end for
12: for $0 \le j < d_0$ do
13:   for $0 \le k < N$ do
14:     $t_{k+b_0[j]} \leftarrow t_{k+b_0[j]} + c_k$
15:   end for
16: end for
17: for $1 \le i \le w - 1$ do
18:   for $0 \le j < d_i$ do
19:     for $0 \le k < N$ do
20:       $t_{k+b_i[j]} \leftarrow t_{k+b_i[j]} + T_i[k]$
21:     end for
22:   end for
23: end for
24: for $0 \le j < N$ do
25:   $t_j \leftarrow (t_j + t_{j+N}) \bmod q$
26: end for
```

precomputation tables. Lines 1 through 8 is the precomputation stage, and it requires $2(w-1)N + (w-1)$ integer additions including index computation. On the other hand, lines 9 through 26 is the convolution stage which requires $2(d_0 + d_1 + \cdots + d_{w-1} + 1)N$ additions and N modular reductions. Thus the total amount of computation of Algorithm 3 is $2(d_0 + d_1 + \cdots + d_{w-1} + w)N + (w-1)$ additions and N modular reductions, and the total amount of temporary memory for the precomputation table is $N(w-1)$ integers. Recall that Algorithm 1 requires $2(d+1)N$ additions and N modular reductions. Since $d_0 + d_1 + \cdots + d_{w-1} + w$ is much smaller than $d+1$, we can expect a significant speed-up by Algorithm 3. For example, if we use parameters $N = 251, d = dF = dr = 48$, then $d + 1 = 49$ and $d_0 + d_1 + \cdots + d_{w-1} + w = 42.399, 38.802, 36.748, 35.707, 35.171, 35.050$ for $w = 2, 3, 4, 5, 6, 7$, respectively, according to our experiment. We also see that large values for w are not so attractive since their amount of computation is almost the same as that of smaller w, while the amount of required memory is almost proportional to w. Hence we decide to fix $w = 5$.

4 Experimental Results

Now we present our experimental results for $w = 5$ with various parameter sets given in [13]. Table 2 shows the performance of NTRU encryption and decryption operations over a Pentium IV 3.0GHz CPU with 1.0GB RAM. We used C language and Microsoft Visual Studio .NET 1.0 environment. The third and fourth columns of this table represent the required time to perform an encryption and a decryption, respectively, using the original convolution algorithm (Algorithm 1). The fifth and seventh columns represent the required time when we use the improved algorithm. We can see that the new algorithm accelerates these operations by 25 to 32%. Note that this result is consistent with the values that we can estimate from the analysis given in Section 3.3 assuming a convolution operation consumes most of the computation time for encryption or decryption. That is, for $w = 5$, the gain is estimated as $(49 - 35.707)/49 \approx 27.13\%$.

Note that in some situation, the sender might know the identity of the recipient in advance, or she might send messages frequently to the same recipient. In this case, information related to the recipient's public key can be preprocessed. By setting $a(X) \leftarrow r(X)$ and $c(X) \leftarrow h(X)$, the values $T_i[j]$ in Algorithm 3 can be precomputed, i.e., lines 1 through 8 can be performed off-line. In this case, $T_i[j]$'s are not any more in a temporary memory, but they should be stored in a precomputation table. By this precomputation, we can further reduce the amount of on-line computation, which is given in the sixth column of Table 2. Now the performance gain over Algorithm 1 becomes 33 to 37%. These values are also consistent with the estimation from Section 3.3, i.e., $(49 - 35.707 + 5 - 1)/49 \approx 35.29\%$.

Table 2. Timings for various NTRU operations with $w = 5$ (μsec)

parameter set	$(N, p, q, dF = dr)$	Using Alg. 1		Using Alg. 3			
		Enc.	Dec.	Enc.1	Enc.2	Dec.	Memory†
ees251ep6	$(251, 2, 197, 48)$	1043	1045	766	683	783	1004
ees347ep2	$(347, 2, 269, 66)$	1937	2001	1380	1260	1392	1388
ees397ep1	$(397, 2, 307, 74)$	2510	2523	1783	1604	1796	1588
ees491ep1	$(491, 2, 367, 91)$	3760	3796	2649	2530	2650	1964
ees587ep1	$(587, 2, 439, 108)$	5327	5379	3685	3517	3742	2348
ees787ep1	$(787, 2, 587, 140)$	9343	9419	6420	6107	6408	3148

†Number of integers to be stored, i.e., $N(w - 1)$

5 Discussion

We proposed a method to speed up NTRU operations by reusing sub-operations that appear frequently. Our experiments show that several kilobytes of memory is sufficient to accelerate NTRU encryption and decryption by 25 to 37%.

The IEEE draft standard [13] proposes to use two classes of polynomials for r or F. The first one is to use binary polynomials with predefined Hamming weight, which is explained in Section 2.3. The second one is to use a product-form polynomial, i.e., $r = r_1 * r_2 + r_3$ or $F = F_1 * F_2 + F_3$, where $r_1, r_2, r_3, F_1, F_2, F_3$ are binary polynomials with much smaller Hamming weight than those of r and F in the first class. We remark that our sliding window method is applied only to the first case, and it is evaluated to be still slower by about 20–30% than the second case without the window method, according to our analysis. Therefore, it could be an interesting research topic to improve the convolution algorithm for product-form polynomials.

We performed the above experiments and comparison on a Pentium IV processor. However, note that speed-ups are more critical on resource-constrained devices such as a Mica-Z mote with ATmega128 microcontroller. Therefore, implementation over such devices is necessary for complete analysis of our algorithm.

References

1. Hoffstein, J., Pipher, J., Silverman, J.: NTRU: A ring-based public key cryptosystem. In: Algorithmic Number Theory – ANTS III. Volume 1423 of LNCS., Springer (1998) 267–288
2. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J., Whyte, W.: NTRUSIGN: Digital signatures using the NTRU lattice. In: CT-RSA 2003. Volume 2612 of LNCS., Springer (2003) 122–140
3. Coppersmith, D., Shamir, A.: Lattice attacks on NTRU. In: Eurocrypt 97. Volume 1233 of LNCS., Springer (1997) 52–61
4. Howgrave-Graham, N., Nguyen, P., Pointcheval, D., Proos, J., Silverman, J., Singer, A., Whyte, W.: The impact of decryption failures on the security of NTRU encryption. In: Crypto 2003. Volume 2729 of LNCS., Springer (2003) 226–246
5. Gentry, C., Jonsson, J., Stern, J., Szydlo, M.: Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In: Asiacrypt 2001. Volume 2248 of LNCS., Springer (2001) 1–20
6. Gentry, C., Szydlo, M.: Cryptanalysis of the revised NTRU signature scheme. In: Eurocrypt 2002. Volume 2332 of LNCS., Springer (2002) 299–320
7. Nguyen, P., Regev, O.: Learning a parallelepiped: cryptanalysis of GGH and NTRU signatures. In: Eurocrypt 2006. Volume 4004 of LNCS., Springer (2006) 271–288
8. Gama, N., Howgrave-Graham, N., Nguyen, P.: Symplectic lattice reduction and NTRU. In: Eurocrypt 2006. Volume 4004 of LNCS., Springer (2006) 233–253
9. Hoffstein, J., Silverman, J.: Optimizations for NTRU. In: Proceedings of Public-Key Cryptography and Computational Number Theory. (2000)
10. Hoffstein, J., Silverman, J.: Random small Hamming weight products with applications to cryptography. Discrete Applied Mathematics **130** (2003) 37–49
11. Bailey, D.V., Coffin, D., Elbirt, A., Silverman, J.H., Woodbury, A.D.: NTRU in constrained devices. In: Cryptographic Hardware and Embedded Systems – CHES 2001. Volume 2162 of LNCS., Springer (2001) 262–272
12. Gaubatz, G., Kaps, J.P., Sunar, B.: Public key cryptography in sensor networks-revisited. In: ESAS 2004. Volume 3313 of LNCS., Springer (2004) 2–18
13. IEEE P1363.1/D8: Draft standard for public-key cryptographic techniques based on hard problems over lattices (2006)

Efficient Certificateless Signature Schemes

Kyu Young Choi, Jong Hwan Park, Jung Yeon Hwang, and Dong Hoon Lee

Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea
{young,decartian,videmot}@cist.korea.ac.kr
donghlee@korea.ac.kr

Abstract. Recently, in order to eliminate the use of certificates in certified public key cryptography and the key-escrow problem in identity based cryptography, the notion of certificateless public key cryptography was introduced. In this paper, to construct an efficient certificateless signature (CLS) scheme, we present a new approach compactly and orthogonally combining short signatures using bilinear maps. Our approach is conceptually simple but effective to improve efficiency greatly. In the proposed CLS scheme a full private key of a user is a single group element and signature verification requires only one pairing operation. In addition, our CLS scheme has a flexible structure which can be easily extended to a certificateless signature scheme with additional properties such as certificateless ring and blind signature schemes.

1 Introduction

In a traditional public key cryptography (PKC), a random public key of a user is associated with the user by a certificate, that is, a signature of trusted Certificate Authority (CA) on the public key. Inevitably this feature causes CA to require a large amount of storage and computing time managing the certificates [8]. To simplify the certificate management process, Shamir introduced the concept of identity based cryptography (ID-PKC) where the certificate of a random public key does not be needed any more since publicly known information such as e-mail address is used as user's public key [15]. However, an inherent problem of ID-PKC is that a Key Generation Center (KGC) generates any user's private key using a master-key of KGC. Obviously a malicious KGC is able to forge the signature of any signer. This is called "key escrow" problem. In 2003, Al-Riyami and Paterson introduced the concept of certificateless public key cryptography (CL-PKC) which eliminates the use of certificates in PKC and solve the key escrow problem in ID-PKC [1]. The basic idea of CL-PKC is to construct a public/private key pair for a user by combining a master key of KGC with a random secret value generated by the user. In this paper we concentrate on a certificateless signature (CLS) scheme.

Despite of the usefulness of a CLS scheme, it is not easy to construct a secure and efficient CLS scheme because the construction of a CLS scheme conceptually involves mechanisms to authenticate an identity of a user, the public key of the user, and a message to be signed at the same time. For the security model of a CLS scheme reflecting such authentication mechanisms, unlike that of an ordinary signature scheme, we should consider two types of forgers, Type I and Type II forgers : A Type I forger represents a normal third party attacker who has no access to the master key but is allowed to replace public keys of users. A Type II forger represents a malicious KGC who is equipped with the master key but is not able to replace public keys. However, although many researches on a CLS scheme [1,17,12,9,10,20,11,6] are performed, only few schemes [10,20] are known to be secure against these forgers.

Naturally, such complicated authentication mechanisms should be a critical consideration to design an efficient CLS scheme. To improve efficiency it would be desirable to integrate the functionalities of authentication imbedded in a CLS scheme compactly while maintaining security.

Our Results. In the paper, to construct an efficient CLS scheme, we first present novel combinations of short signature schemes using a bilinear group: In the key setup phase, the Boneh-Shacham-Lynn short signature [5] or Boneh-Boyen's short signature [2] is used for KGC to generate a signature, that is, a partial private key corresponding to an identity of a user. The Boneh-Boyen short signature [2] is used for the user to generate a full private key, which plays a crucial role of a private *self-certificate* on the public-key of his/her choice, using the previous partial private key. In our scheme, signature verification requires only one pairing operation, compared to at least four pairing operations in the previous works [10,12,20], and signature generation requires no pairing operation. Moreover, a full private key for a user, which is computed by applying two short signature schemes sequentially, is just a single group element.

The compact feature of a full private key of a user, which aggregates two signatures, provides the minimum loss against key exposure. In other words, even if an adversary obtains a full private key, he cannot extract the partial private key from the full private key which is a signature on the partial private key. Because the partial private key is used as a long-lived key this provides a proactive property such that the user's public key can be replaced periodically.

We show that our CLS schemes are provably secure in the random oracle model. For security model we consider a realistic model where a Type I forger is not allowed to obtain a valid signature for the public key replaced by the forger. In practical environments, it is too strong to assume that the signer knows the private key associated with the replaced public key (by others). This model was already developed in several recent works [11,20].

Finally our method provides flexibility for extending to CLS schemes with additional properties such as certificateless blind and ring signature schemes. In fact, applying a similar method in [18,19,7] to our schemes we can directly construct certificateless ring and blind signature schemes.

Related Works. The first CLS scheme was proposed by Al-Riyami and Paterson [1]. Unfortunately, it was found insecure against a Type I forger by Huang et al. [10]. They also proposed a CLS scheme and proved its security in the random oracle model. In [17], Yum and Lee proposed a generic construction of CLS. However, Hu et al. presented that their construction is insecure against a Type I forger and improved it. Gorantla and Saxena [9] proposed an efficient CLS scheme. However, it was also found insecure against a Type I forger by Cao et al. [6]. In [12], Li et al. proposed a CLS scheme. It seems to be secure but a security analysis for the scheme was not formalized. Recently, Zhang et al. [20] proposed a CLS scheme and showed its security in the random oracle model.

Organization. The rest of this paper is organized as follows. In Section 2, we describe some fundamental backgrounds and define our security model for a CLS scheme. In Section 3 we propose an efficient CLS scheme and its security proofs. In Section 4 we propose a CLS scheme with a pairing operation and its security proofs. In Section 5, we analyze the performances of the proposed CLS schemes. In Section 6, we present extension of our CLS schemes. We conclude the paper in Section 7.

2 Preliminaries

We review some fundamental backgrounds required in this paper, namely bilinear pairing, certificateless signature scheme.

2.1 Bilinear Pairings and Some Problems

Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group of same order q. We assume that the discrete logarithm problems (DLP) in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are intractable.

Admissible Bilinear Map. We call $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ an *admissible bilinear* map if it satisfies the following properties:

- Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$.
- Non-degenerancy: There exists $P \in \mathbb{G}_1$ such that $e(P, P) \neq 1$.
- Computability: There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

The modified Weil and Tate pairings in elliptic curve are examples of the admissible bilinear maps. We consider following problems in the group $\mathbb{G}_1$.

Computational Diffie-Hellman (CDH) problem: The CDH problem is to compute abP when given P, aP and bP for some $a, b \in \mathbb{Z}_q^*$.

Inverse Computational Diffie-Hellman (ICDH) problem: The ICDH problem is to compute $a^{-1}P$ when given P and aP for some $a \in \mathbb{Z}_q^*$.

Modified Inverse Computational Diffie-Hellman (mICDH) problem: The mICDH problem is to compute $(a + b)^{-1}P$ when given b, P and aP for some $a, b \in \mathbb{Z}_q^*$.

The CDH, ICDH and mICDH problems are polynomial time equivalent [16]. We assume that the CDH, ICDH and mICDH problems in $\mathbb{G}_1$ are intractable. That is, there is no polynomial time algorithm solving these problems with non-negligible probability.

2.2 Certificateless Signature Scheme

We briefly recall a formal definition of a certificateless signature scheme [10]. The CLS scheme is specified by seven polynomial time algorithms.

Setup: This algorithm takes a security parameter k as input and returns the system parameters `params` and a secret master key `master-key`.

Partial-Private-Key-Extract: This algorithm takes `params`, `master-key` and a user's identity ID as input. It returns a partial private key D_{ID} corresponding to the user.

Set-Secret-Value: This algorithm takes the security parameter k and a user's identity ID as input. It returns the user's secret value x_{ID}.

Set-Public-Key: This algorithm takes a user's secret value x_{ID} as input. It returns the user's public key PK_{ID}.

Set-Private-Key: This algorithm takes a user's partial private key D_{ID} and public key PK_{ID}, and his secret value x_{ID} as input. It returns the user's full private key SK_{ID}.

Sign: This algorithm takes `params`, a message m, and a user's full private key SK_{ID} as input. It returns a signature σ.

Verify: This algorithm takes `params`, a message m, a user's identity ID, a public key PK_{ID}, and a signature σ as input. It returns 0 or 1. With output value 1, we say that σ is a valid signature of a message m.

The **Setup** and **Partial-Private-Key-Extract** algorithms are performed by a Key Generation Center (KGC). Once a partial private key is given to a user via secure channel, the user runs the **Set-Secret-Value** algorithm and chooses a secret value to generate its own public/private key pair.

The security model of CLS is different from that of a normal signature scheme. As defined in [1,10,20], we should consider two types of forger for a CLS scheme, a Type I forger $\mathcal{F}_I$ and a Type II forger $\mathcal{F}_{II}$. The forger $\mathcal{F}_I$ represents a normal third party attacker against the CLS scheme. That is, $\mathcal{F}_I$ is not allowed to access to the master-key but $\mathcal{F}_I$ may request public keys and replace public keys with values of its choice. The forger $\mathcal{F}_{II}$ represents a malicious KGC who generates partial private key of users. The forger $\mathcal{F}_{II}$ is allowed to have access to the master-key but not replace a public key. We consider two games against the Type I and Type II forgers as follows.

Game I. The first game is performed between a challenger $\mathcal{C}$ and the Type I forger $\mathcal{F}_I$ for a certificateless signature scheme $\mathit{\Pi}$ as follows.

- **Initialization:** $\mathcal{C}$ runs **Setup** algorithm and generates a master secret key `master-key`, public system parameters `params`. $\mathcal{C}$ keeps `master-key` secret and then gives `params` to $\mathcal{F}_I$. Note that $\mathcal{F}_I$ does not know the master key `master-key`.
- **Queries:** $\mathcal{F}_I$ may adaptively issue the following queries to $\mathcal{C}$.
  - $\mathsf{ExtrPartSK}(ID)$: When $\mathcal{F}_I$ requests the partial private key for a user with identity ID, $\mathcal{C}$ responds the user's partial private key D_{ID} running **Partial-Private-Key-Extract** algorithm.
  - $\mathsf{ExtrFullSK}(ID)$: When $\mathcal{F}_I$ requests the full private key for a user with identity ID, $\mathcal{C}$ responds the user's full private key SK_{ID} running **Partial-Private-Key-Extract**, **Set-Secret-Value** and **Set-Private-Key** algorithms.
  - $\mathsf{ReqestPK}(ID)$: When $\mathcal{F}_I$ requests the public key for a user with identity ID, the challenger $\mathcal{C}$ responds the user's public key PK_{ID} running **Set-Secret-Value** and **Set-Public-Key** algorithms.
  - $\mathsf{RepalcePK}(ID)$: $\mathcal{F}_I$ can replace the original public key PK_{ID} to a new public key PK'_{ID} chosen by him.
  - $\mathsf{SIGN}(m, ID)$: When $\mathcal{F}_I$ requests a signature on a message m for a user with identity ID, the challenger $\mathcal{C}$ responds a valid signature σ for m running **Sign** algorithm with the matching public key PK_{ID} for ID. If the public key PK_{ID} has been replaced earlier by $\mathcal{F}_I$, then $\mathcal{C}$ cannot know the corresponding private key SK_{ID} and thus the signing oracle's answer may not be correct. In such case, to correctness of the signing oracle's answer, we assume that $\mathcal{F}_I$ additionally submits the corresponding secret information to the signing oracle.
- **Output:** Eventually, $\mathcal{F}_I$ outputs (ID_t, m_t, σ_t), where ID_t is the identity of a target user, m_t is a message, and σ_t is a signature for m_t. $\mathcal{F}_I$ wins the game if
  1. $\mathsf{ExtrPartSK}(ID_t)$, $\mathsf{ExtrFullSK}(ID_t)$, and $\mathsf{SIGN}(m_t, ID_t)$ queries have never been queried.
  2. **Verify**$(\texttt{params}, m_t, ID_t, PK_t, \sigma_t)$ outputs 1, that is, the signature σ_t for a message m_t is valid under PK_t which may be replaced by $\mathcal{F}_I$.

We define $\mathsf{Succ}_{\mathcal{F}_I}^{\Pi}$ to be the success probability that $\mathcal{F}_I$ wins in the above game.

Game II. The second game is performed between a challenger $\mathcal{C}$ and the Type II forger $\mathcal{F}_{II}$ for a certificateless signature scheme Π as follows.

- **Initialization:** $\mathcal{C}$ runs **Setup** algorithm and generates a master secret key `master-key`, public system parameters `params`. The challenger $\mathcal{C}$ gives public `params` and secret `master-key` to $\mathcal{F}_{II}$.
- **Queries:** $\mathcal{F}_{II}$ may adaptively issue the following queries to $\mathcal{C}$.
  - $\mathsf{ExtrFullSK}(ID)$: When $\mathcal{F}_{II}$ requests the full private key for a user with identity ID, $\mathcal{C}$ responds the user's full private key SK_{ID} running **Partial-Private-Key-Extract**, **Set-Secret-Value** and **Set-Private-Key** algorithms.

  - $\mathsf{RequestPK}(ID)$: When $\mathcal{F}_{II}$ requests the public key for a user with identity ID, the challenger $\mathcal{C}$ responds the user's public key PK_{ID} running **Set-Secret-Value** and **Set-Public-Key** algorithms.
  - $\mathsf{SIGN}(m, ID)$: When $\mathcal{F}_{II}$ requests a signature on a message m for a user with identity ID, the challenger $\mathcal{C}$ responds a valid signature σ for m running **Sign** algorithm with matching public key PK_{ID} for ID.
- **Output:** Eventually, $\mathcal{F}_{II}$ outputs (ID_t, m_t, σ_t), where ID_t is the identity of a target user, m_t is a message, and σ_t is a signature for m_t. $\mathcal{F}_{II}$ wins the game if
  1. $\mathsf{ExtrFullSK}(ID_t)$ and $\mathsf{SIGN}(m_t, ID_t)$ queries have never been issued.
  2. **Verify**$(\texttt{params}, m_t, ID_t, \sigma_t)$ outputs 1, that is, the signature σ_t for a message m_t is valid under PK_t.

We define $\mathsf{Succ}^{\Pi}_{\mathcal{F}_{II}}$ to be the success probability that $\mathcal{F}_{II}$ wins in the above game. Note that $\mathcal{F}_{II}$ does not need additional extraction query to obtain partial private keys since the master key `master-key` is given to $\mathcal{F}_{II}$.

Definition 1. *We say that a certificateless signature scheme Π is existentially unforgeable against chosen message attacks, if for any polynomially bounded forgers $\mathcal{F}_I$ and $\mathcal{F}_{II}$, the success probabilities of both $\mathcal{F}_I$ and $\mathcal{F}_{II}$ are negligible. In other words,*

$$\mathsf{Succ}^{\Pi}_{\mathcal{F}_I}(k) < \epsilon \quad and \quad \mathsf{Succ}^{\Pi}_{\mathcal{F}_{II}}(k) < \epsilon$$

where k is the security parameter.

3 New Certificateless Signature Scheme

In this section, we propose a new efficient CLS scheme and prove the security of the proposed scheme. We denote this CLS scheme by eCLS.

3.1 Our Construction

Setup. To generate system parameters and master key, run as follows:
  1. Generate $(\mathbb{G}_1, \mathbb{G}_2, e)$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order q and e is an admissible bilinear map.
  2. Choose a random $s \in \mathbb{Z}_q^*$ and a generator P of $\mathbb{G}_1$. Compute $P_{pub} = sP$.
  3. Choose three cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \mathbb{G}_1 \to \mathbb{Z}_q^*$, and $H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$.

  Return the private `master-key`$= s$ and the system parameters `params`$= \{e, \mathbb{G}_1, \mathbb{G}_2, q, P, P_{pub}, H_1, H_2, H_3\}$. We assume that `params` is available to all users.

Partial-Private-Key-Extract. On input `params`, `master-key`, and identity ID_A of user A. Compute $Q_A = H_1(ID_A)$ and return a partial private key $D_A = sQ_A$ for user A.

Set-Secret-Value. On input k and ID_A, choose a random value $x_A \in \mathbb{Z}_q^*$ and return x_A as A's secret value.

Set-Public-Key. On input `params` and x_A, compute $R_A = x_A P$ and return the public key $PK_A = R_A$.

Set-Private-Key. On input x_A, R_A, and D_A. Compute $y_A = H_2(R_A)$ and $S_A = \frac{1}{x_A + y_A} D_A$. Return the (full) private key $SK_A = S_A$.

Sign. On input `params`, ID_A, S_A, and a message m, perform the following steps:

1. Choose a random $r \in \mathbb{Z}_q^*$.
2. Compute $U = rQ_A = rH_1(ID_A)$.
3. Set $h = H_3(m, U)$.
4. Compute $V = (r + h)S_A$.
5. Return $\sigma = (U, V)$ as the signature on the message m.

Verify. On input `params`, ID_A, R_A, m, and $\sigma = (U, V)$. Compute $Q_A = H_1(ID_A)$, $y_A = H_2(R_A)$, and $h = H_3(m, U)$. Check if $e(V, R_A + y_A P) = e(U + hQ_A, P_{pub})$ holds. If the equation holds, it outputs 1, otherwise 0.

We can easily show that our CLS scheme satisfies completeness property as follows:

$$\begin{aligned} e(V, R_A + y_A P) &= e((r+h)S_A, x_A P + y_A P) \\ &= e((r+h)(x_A + y_A)^{-1} sQ_A, (x_A + y_A)P) \\ &= e((r+h)sQ_A, P) \\ &= e((rQ_A + hQ_A, sP) = e(U + hQ_A, P_{pub}). \end{aligned}$$

3.2 Security Analysis

Theorem 1. *Our certificateless signature scheme* eCLS *is existentially unforgeable against a Type I forger in random oracle model under the CDH assumption.*

Proof. Suppose there exists a forger $\mathcal{F}_I$ which has advantage in attacking our CLS scheme eCLS. We want to build an algorithm $\mathcal{C}$ that uses $\mathcal{F}_I$ to solve the CDH problem. $\mathcal{C}$ receives a CDH instance (P, aP, bP) for randomly chosen $a, b \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$. Its goal is to compute abP. $\mathcal{C}$ runs $\mathcal{F}_I$ as a subroutine and simulates its attack environment. $\mathcal{C}$ sets $P_{pub} = aP$ where a is the master key, which is unknown to $\mathcal{C}$, and gives system parameters to $\mathcal{F}_I$. Without loss of generality, we assume that any extraction (ExtrPartSK, RequestPK, ExtrFullSK) and signature (SIGN) queries are preceded by H_1 query, and the SIGN and ExtrFullSK queries are preceded by RequestPK query. To avoid collision and consistently respond to these queries, $\mathcal{C}$ maintains four lists $L_{H_1}, L_{H_2}, L_{H_3}, L_K = \{\langle ID, PK_{ID}, x_{ID}, c(= 0 \text{ or } 1)\rangle\}$ which are initially empty. $\mathcal{C}$ then simulates the oracle queries of $\mathcal{F}_I$ as follows:

- H_1 query: Suppose $\mathcal{F}_I$ makes at most q_{H_1} queries to H_1 oracle. First, $\mathcal{C}$ chooses $j \in [1, q_{H_1}]$ randomly. When $\mathcal{F}_I$ makes an H_1 query on ID_i where $1 \le i \le q_{H_1}$, if $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ returns $Q_{ID_i} = bP$ and adds $\langle ID_i, Q_{ID_i}, k_i = \perp\rangle$ to L_{H_1}. Otherwise $\mathcal{C}$ picks a random $k_i \in \mathbb{Z}_q^*$ and returns $Q_{ID_i} = k_i P$, and adds $\langle ID_i, Q_{ID_i}, k_i\rangle$ to L_{H_1}.

- H_2 query: When $\mathcal{F}_I$ makes this query on PK_{ID_i}, if the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns y_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- H_3 query: When $\mathcal{F}_I$ makes this query on (m_i, U_i), if the list L_{H_3} contains $\langle m_i, U_i, h_i\rangle$, $\mathcal{C}$ returns h_i. Otherwise $\mathcal{C}$ picks a random $h_i \in \mathbb{Z}_q^*$ and returns h_i, and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3}.
- $\mathsf{ExtrPartSK}(ID_i)$ query: When $\mathcal{F}_I$ makes this query on ID_i, if $ID_i \neq ID^*$, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ in L_{H_1}, and returns $D_{ID_i} = k_i aP$. Otherwise $\mathcal{C}$ outputs FAIL and aborts the simulation.
- $\mathsf{RequestPK}(ID_i)$ query: When $\mathcal{F}_I$ makes this query on ID_i, if the list L_K contains $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ returns PK_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $x_{ID_i} \in \mathbb{Z}_q^*$. Then $\mathcal{C}$ returns $PK_{ID_i} = x_{ID_i}P$ and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}, 1\rangle$ to L_K.
- $\mathsf{ExtrFullSK}(ID_i)$ query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ performs as follows:
  - If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}} k_i aP$.
  - If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}} k_i aP$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- $\mathsf{ReplacePK}(ID_i, PK'_{ID_i})$ query: When $\mathcal{F}_I$ makes this query on (ID_i, PK'_{ID_i}), $\mathcal{C}$ performs as follows:
  - If the list L_K contains $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ sets $PK_{ID_i} = PK'_{ID_i}$ and $c = 0$.
  - If the list L_K does not contain $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ makes a $\mathsf{RequestPK}$ query on ID_i itself. Then $\mathcal{C}$ sets $PK_{ID_i} = PK'_{ID_i}$ and $c = 0$.
- $\mathsf{SIGN}(m, ID_i)$ query: When $\mathcal{F}_I$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in L_{H_1} and L_K, respectively. Then $\mathcal{C}$ performs as follows:
  - If $c = 1$, $\mathcal{C}$ picks two random $r_i, h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2}. If it does not exist, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. $\mathcal{C}$ computes $U_i = r_i(x_{ID_i} + y_{ID_i})P - h_i Q_{ID_i}$ and $V_i = r_i aP$. $\mathcal{C}$ then returns (U_i, V_i) and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).
  - If $c = 0$, $\mathcal{C}$ gets additionally information x'_{ID_i} from $\mathcal{F}_I$. Using the x'_{ID_i}, $\mathcal{C}$ then simulates as in the above case ($c = 1$).

Eventually, $\mathcal{F}_I$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \neq ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle m_t, U_t, h_t\rangle$ in L_{H_3}. Then by replays of $\mathcal{C}$ with the same random tape but different choices of H_3 (it is to apply the 'forking' technique formalized in [14]), $\mathcal{C}$ gets another valid signature tuple $(ID_t, m_t, h'_t, \sigma_t = (U_t, V'_t))$ such that $h_t \neq h'_t$. $\mathcal{C}$ finds $\langle ID_t, PK_{ID_t}, x_{ID_t}, c\rangle$ in L_K. If $c = 0$, that is, $\mathcal{F}_I$ generated a public/private key pair and replaced the public key of ID_t. In such case, as the proof in [10], we assume that $\mathcal{C}$ keeps track of the public/private key pair generated by $\mathcal{F}_I$. Hence, after $\mathcal{C}$ finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2}, he can compute as follows:

$$(x_{ID_t} + y_{ID_t})\frac{V_t - V_t'}{h_t - h_t'} = SK_{ID_t} = abP.$$

Therefore, if a Type I forger who can break our scheme eCLS exists, then an attacker who solves the CDH problem exists. □

Theorem 2. *Our certificateless signature scheme* eCLS *is existentially unforgeable against a Type II forger in random oracle model under the mICDH assumption.*

Proof. Suppose there exists a forger $\mathcal{F}_{II}$ which has advantage in attacking our CLS scheme eCLS. We want to build an algorithm $\mathcal{C}$ that uses $\mathcal{F}_{II}$ to solve the mICDH problem. $\mathcal{C}$ receives a mICDH instance (P, aP, b) for randomly chosen $a, b \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$. Its goal is to compute $(a+b)^{-1}P$. $\mathcal{C}$ runs $\mathcal{F}_{II}$ as a subroutine and simulates its attack environment. $\mathcal{C}$ picks a random $s \in \mathbb{Z}_q^*$ and sets `master-key`$= s$. $\mathcal{C}$ then gives system parameters with `master-key` to $\mathcal{F}_{II}$. Without loss of generality, we assume that any extraction (RequestPK, ExtrFullSK) and signature (SIGN) queries are preceded by H_1 query, and the SIGN and ExtrFullSK queries are preceded by RequestPK query. To avoid collision and consistently respond to these queries, $\mathcal{C}$ maintains four lists L_{H_1}, L_{H_2}, L_{H_3}, $L_K = \{\langle ID, PK_{ID}, x_{ID}\rangle\}$ which are initially empty. $\mathcal{C}$ then simulates the oracle queries of $\mathcal{F}_{II}$ as follows:

- H_1 query: When $\mathcal{F}_{II}$ makes this query on ID_i, $\mathcal{C}$ picks a random $k_i \in \mathbb{Z}_q^*$ and returns k_iP, and adds $\langle ID_i, k_i\rangle$ to L_{H_1}.
- H_2 query: When $\mathcal{F}_{II}$ makes this query on PK_{ID_i}, if $PK_{ID_i} = aP$, $\mathcal{C}$ sets $y_{ID_i} = b$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns y_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- H_3 query: When $\mathcal{F}_{II}$ makes this query, $\mathcal{C}$ performs as in the proof of Theorem 1.
- RequestPK(ID_i) query: Suppose $\mathcal{F}_{II}$ makes at most q_{PK} queries to public key request oracle. First, $\mathcal{C}$ chooses $j \in [1, q_{PK}]$ randomly. When $\mathcal{F}_{II}$ makes a RequestPK query on ID_i, if $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ sets $PK_{ID_i} = aP$ and returns PK_{ID_i}, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i} = \perp\rangle$ to L_K. Otherwise $\mathcal{C}$ picks a random x_{ID_i} and returns $PK_{ID_i} = x_{ID_i}P$, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ to L_K.
- ExtrFullSK(ID_i) query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ performs as follows:
  - If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}}k_isP$.
  - If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}}k_isP$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.

– $\mathsf{SIGN}(m, ID_i)$ query: When $\mathcal{F}_{II}$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ picks two random r_i, $h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2} (if it does not exist, $\mathcal{C}$ makes a H_2 query on PK_{ID_i} itself). $\mathcal{C}$ computes $(U_i = r_i(PK_{ID_i} + y_{ID_i}P) - h_ik_iP$, $V_i = r_isP$ and returns (U_i, V_i), and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).

Eventually, $\mathcal{F}_{II}$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \neq ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle m_t, U_t, h_t\rangle$ in L_{H_3}. Then, as in the proof of Theorem 1, $\mathcal{C}$ gets another valid signature tuple $(ID_t, m_t, h'_t, \sigma_t = (U_t, V'_t))$ such that $h_t \neq h'_t$. Since $\mathcal{C}$ knows the master key s, after $\mathcal{C}$ finds $\langle ID_t, k_t\rangle$ in L_{H_1}, he can compute as follows:

$$\frac{V_t - V'_t}{k_ts(h_t - h'_t)} = \frac{1}{a+b}P.$$

Therefore, if a Type II forger who can break our scheme eCLS exists, then an attacker who solves the CDH problem exists. □

4 Certificateless Signature Scheme with a Pairing Operation

In this section, we propose a CLS scheme with a pairing operation, and prove the security of the proposed scheme. We denote this CLS scheme by oCLS.

4.1 Our Construction

Setup. To generate system parameters and master key, run as follows:

1. Generate $(\mathbb{G}_1, \mathbb{G}_2, e)$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order q and e is an admissible bilinear map.
2. Choose a random $s \in \mathbb{Z}_q^*$ and a generator P of $\mathbb{G}_1$. Compute $P_{pub} = sP$ and $g = e(P, P)$.
3. Choose three cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_1 \to \mathbb{Z}_q^*$, and $H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$.

Return the private `master-key`$=$ s and the system parameters `params`$=$ $\{e, \mathbb{G}_1, \mathbb{G}_2, q, g, P, P_{pub}, H_1, H_2, H_3\}$. We assume that `params` is available to all users.

Partial-Private-Key-Extract. On input `params`, `master-key`, and identity ID_A of user A. Compute $q_A = H_1(ID_A)$ and return a partial private key $D_A = \frac{1}{s+q_A}P$ for user A.

Set-Secret-Value. On input k and ID_A, choose a random value $x_A \in \mathbb{Z}_q^*$ and return x_A as A's secret value.

Set-Public-Key. On input `params` and x_A, compute $Q_A = P_{pub} + H_1(ID_A)P$ and $R_A = x_AQ_A$. Return the public key $PK_A = R_A$.

Set-Private-Key. On input x_A, R_A, and D_A. Compute $y_A = H_2(R_A)$ and $S_A = \frac{1}{x_A+y_A} D_A$. Return the (full) private key $SK_A = S_A$.

Sign. On input `params`, ID_A, S_A, and a message m, perform the following steps:

1. Choose a random $r \in \mathbb{Z}_q^*$.
2. Compute $U = g^r = e(P, P)^r$.
3. Set $h = H_3(m, U)$.
4. Compute $V = (r + h)S_A$.
5. Return $\sigma = (U, V)$ as the signature on the message m.

Verify. On input `params`, ID_A, R_A, m, and $\sigma = (U, V)$. Compute $Q_A = (s + q_A)P = P_{pub} + H_1(ID_A)P$, $y_A = H_2(R_A)$, and $h = H_3(m, U)$. Check if $e(V, R_A + y_A Q_A) = Ug^h$ holds. If the equation holds, it outputs 1, otherwise 0.

We can easily show that our CLS scheme satisfies completeness property as follows:

$$\begin{aligned} e(V, R_A + y_A Q_A) &= e((r+h)S_A, x_A(P_{pub} + q_A P) + y_A(P_{pub} + q_A P)) \\ &= e\Big((r+h)\frac{1}{(x_A+y_A)(s+q_A)}P, (x_A+y_A)(s+q_A)P\Big) \\ &= e((r+h)P, P) \\ &= e(P, P)^{r+h} = Ug^h. \end{aligned}$$

4.2 Security Analysis

To prove the security of the oCLS, we review the k-CAA (Collusion Attack Algorithm with k traitor) problem.

k-CAA [13] problem: The k-CAA problem is to compute $\frac{1}{s+t_0}P$ for some $t_0 \in \mathbb{Z}_q^*$ when given

$$P, sP, \quad t_1, t_2 ..., t_k \in \mathbb{Z}_q^*, \ \ \frac{1}{s+t_1}P, \ \frac{1}{s+t_2}P, ..., \frac{1}{s+t_k}P.$$

Theorem 3. *Our certificateless signature scheme oCLS is existentially unforgeable against a Type I forger in random oracle model under the k-CAA assumption.*

Proof. Suppose there exists a forger $\mathcal{F}_I$ which has advantage in attacking our CLS scheme oCLS. We want to build an algorithm $\mathcal{C}$ that uses $\mathcal{F}_I$ to solve the k-CAA problem. $\mathcal{C}$ receives a k-CAA instance $(P, sP, t_1, ..., t_k, \frac{1}{s+t_1}P, ..., \frac{1}{s+t_k}P)$ where $k \geq q_{H_1}$ (we suppose $\mathcal{F}_I$ makes at most q_{H_1} queries to H_1 oracle). Its goal is to compute $\frac{1}{s+t_0}P$ for some t_0. $\mathcal{C}$ runs $\mathcal{F}_I$ as a subroutine and simulates its attack environment. $\mathcal{C}$ sets $g = e(P, P)$ and $P_{pub} = sP$ where s is the master key, which is unknown to $\mathcal{C}$, and gives system parameters to $\mathcal{F}_I$. Without loss of generality, we assume that any extraction (ExtrPartSK, RequestPK, ExtrFullSK) and signature (SIGN) queries are preceded by H_1 query, and the

SIGN and ExtrFullSK queries are preceded by RequestPK query. To avoid collision and consistently respond to these queries, $\mathcal{C}$ maintains four lists L_{H_1}, L_{H_2}, L_{H_3}, $L_K = \{\langle ID, PK_{ID}, x_{ID}, c(= 0 \text{ or } 1)\rangle\}$ which are initially empty. $\mathcal{C}$ then simulates the oracle queries of $\mathcal{F}_I$ as follows:

– When $\mathcal{F}_I$ makes H_2, H_3, and ReplacePK queries, $\mathcal{C}$ performs as in the proof of Theorem1.
– H_1 query: $\mathcal{F}_I$ makes an H_1 query on ID_i where $1 \leq i \leq q_{H_1}$, $\mathcal{C}$ chooses $j \in [1, q_{H_1}]$ randomly. If $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ returns $q_{ID_i} = t_0$, otherwise $q_{ID_i} = t_i$. $\mathcal{C}$ then computes $Q_{ID_i} = sP + q_{ID_i}P$ and adds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ to L_{H_1}.
– ExtrPartSK(ID_i) query: When $\mathcal{F}_I$ makes this query on ID_i, if $ID_i \neq ID^*$, $\mathcal{C}$ returns $D_{ID_i} = \frac{1}{s+t_i}P$. Otherwise $\mathcal{C}$ outputs FAIL and aborts the simulation.
– RequestPK(ID_i) query: When $\mathcal{F}_I$ makes this query on ID_i, if the list L_K contains $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ returns PK_{ID_i}. Otherwise, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ in L_{H_1}, and picks a random $x_{ID_i} \in \mathbb{Z}_q^*$. $\mathcal{C}$ then returns $PK_{ID_i} = x_{ID_i}Q_{ID_i}$ and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}, 1\rangle$ to L_K.
– ExtrFullSK(ID_i) query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in and L_K, respectively. $\mathcal{C}$ performs as follows:
  • If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$.
  • If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
– SIGN(m, ID_i) query: When $\mathcal{F}_I$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in L_{H_1} and L_K, respectively. Then $\mathcal{C}$ performs as follows:
  • If $c = 1$, $\mathcal{C}$ picks two random $r_i, h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2}. If it does not exist, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. $\mathcal{C}$ computes $U_i = g^{-h_i}e((r_i + h_i)P, Q_{ID_i})$ and $V_i = (r_i + h_i)\frac{1}{s+q_{ID_i}}P$. $\mathcal{C}$ then returns (U_i, V_i) and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).
  • If $c = 0$, $\mathcal{C}$ gets additionally information x'_{ID_i} from $\mathcal{F}_I$. Using the $Q'_{ID_i} = x'_{ID_i}(sP + q_{ID_i}P)$, $\mathcal{C}$ then simulates as in the above case $(c = 1)$.

Eventually, $\mathcal{F}_I$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \neq ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ performs as in the proof of Theorem 1., and then he computes as follows:

$$(x_{ID_t} + y_{ID_t})\frac{V_t - V'_t}{h_t - h'_t} = SK_{ID_t} = \frac{1}{s + q_0}P.$$

Therefore, if a Type I forger who can break our scheme oCLS exists, then an attacker who solves the k-CAA problem exists. □

Theorem 4. *Our certificateless signature scheme* oCLS *is existentially unforgeable against a Type II forger in random oracle model under the mICDH assumption.*

Proof. This proof is same as the proof of Theorem 2. The different points are as follows:

– H_1 query: When $\mathcal{F}_{II}$ makes this query on ID_i, $\mathcal{C}$ picks a random $q_{ID_i} \in \mathbb{Z}_q^*$ and returns q_{ID_i}. $\mathcal{C}$ then computes $Q_{ID_i} = sP + q_{ID_i}P$ and adds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ to L_{H_1}.

– H_2 query: When $\mathcal{F}_{II}$ makes this query on PK_{ID_i}, if $PK_{ID_i} = saP + q_{ID_i}aP$, $\mathcal{C}$ sets $y_{ID_i} = b$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns y_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.

– H_3 query: When $\mathcal{F}_{II}$ makes this query, $\mathcal{C}$ performs as in the proof of Theorem 1.

– $\mathsf{RequestPK}(ID_i)$ query: Suppose $\mathcal{F}_{II}$ makes at most q_{PK} queries to public key request oracle. First, $\mathcal{C}$ chooses $j \in [1, q_{PK}]$ randomly. When $\mathcal{F}_{II}$ makes a $\mathsf{RequestPK}$ query on ID_i, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ in L_{H_1}. If $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ sets $PK_{ID_i} = saP + q_{ID_i}aP$ and returns PK_{ID_i}, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i} = \perp\rangle$ to L_K. Otherwise $\mathcal{C}$ picks a random x_{ID_i} and returns $PK_{ID_i} = x_{ID_i}(sP + q_{ID_i}P)$, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ to L_K.

– $\mathsf{ExtrFullSK}(ID_i)$ query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ performs as follows:
  - If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$.
  - If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.

– $\mathsf{SIGN}(m, ID_i)$ query: When $\mathcal{F}_{II}$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ picks two random r_i, $h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2} (if it does not exist, $\mathcal{C}$ makes a H_2 query on PK_{ID_i} itself). $\mathcal{C}$ computes $U_i = g^{-h_i} \cdot e(PK_{ID_i}, r_iP) \cdot e(y_{ID_i}(s + q_{ID_i})P, r_iP)$, $V_i = r_iP$ and returns (U_i, V_i), and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).

Eventually, $\mathcal{F}_{II}$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \neq ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle m_t, U_t, h_t\rangle$ in L_{H_3}. Then, as in the proof of Theorem 1, $\mathcal{C}$ gets another valid signature tuple $(ID_t, m_t, h'_t, \sigma_t = (U_t, V'_t))$ such that $h_t \neq h'_t$. Since $\mathcal{C}$ knows the master key s, after $\mathcal{C}$ finds $\langle ID_t, Q_{ID_t}, q_{ID_t}\rangle$ in L_{H_1}, he can compute as follows:

$$(s + q_{ID_t})\frac{V_t - V'_t}{h_t - h'_t} = \frac{1}{a+b}P.$$

Therefore, if a Type II forger who can break our scheme oCLS exists, then an attacker who solves the CDH problem exists. □

5 Performance Analysis

We now compare our CLS schemes with other previously known CLS schemes [10,12,20] in Table 1.

Table 1. Comparison of Certificateless Signature Schemes

Schemes	Sign			Verify		
	e	G$_1$	G$_2$	e	G$_1$	G$_2$
HSMZ05 [10]	2	2	0	5	1	0
LCS05 [12]	0	2	0	4	1	0
ZWXF06 [20]	0	3	0	4	0	0
eCLS	0	2	0	2	2	0
oCLS	0	1	1	1	1	1

(e: pairing operation, G$_1$: multiplication in $\mathbb{G}_1$, G$_2$: exponentiation in $\mathbb{G}_2$)

According to the result in [3,4], the pairing operation is several times more expensive than the scalar multiplication in $\mathbb{G}_1$. Hence reducing the number of pairing operations is critical. As we shown in Table 1, our CLS schemes are more efficient than other previous schemes. In particular, our verification procedure requires only one (or two) pairing operation because checking the validity of the public key is not done separately.

6 Extension

The ring signature guarantees the anonymity of the signer. In certificateless ring signature (CLRS) schemes, to guarantee the signer anonymity, a signature generated should not reveal any information about both a signer's identity and his public key. This is different from the previous (certificate or identity-based) ring signature schemes, because the ring signature schemes hide a public key or identity of the actual signer. Using the similar techniques as in [19,7], it is possible that our CLS schemes are expanded to CLRS schemes. Also, applying the method in [18] to our CLS scheme, we can easily modify our eCLS to a certificateless blind signature scheme.

7 Conclusion

The certificateless public key cryptography is receiving significant attention because it is a new paradigm that simplifies the public key cryptography. In this paper, we proposed two efficient CLS schemes which are provably secure in the random oracle model. Particularly, our signature verification requires only one pairing operation. In addition, our CLS scheme can be easily extended to certificateless ring and blind signature schemes.

References

1. S. Al-Riyami and K. Paterson, *Certificateless public key cryptography*, Asiacrypt 2003, LNCS 2894, pp. 452-473, Springer-Verlag, 2003.
2. D. Boneh and X. Boyen, *Short Signatures Without Random Oracles*, Eurocrypt 2004, LNCS 3027, pp. 56-73, 2004.
3. P. S. L. M. Barreto, S. Galbraith, C. O. hEigeartaigh, and M. Scott, *Efficient Pairing Computation on Supersingular Abelian Varieties*, Crypto 2002, LNCS 2442, pp. 354-368, Springer-Verlag, 2002.
4. P. S. L. M. Barreto, B. Lynn, and M. Scott, *Efficient implementation of pairing-based cryptosystems*, Journal of Cryptology, pp. 321-334, 2004.
5. D. Boneh, H. Shacham, and B. Lynn, *Short signatures from the Weil pairing*, Journal of Cryptology, Vol. 17, No. 4, pp. 297-319, 2004.
6. X. Cao, K. G. Paterson and W. Kou *An Attack on a Certificateless Signature Scheme*, http://eprint.iacr.org/2006/367.
7. Sherman S.M. Chow, S.M. Yiu, and Lucas C.K. Hui, *Efficient Identity Based Ring Signature*, ACNS 2005, LNCS 3531, pp. 499-512, Springer-Verlag, 2005.
8. P. Gutmann, *PKI: It's not dead, just resting*, IEEE Computer, 35(8), pp. 41-49, 2002.
9. M. C. Gorantla and A. Saxena, *An Efficient Certificateless Signature Scheme*, CIS 2005, LNAI 3802, pp. 110-116, Springer-Verlag, 2005.
10. X. Huang, W. Susilo, Y. Mu, and F. Zhang, *On the security of certificateless signature schemes from asiacrypt 2003*, CANS 2005, LNCS 3810, pp. 13-25, Springer-Verlag, 2005.
11. B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng, *Key Replacement Attack Against a Generic Construction of Certificateless Signature*, ACISP 2006, LNCS 4058, pp. 235-246, Springer-Verlag, 2006.
12. X. Li, K. Chen, and L. Sun, *Certificateless signature and proxy signature schemes from bilinear pairings*, Lithuanian Mathematical Journal, Vol. 45, No. 1, pp. 76-83, 2005.
13. S. Mitsunari, R. Sakai and M. Kasahara, *A new traitor tracing*, Proc. of IEICE Trans. Vol. E85-A, No.2, pp.481-484, 2002.
14. D. Pointcheval and J. Stern, *Security Proofs for Signature Schemes*, Eurocrypt 1996, LNCS 1070, pp. 387-398, Springer-Verlag, 1996.
15. A. Shamir, *Identity based cryptosystems and signature schemes*, Crypto 1984, LNCS, Vol.196, pp. 47-53, Springer-Verlag, 1984.
16. A. R. Sadeghi and M. Steiner, *Assumptions related to discrete logarithms: why subtleties make a real difference*, Eurocrypt 2001, LNCS 2045, pp. 243-260, Springer-Verlag, 2001.
17. D. H. Yum and P. J. Lee, *Generic Constructin of Certificateless Signature*, ACISP 2004, LNCS 3108, pp. 200-211, Springer-Verlag, 2004.

18. F. Zhang and K. Kim, *Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairings*, ACISP 2003, LNCS 2727, pp. 312-323, Springer-Verlag, 2003.
19. F. Zhang, R. Safavi-Naini and W. Susilo, *An Efficient Signature Scheme from Bilinear Pairings and Its Applications*, PKC 2004, LNCS 2947, pp. 277-290, Springer-Verlag, 2004.
20. Z. Zhang, D. Wong, J. Xu and D. Feng, *Certificateless Public-Key Signature: Security Model and Efficient Construction*, ACNS 2006, LNCS 3989, pp. 293-308, Springer-Verlag, 2006.

Security Mediated Certificateless Signatures

Wun-She Yap[1], Sherman S.M. Chow[2], Swee-Huay Heng[3], and Bok-Min Goi[1]

[1] Centre for Cryptography and Information Security, FOE
Multimedia University, 63100 Cyberjaya, Malaysia
{wsyap,bmgoi}@mmu.edu.my

[2] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu

[3] Centre for Cryptography and Information Security, FIST
Multimedia University, Jln Ayer Keroh Lama, 75450 Melaka, Malaysia
shheng@mmu.edu.my

Abstract. In PKC 2006, Chow, Boyd and González Neito introduced the notion of security mediated certificateless (SMC) cryptography. SMC cryptography equips certificateless cryptography with instantaneous revocation. They presented a formal security model with two constructions for SMC encryption. This paper studies SMC signatures. We first present a security analysis of a previous attempt by Ju *et al.* in constructing a SMC signature scheme. We then formalize the notion of SMC signatures and propose the first concrete provable scheme without bilinear pairing. Our scheme is existential unforgeable in the random oracle model based on the intractability of the discrete logarithm problem, has a short public key size, and achieves a trust level which is the same as that of a traditional public key signature.

Keywords: security mediated, certificateless, SMC cryptography.

1 Introduction

Efficient revocation of public key certificates has always been a critical issue in the public key infrastructure (PKI). Several methods such as certificate revocation list (CRL), online certificate status protocol (OCSP) [22] and Novomodo (a scalable and small-bandwidth certificate validation scheme) [21] had been proposed to solve the certificate management issue. However, the search for a satisfactory solution continues. In USENIX 2001, Boneh *et al.* [5] first introduced a method for obtaining instantaneous revocation in RSA-type cryptosystems (more details are given in [4]). Rather than revoking the user's certificate, this new approach revokes the user's ability to perform the cryptographic operations such as signing and decryption. They introduced a new entity, which is an online semi-trusted server called as a security mediator (SEM). To sign or decrypt a message, a client must first obtain a message-specific token from its SEM. Without this token, the client cannot accomplish the intended task. To revoke the user's ability to sign or decrypt, the SEM is instructed to stop issuing tokens for the future request of that user.

In 2003, Al-Riyami and Paterson [1] introduced an intermediate model between the traditional public key cryptography (TPKC) and the identity-based cryptography (IBC), known as certificateless public key cryptography (CLPKC). In CLPKC, both the partial private key and user secret key are necessary in signing or decryption. On the other hand, the identity (ID) and the user public key are both required in verification and encryption respectively. Without using certificates that are essential in IBC, CLPKC achieves implicit certification (through the partial private key) while does not suffer from the inherent key escrow problem in IBC (through the user secret key). Thus, CLPKC still maintains the advantages enjoyed in TPKC and IBC.

For the key revocation in CLPKC, a pragmatic way to deal with the revocation problem is to concatenate a validity period to the ID. However, this involves the need to re-issue the partial private key periodically for each validity period, thus burdens the trusted key generation center (KGC), and requires users to store different IDs that are concatenated with different validity dates. Besides, this method does not fit an environment when immediate revocation is required.

Recently, Chow, Boyd and González Nieto initiated the study of SMC cryptography from a formal point of view in [9]. SMC cryptography solves the instantaneous key revocation problem in CLPKC while maintaining the merits in CLPKC: implicit certification without key escrow. In short, SMC cryptography achieves a set of features that no previous paradigms can satisfy simultaneously, so it provided a new compromise between the various desirable features.

1.1 Related Work

Chow, Boyd and González Neito discussed various facets of SMC cryptography and formalized the notion of SMC encryption in [9]. We note that Ju *et al.* has also discussed the idea of SMC cryptography briefly in [17]. Based on the Libert and Quisquater revocation mechanism [20] and Al-Riyami and Paterson various certificateless schemes [1], Ju *et al.* introduced a signature scheme, an encryption scheme and a hierarchical variant. Generally, [17] did not provide the necessary details clearly compared with [9]. More details are given in the Appendix. Ju *et al.*'s [17] signature scheme was derived from the Al-Riyami and Paterson certificateless signature (CLS) scheme [1] (the key construction and the `Verify` algorithms are just identical), although the authors did not cite it clearly. Their scheme only achieves level 2.

There are a number of CLS scheme proposals after [1], which are based on either bilinear pairing [8,13,16,18,19,27] or identity-based signatures (IBS) [15,26]. Huang *et al.*'s scheme [16] is the revised version of Al-Riyami and Paterson's insecure scheme [1], and Cao *et al.*'s scheme [8] is revised from the insecure scheme of Gorantla-Saxena [13]. Liu *et al.*'s scheme [18] is the only one proposed without random oracles; however, it is also the least efficient.

1.2 Trust Levels

Trust levels referred to the three different levels of trust placed on the trusted third party as defined by Girault [12], the higher level the more desirable.

– *Level 1*: The authority knows (or can easily compute) the private keys and is capable of impersonating any user without being detected.
– *Level 2*: The authority does not know the private keys, but it can impersonate any user by generating false certificates without being detected.
– *Level 3*: The authority cannot compute the private keys and if it generates false certificates for users, such generation can be detected.

Table 1 summarizes the comparison of various security mediated signature.

Table 1. Properties of Security Mediated Signatures

Schemes in Different Paradigms	Implicit Certification	Escrow Freeness	Trust Level
Security mediated (traditional) signature [4,5,20]	✘	✘	1
Security mediated identity-based signature [11]	✓	✘	1
Security mediated certificateless signature	✓	✓	3

While key escrow is inherent in IBC, the same problem is not necessary inherent in all security mediated traditional signatures (with trust level 1). The above table considers the security mediated traditional signatures in [4,5,20], in which the certification authority is the one who is responsible in generating the user private key, they could only achieve trust level 1.

1.3 Our Contributions

We initiate the formal study of revocation in certificateless signatures paradigm. Our contributions are three-fold.

Attack. We show that the previous attempt by Ju *et al.* in constructing SMC signature scheme is flawed. More precisely, their scheme does not support revocation at all since the user can get the SEM private key after interacting with the SEM, i.e. after receiving a partial signature. Consequently, the user can sign any subsequent messages thereafter without the assistance of the SEM. In short, the complication involved the SEM is redundant and useless.

Recently, the Al-Riyami and Paterson CLS scheme was shown to be insecure by a realistic key replacement attack by Huang *et al.* in [16]. We show that Ju *et al.*'s SMC signature scheme is vulnerable to a similar attack as well.

Security Model. We formalize the security model of SMC signature. We highlight the differences between our model and existing certificateless signature model. We also discuss the subtleties in the definition of "public key" arose from the probabilistic key generation, which introduces a public component selected by the KGC instead of the user.

Efficient Construction. We then present the first concrete provable secure SMC signature scheme without bilinear pairing. Our scheme is existential unforgeable in the random oracle model based on the intractability of the discrete logarithm problem and achieves a trust level which is the same as that of a traditional digital signature scheme. It is worth noting that our proposed scheme can be transformed into an efficient CLS scheme without pairing.

Organization. Section 2 provides the framework and the security model of SMC signatures. Section 3 presents both the insider and outsider attacks on Ju *et al.*'s SMC signature. Section 4 shows the proposed construction of SMC signature together with its security and performance analysis.

2 Security Mediated Certificateless Signatures

Here we present the framework and the security model of SMC signature scheme. We adopt the compact but versatile model of [15] improved from [1,16,26].

2.1 Framework

Definition 1. *A security mediated certificateless signature scheme consists of five tuples of polynomial time algorithms as follows:*

1. `Setup` *is a probabilistic algorithm that takes as input a security parameter in the form of* 1^k *and returns a master key* s *and a parameter list params.*
2. `KeyGen` *is a probabilistic algorithm that takes as input a parameter list params. It picks a secret value at random. The public key is computed based on the selected value. It returns a pair of matching public and private keys* (P_{ID}, x_{ID}).
3. `Register` *is a probabilistic algorithm that takes as input a parameter list params, the master key* s*, an user identity* ID *and a public key* P_{ID}*. It returns the SEM private signing key* D_{ID}.
4. `Sign` *is an interactive probabilistic protocol between the user and the SEM. Their common inputs include a parameter list params, a message* m*, and an user identity* ID*. The SEM has an additional input of* D_{ID} *to run the sub-algorithm* `SEM-Sign`*; while the user has an additional input of* x_{ID} *to run the sub-algorithm* `User-Sign`*. The protocol finishes with either a signature* σ*, or* $\perp$ *when the SEM refuses to give a valid partial signature, for example in the case where the user's signing capability has been revoked.*
5. `Verify` *is a deterministic algorithm that takes as input a parameter lists params, a message* m*, an user identity* ID*, an user public key* P_{ID} *and a signature* σ*. It returns* $true$ *or* $false$.

More precisely, the scheme flows as follows. First, the KGC will take as input a security parameter 1^k to generate the *params* and the master key s by running the `Setup` algorithm. Then, the user who holds identity ID runs the `KeyGen` algorithm to generate the public key P_{ID} and the secret value x_{ID}. The public key P_{ID} is registered with the KGC via authentication and the `Register` algorithm, during which the KGC will use the master key s to generate the SEM private signing key D_{ID}. This key needs to be transmitted to the SEM authentically and confidentially through a secure channel. The secrets held by the SEM and the user are different. During the running of the `Sign` protocol, the SEM and the user uses their corresponding secret to sign the message cooperatively. Finally, a recipient needs both the public key and ID to verify the validity of the signature by using the `Verify` algorithm.

We keep the following new features of the model introduced in [15]:

1. The SEM private key generation by the KGC is probabilistic.
2. The SEM private key generated and the user private key can be "mixed" together in some randomized way each time during signature generation.
3. A single probabilistic algorithm is used for creating the public/private key pair.

The first point worths more discussion. In most of the existing concrete CLS schemes (i.e. excluding generic constructions using other signature schemes like IBS as a building block), deterministic algorithm is used such that there is only one partial private key (the SEM private key in our context) corresponding to each identity. If a probabilistic algorithm is used, the randomness introduced may give rise to a new "public component" that should be included in the signature. This gives ambiguity in what is meant by "user public key". No discussion has been made on this subtle point so far. We defer our discussion on this issue to Section 4.4, with the hope that the abstract concept can be better understood based on a concrete scheme.

2.2 Security Model

We consider the security against existential forgery on adaptive chosen message and identity attacks. The security model of SMC signature is a mixture of the models in [9,15]. There are two types of adversary with different capabilities. The Type I adversary $\mathcal{A}_{\mathcal{I}}$ acts as a dishonest user. The Type II adversary $\mathcal{A}_{\mathcal{II}}$ acts as a malicious KGC (we do not consider a rogue SEM explicitly since it is strictly weaker than the Type II adversary).

A SMC signature scheme is secure against the existential forgery on adaptive chosen message and identity attacks (EUF-CMIA) against adversary $\mathcal{A} = \langle \mathcal{A}_{\mathcal{I}}, \mathcal{A}_{\mathcal{II}} \rangle$ if no polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage against a challenger $\mathcal{C}$ in the following game:

1. `Setup`: $\mathcal{C}$ takes as input 1^k, runs the `Setup` algorithm, and gives $\mathcal{A}$ the resulting *params*. The master key is given to $\mathcal{A}$ if it is a Type II adversary.
2. `Attack`: $\mathcal{A}$ issues a sequence of requests, each request being either a query of `Create`, `Replace`, `SEM-Extract`, `User-Extract`, `SEM-Sign`, `User-Sign` or `Complete-Sign` for a particular entity.
   `Create` queries create users by either registering the user public key when playing against $\mathcal{A}_I$, or executing the `Key-Gen` algorithm for $\mathcal{A}_{II}$. `Replace` queries let $\mathcal{A}_I$ to change user public key at its wish. Two `Extract` queries return the SEM and user private key respectively. `Sign` queries are to be

explained shortly. These queries may be asked adaptively, subjected to the rules on adversary behaviors to be defined below.

3. `Forgery`: $\mathcal{A}$ outputs a signature σ on message m signed by user ID with public key P_{ID}. The only restriction is that (m, ID) does not appear in the set of previous `Sign` queries. $\mathcal{A}$ wins the game if $\texttt{Verify}(params, \sigma, m, ID, P_{ID})$ is *true*. The advantage of $\mathcal{A}$ is defined as the probability that it wins.

Differences from the Existing Certificateless Signature Models: We highlight the differences from the existing models on the following types of query.

1. `SEM-Sign`: On input a message m and an identity ID, the adversary is returned with the partial signing result by using D_{ID}.
2. `User-Sign`: On input a message m and a public key P_{ID}, the adversary is returned with the partial signing result by using x_{ID}, *even* if the public key P_{ID} is *previously replaced* by the (Type I) adversary.
3. `Complete-Sign`: On input a message m and a public key P_{ID}, the adversary is returned with the complete signing result by using x_{ID} and D_{ID}, *even* if the user public key P_{ID} is *previously replaced* by the (Type I) adversary.

The first two queries capture the adversary's capabilities to ask for partial signing results, which are not considered in the traditional certificateless setting. The last one appears in a weaker form in the existing certificateless signature model [15,26], such that signing query for a replaced public key requires the adversary to submit the corresponding private key. If an invalid one is submitted (or no key is submitted at all), an invalid signature will be given. This restriction makes the model weaker than its counterpart in encryption [1,9], where the decryption oracle gives valid result even if the public key has been replaced.

It may seem unrealistic to entertain any signing query of the replaced public key from the first glance. We try to give some intuition here. Suppose the adversary has an access of a cryptographic device (e.g. giving standard signatures) with some public/private key pair. It is not impossible that the combination of this particular public key with a certain identity may enable the adversary to forge. In this case, the signing query models some useful knowledge about the replaced public key that the adversary may obtain *outside* our system. Our security formulation guarantees unforgeability in this case.

Now we spell out some restrictions placed on the adversaries.

SMC Signatures Type I Adversary: Adversary $\mathcal{A}_{\mathcal{I}}$ does not have access to the master key. On the other hand, $\mathcal{A}_{\mathcal{I}}$ may request and replace the public keys, extract the SEM private key and the user private key and make the sign queries. Here are several natural restrictions on such a Type I adversary:

1. $\mathcal{A}_{\mathcal{I}}$ cannot extract the SEM private key of the challenge identity ID^*.
2. $\mathcal{A}_{\mathcal{I}}$ has not issued any `SEM-Sign` query on the forged message m for the challenged identity ID^*.
3. $\mathcal{A}_{\mathcal{I}}$ cannot make a `Complete-Sign` query on the forged message m for the challenged identity ID^*.

SMC Signatures Type II Adversary: Adversary $\mathcal{A}_{\mathcal{II}}$ does have access to the master key, but cannot replace the public keys of entities. Adversary $\mathcal{A}_{\mathcal{II}}$ can compute the SEM private key itself, request the public keys, extract users' private key and make the sign queries, all for the identities of its choice. The restrictions on this type of adversary are:

1. $\mathcal{A}_{\mathcal{II}}$ cannot replace the public keys of the challenge identity ID^*.
2. $\mathcal{A}_{\mathcal{II}}$ cannot extract the user private key for ID^* at any point.
3. $\mathcal{A}_{\mathcal{II}}$ cannot make an `User-Sign` query on the forged message m for the challenged identity ID^*.
4. $\mathcal{A}_{\mathcal{II}}$ cannot make a `Complete-Sign` query on the forged message m for the challenged identity ID^*.

Definition 2. *A SMC signature scheme is secure against EUF-CMIA if there is no efficient adversary in the above game with a non-negligible advantage in the security parameter k for both types of adversary.*

3 Critical Review on Ju *et al.*'s Scheme

First, we review the construction of Ju *et al.*'s scheme.

– `Setup`: Given a security parameter k, the KGC performs the steps below:
  1. Run the Bilinear Diffie-Hellman parameter generator $\mathcal{IG}$ [6] with input k in order to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of some prime order q and a *bilinear map* $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
  2. The system parameters are *params*= $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}(\cdot,\cdot), P, P_0, H_1(\cdot), H_2(\cdot)\rangle$ where P is an arbitrary generator in $\mathbb{G}_1$, $P_0 = sP$ where s is picked uniformly at random from Z_q^*, H_1 and H_2 are cryptographic hash functions where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \times \mathbb{G}_2 \rightarrow Z_q^*$.
  3. The master key is s.

– `Key Generation`: Given an user with identity $ID_A \in \{0,1\}^*$,
  1. The KGC computes $Q_A = H_1(ID_A) \in \mathbb{G}_1$ and $D_{ID_A} = sQ_A$.
  2. The KGC randomly chooses $D^U_{ID_A} \in \mathbb{G}_1$.
  3. The KGC sends the SEM private key $D^U_{ID_A}$ to the user A and $D^S_{ID_A} = D_{ID_A} - D^U_{ID_A}$ to the SEM, over a confidential and authentic channel.
  4. The user A selects $x_A \in Z_q^*$ as his private key and constructs his public key as $\langle X_A, Y_A\rangle = \langle x_A P, x_A P_0\rangle$.

– `Sign`: Now user A with identity ID_A wants to get a signature on $m \in \{0,1\}^*$.
  1. A chooses a random element $a \in Z_q^*$, computes the following values: $r = \hat{e}(aP, P) \in \mathbb{G}_2$, $\upsilon = H_2(m, r) \in Z_q^*$, and $T^U = \upsilon D^U_{ID_A}$.
  2. A sends υ to the SEM, who checks if A's public key has been revoked.
  3. If not, the SEM computes $T^S = \upsilon D^S_{ID_A}$ and sends it to A.
  4. Upon receiving T^S, A computes $T = x_A(T^S + T^U) + aP$.
  5. The final signature is $\langle T, \upsilon\rangle$.

– `Verify`: When receiving $\langle T, \upsilon\rangle$ on message $m \in \{0,1\}^*$ for identity ID_A and public key $\langle X_A, Y_A\rangle$, the verifier performs the following steps:
  1. If $\hat{e}(X_A, P_0) \neq \hat{e}(Y_A, P)$, return $\perp$ and abort.
  2. Compute $r' = \hat{e}(T, P) \cdot \hat{e}(Q_A, -Y_A)^{\upsilon}$.
  3. Accept the signature if and only if $\upsilon = H_2(m, r')$ holds.

3.1 Insiders Attack Against Revocation

The whole point of introducing a SEM is for the instant revocation of the user's ability to sign. The user cannot issue signature without the cooperation of the SEM since the SEM holds a SEM private key of each user, which is essential to compute a signature. In other words, the user must seek help from the SEM every time he/she wants to issue a new signature. It is thus natural to assume that a malicious user may interact with the SEM in anyway to gain knowledge about the partial private key and later produce a signature autonomously.

Now we show that it is easy for a malicious user to get his/her SEM private key by only one interaction with the SEM. After getting hold of the partial private key, the user can sign any subsequent messages thereafter at will without the assistance of the SEM anymore. This flaw is very disastrous since the scheme is no longer having the merit of security mediated cryptography, i.e. it does not provide the instantaneous key revocation property as claimed.

The flaw is that in their `Sign` algorithm, the SEM computes $T^S = \upsilon D^S_{ID_A}$ and sends it to the user A. The user A can easily compute the value of $D^S_{ID_A}$ by using the equation $D^S_{ID_A} = \upsilon^{-1} \cdot T^S$ as υ is a value to be included in the final signature anyway. Then, the user A can generate signature on any message with the complete knowledge of $D^S_{ID_A}$, $D^U_{ID_A}$ and x_A.

Here, we point out some considerations in proposing a provable secure SMC signature scheme. Firstly, we should ensure that the adversary does not gain any advantage from the above queries. The SEM private key and the user private key must be protected when issuing a partial signature. Leaking either one part of the private keys will result in an insecure scheme.

3.2 Key Replacement Attack by Any Malicious Outsider

Ju *et al.* scheme [17] follows the approach used in the Al-Riyami and Paterson CLS scheme [1] while the latter was derived from the Hess IBS scheme [14]. The Al-Riyami and Paterson scheme was proven insecure in the defined model of [16]. The insecurity lies in that [1] adopted a similar approach as [14] without considering the adversary's ability in CLPKC. Caution must be taken considering the ability of the adversary in replacing any public key, as no certificate is needed in authenticating the user public key.

Now we show that Ju *et al.*'s scheme [17] is vulnerable to the public key replacement attack by Type I adversary, similar to the technique in [16]. As defined in [1], Type I adversary represents a dishonest user who can replace user public key at will (as there is no certificate to authenticate the public key).

Our break is a strong one, which is universal forgery against no message attack, i.e. no signing oracle is required in the Type I adversarial model (the attack does not need any help from the SEM and the user in signing any message for any identity ID) and the forger can sign any message. Details are given below:

`Sign`: To sign a message m with identity ID_i, where $Q_i = H_1(ID_i)$:

1. Select a random $T \in \mathbb{G}_1$, compute $r = \hat{e}(T, P)\hat{e}(Q_i, -P_0)$ and $\upsilon = H_2(m, r)$.
2. Set $x_i = \upsilon^{-1} \in Z^*_q$, compute $X'_i = x_i P$ and $Y'_i = x_i P_0$.

3. Replace the user public key with $\langle X_i', Y_i' \rangle$, a clearly valid one.
4. Return $\langle T, v \rangle$ as a signature of m.

The forged signature is valid since

$$r' = \hat{e}(T, P) \cdot \hat{e}(Q_i, -Y_i')^v = \hat{e}(T, P) \cdot \hat{e}(Q_i, -x_i P_0)^v = \hat{e}(T, P) \cdot \hat{e}(Q_i, -P_0) = r.$$

Although it may be possible to fix the scheme by the countermeasures in [16], it still fails to provide instantaneous revocation capability. Instead of fixing, we propose a much more efficient provably secure scheme in the next section.

4 Our Proposed Construction

This section presents our proposed scheme derived from the Schnorr signature scheme [24]. The similar partial private key construction was used in [2]. The merit of this approach is that no pairing computation is needed at all.

- `Setup`. Given the security parameter k, the KGC performs the following.
  1. Generate two primes p and q such that $q|p-1$.
  2. Pick a generator g of Z_p^*.
  3. Pick $s \in Z_q^*$ uniformly at random and compute $Y = g^s$.
  4. Choose hash functions $H_0 : \{0,1\}^* \to \{0,1\}^\ell$, $H_1 : \{0,1\}^* \to Z_q^*$ and $H_2 : \{0,1\}^* \to Z_q^*$.
  5. Return *params* $= (p, q, g, Y, H_0, H_1, H_2)$ and master key is s.
- `KeyGen`: The user performs the following.
  1. Randomly select $x_{ID} \in Z_q^*$ as the user private key.
  2. Compute $P_{ID} = g^{x_{ID}} \in Z_q^*$ as the user public key.
- `Register`: Now user ID wants to register a public key P_{ID}:
  1. The KGC authenticates and registers (ID, P_{ID}), randomly picks $w \in Z_q^*$.
  2. The KGC computes $W = g^w$ and $d = w + sH_1(ID||W)$.
  3. The KGC sends the SEM private key $D_{ID} = \langle W, d \rangle$ and the (ID, P_{ID}) pair to the SEM over a confidential and authentic channel.
- `Sign`: Suppose the user ID wants to get the signature of message m, the SEM checks whether ID is revoked; if not, the interaction between the signer and the SEM is as follows (I denotes the first part and II denotes the second):
  - `SEM-Sign` (I): randomly chooses $r_S \in Z_q^*$, computes $R_S = g^{r_S}$, sends $c = H_0(R_S)$ to the user.
  - `User-Sign` (I): randomly chooses $r_U \in Z_q^*$, sends $R_U = g^{r_U}$ to the SEM.
  - `SEM-Sign` (II): computes

$$R = R_S \cdot R_U, h_S = H_2(ID||W||0||P_{ID}||R||m), t = r_S + d \cdot h_S;$$

  and sends back $\langle R_S, t \rangle$ to the user.

  Note that the partial signing result generated by the SEM does not need to transmit to the user through a secure channel since any party other than the SEM and the user does not gain any advantage from it.

  Depending on the scenarios, the user may want to verify the correctness of the partial signature given by the SEM by checking whether the equality $R_S = g^t(W \cdot Y^{H_1(ID||W)})^{-h_S}$ holds.

- `User-Sign` (II): checks if $c = H_0(R_s)$; if they are equal, computes

$$R = R_S \cdot R_U, h_U = H_2(ID||P_{ID}||1||W||R||m), v = r_U + x_{ID} \cdot h_U + t.$$

The final signature is $\sigma = \langle R, V, (ID||P_{ID}||W)\rangle$.

– `Verify`: Given σ, accepts if and only if the following equality holds:

$$R = g^V {P_{ID}}^{-H_2(ID||P_{ID}||1||W||R||m)}(W \cdot Y^{H_1(ID||W)})^{-H_2(ID||W||0||P_{ID}||R||m)}.$$

The correctness of the proposed scheme can be easily verified as follows.

$$\begin{aligned} R' &= g^V {P_{ID}}^{-h_U}(W \cdot Y^{H_1(ID||W)})^{(-h_S)} \\ &= g^{r_S + r_U + d \cdot h_S + x_{ID} \cdot h_U} g^{-x_{ID} \cdot h_U}(g^w \cdot g^{sH_1(ID||W)})^{(-h_S)} \\ &= g^{r_S + r_U + d \cdot h_S} g^{(w + sH_1(ID||W))(-h_S)} \\ &= g^{r_S + r_U + d \cdot h_S} g^{(d)(-h_S)} \\ &= g^{r_S + r_U} = R \end{aligned}$$

4.1 Generic Construction

One of the major characteristics of SMC signature is that the SEM private key generated by the KGC and the user private key are required in different partial signing algorithm; which is different with

1. the traditional CLS model [1,16,26], where the partial private key generated by the KGC and the user private key are combined into a single private key,
2. the new CLS model [15], where the partial private key generated by the KGC and the user private key are both inputs of the same signing algorithm.

The signing algorithm of the generic CLS construction in [15] can actually be partitioned into two parts – the first part takes the partial private key (but not the user private key) for partial signature generation, and the second part which takes the user private key and the partial signature to give the final one.

Based on these observations, it is tempting to use the above partition to give a generic construction of SMC signature scheme. Nevertheless, similar to the weakness of the generic construction of SMC encryption in [9] (where decryption oracle may not work if the public key has been replaced), such approach cannot give us signing oracles which still work after the public key has been replaced.

4.2 Extension from CLS

Another possible approach to construct SMC signature scheme is to extend from existing concrete CLS scheme. We note that Zhang *et al.*'s scheme [27] can be partitioned in the way as described above, and the resulting scheme supports partial signing queries even if the public key has been replaced. However, such level of security is achieved at the expense of high computational complexity.

4.3 Efficiency Analysis

We denote "Exp" the exponentiation in Z_p^*, "MtP" the `MapToPoint` encoding function in [6] and $\hat{e}(\cdot,\cdot)$ the pairing operation. The "n" letters appear in the "MtP" columns denote n multiplications in Z_p^* are required in encoding an identity or a message, where n denotes the bit-length of an identity or a message to be signed. Generally speaking, the operations are listed according to their efficiency in descending order.

Regarding our scheme's performance, the SEM private key generation requires only 1 Exp. To verify, 4 Exp's are needed. For signing, each of the user and the SEM compute 1 Exp. Table 2 compares the performance of our proposal with existing CLS schemes. Note that we include the total computational requirement of both the SEM and user in signing, and ignore the `MapToPoint` operation to derive the public key during the signing and the verification. These considerations are unfavorable to our proposal.

It is clear that our proposal outperforms all existing ones.

Table 2. Efficiency Analysis of Certificateless Signature Schemes

	Key Generation			Signing			Verification		
	Exp	MtP	$\hat{e}(\cdot,\cdot)$	Exp	MtP	$\hat{e}(\cdot,\cdot)$	Exp	MtP	$\hat{e}(\cdot,\cdot)$
Cao *et al.* [8]	1	1	0	3	0	1	1	0	4
Huang *et al.* [16]	1	1	0	3	0	1	1	0	4
Li *et al.* [19]	1	1	0	2	0	0	1	0	4
Zhang *et al.*[27]	1	1	0	3	2	0	0	2	4
Liu *et al.*[18]	2	n	0	4	n	0	0	n	6
Our Proposed Scheme	1	0	0	2	0	0	4	0	0

Generic constructions of certificateless signature are proposed in [26] and [15]. ([26] is later shown to be insecure by [15]). Comparing our proposed scheme with the construction in [15], our scheme produces signatures of shorter length, as the signature in [15] is a concatenation of an IBS and a traditional signature.

4.4 Subtleties of the Public Component Selected by the KGC

In our scheme, there is a public component $W = g^w$ where w is randomly chosen by the KGC. W is public since it is in the final signature. No existing certificateless signature schemes have such component. Subtleties appear such that whether this element can be considered as part of the public key.

This component W appears is in the generation and extraction of the SEM key. It also appears in the `SEM-Sign` query to get signatures for some purported public key as well as in the forgery returned by the adversary.

For the SEM key generation, W is generated together with the SEM private key $d = w + sH_1(ID||W)$, which both are sent to the KGC. The KGC only knows the correct value of d corresponding to the value of W given by the KGC. Since the KGC is responsible for checking whether the user's signing right has been revoked anyway, it is actually natural to assume the KGC will *not* sign for whatever purported W. In our security analysis, we also impose such restriction.

Similarly, no extraction of the SEM key according to an adversarially chosen W, since (W, d) is considered as a SEM key *as a whole*. However, considering the fact that the verifier does not hold a legitimate copy of W (that the SEM is holding), i.e. any verifier can only use the W included in the signature during the verification, then forgery with W *replaced* is considered as *valid*.

4.5 The Importance of the Key Binding

In order to achieve trust level 3, we can use the binding technique that ensures only one public key, for which the user knows the corresponding private key can be created. This technique was first employed in [1,26] in order to prevent the KGC from issuing two valid partial private keys (in certificateless context, or the SEM private keys in our terms) for a single user.

With the binding technique in place, the existence of two working public keys for an ID can only result from the existence of the SEM private keys binding that identity to two different public keys; only the KGC could have created these two SEM private keys since only the KGC has the master key.

This technique is useful when higher security level is required, especially in corporate and military environment. It is acceptable that only one permanent public key is available for one identity. When the user private key found compromised or the user is removed from his/her duty, the ID will be revoked permanently. Nevertheless, we note a drawback of the binding technique, which makes the user who holds ID can no longer use a new public key P'_{ID}.

If such permanent revocation is undesirable, when either the SEM private key or the user private key is compromised, some "out-of-band" non-cryptographic signature should be given by the party concerned, so no one can claim later that something malicious is happening even there exist two "working" public keys.

4.6 Security

Before the security proof, we review the definition of discrete logarithm problem.

Definition 3. ***Discrete Logarithm Problem (DLP)****. The DLP is the problem of finding a given (p, q, g, g^a) with uniformly random choices of $a \in Z_q^*$ and $g \in Z_p^*$. The DL assumption states that there is no polynomial time algorithm with a non-negligible advantage in solving the DLP.*

The security proofs below borrow some proof ideas from [7]. The first one is to ensure certain random oracle responses to be the same in two executions of the adversary, irrespective of when the corresponding queries are made by the adversary. This is done by letting one oracle query to trigger the determination of a number of related oracle responses. Another one is the use of commitment to help the correct programming of the random oracle. This trick has been adopted elsewhere, for example, in Nicolosi *et al.*'s proactive two-party signatures [23].

Theorem 1. *Our SMC signature scheme is existential unforgeable against the Type I adversary in the random oracle model assuming the DLP in $\mathbb{G}$ is hard.*

Proof. Let $\mathcal{A}_{\mathcal{I}}$ be a forger that breaks the proposed signature scheme under adaptive chosen message and identity attack. We show that how $\mathcal{B}$ can use $\mathcal{A}_{\mathcal{I}}$ to solve the DLP instance (p, q, g, g^a).

$\mathcal{B}$ sets $Y = g^a$ and system parameters as (p, q, g, Y). $\mathcal{B}$ then gives the system parameters to $\mathcal{A}_{\mathcal{I}}$. It also maintains a list of tuples $(ID_i, W_i, e_i, d_i, P_i, x_i)$ which is denoted as H_1^{list}. Next, $\mathcal{B}$ randomly selects an index I such that $1 \le I \le q_{H_1}$, where q_{H_1} denotes the maximum number of queries to the random oracle H_1. $\mathcal{B}$ also picks $e_I \in Z_q^*$ at random and sets $W_I = g^a Y^{-e_I}$.

Adversary $\mathcal{B}$ interacts with $\mathcal{A}_{\mathcal{I}}$ in the `Attack` phase of the game as follows:

H_0 `Queries`: When $\mathcal{A}_{\mathcal{I}}$ queries H_0, $\mathcal{B}$ checks the corresponding H_0^{list} and outputs c_i if such query has already been made. Otherwise, $\mathcal{B}$ picks $c_i \in \{0,1\}^\ell$ at random, updates the H_0^{list} and outputs c_i as answer.

H_1 `Queries`: When $\mathcal{A}_{\mathcal{I}}$ queries H_1 on input $(ID_i||W_i)$, $\mathcal{B}$ checks the corresponding H_1^{list} and outputs e_i if such value is defined. Otherwise, $\mathcal{B}$ picks $e_i \in Z_q^*$ at random and outputs e_i as answer. H_1^{list} is also updated, such that $(ID_i, W_i, e_i, \perp, \tilde{\Delta}, \tilde{\Delta})$ is stored, where $\tilde{\Delta}$ means the old value (if exists) is kept.

H_2 `Queries`: When $\mathcal{A}_{\mathcal{I}}$ issues a query on these hash values, $\mathcal{B}$ parses the input as $(ID||\mu||b||\nu||R_i||m_i)$. If the query cannot be parsed as this form, simply returns a random value $h_i \in Z_q^*$. If $b = 0$, $\mathcal{B}$ sets $W_i = \mu$ and $P_i = \nu$. If $b = 1$, $\mathcal{B}$ sets $W_i = \nu$ and $P_i = \mu$. After that, $\mathcal{B}$ checks whether the value $H_1(ID||W_i)$ has been defined. If no, $\mathcal{B}$ runs the simulation of H_1 as usual to define it.

$\mathcal{B}$ then checks the corresponding H_2^{list}. If an entry for the query is found, the same answer will be given to $\mathcal{A}$. Otherwise, $\mathcal{B}$ randomly selects h_i and h_i' from Z_q^*, assigns $h_i = H_2(ID||\mu||0||\nu||R_i||m_i)$ and $h_i' = H_2(ID||\nu||1||\mu||R_i||m_i)$. By symmetry, $H_2(ID||\nu||1||\mu||R_i||m_i)$ must have not been defined beforehand if $H_2(ID||\mu||0||\nu||R_i||m_i)$ has not. Finally, all queries and the corresponding answer will be stored in the H_2^{list}.

`SEM-Extract`: Suppose the query is on ID_i.

1. If $i \neq I$, then $\mathcal{B}$ picks $d_i, e_i \in Z_q^*$ at random and computes $W_i = g^{d_i} Y^{-e_i}$. $\mathcal{B}$ updates the corresponding record in the H_1^{list} to $(ID_i, W_i, e_i, d_i, \tilde{\Delta}, \tilde{\Delta})$, where $\tilde{\Delta}$ means the old value is kept. (W_i, d_i) is returned as the answer.
2. Else if $i = I$, $\mathcal{B}$ aborts.

The above simulation is faithful since $W_i \cdot Y^{H_1(ID_i||W_i)} = g^{d_i} Y^{-e_i} Y^{e_i} = g^{d_i}$.

`User-Extract`: Suppose the query is on ID_i. Assume the public key for ID_i has not been replaced (no such query is allowed otherwise), $\mathcal{B}$ responds as follow.

1. $i \neq I$: If the public key does not exist, $\mathcal{B}$ picks $x_i \in Z_q^*$ at random and updates the H_1^{list} (i.e. stores x_i and g^{x_i}). Otherwise, returns the x_i stored.
2. $i = I$: $\mathcal{B}$ aborts.

`Replace`: Suppose the public key for ID_i is replaced with value P'_i, $\mathcal{B}$ just updates H_1^{list} accordingly (i.e. stores $(P'_i, \perp)$ for user public/private key).

`SEM-Sign`: Note that at any time during the simulation, equipped with those SEM private keys for any $ID_i \neq ID_I$, $\mathcal{A}_\mathcal{I}$ is able to generate partial signatures on any message if the public key had not been replaced. We assume that the $\mathcal{B}$ would not help the user to sign for any purported W value, instead, SEM uses the stored one. For $ID_i = ID_I$, assume that $\mathcal{A}_\mathcal{I}$ issues a query (m_i, P_I) where m_i denotes a message and P_I denotes a current public key chosen by $\mathcal{A}_\mathcal{I}$ that is associated with ID_I. Given $W_I = g^a Y^{-e_I}$ where $e_I = H_1(ID_I||W_I)$, $\mathcal{B}$ creates a partial signature as follows:

1. Pick $t_i, h_i \in Z_q^*$ at random and set $R_S = g^{t_i} g^{a(-h_i)}$.
2. Execute H_0 oracle simulation, get $c_i = H_0(R_s)$ and send it to $\mathcal{A}$.
3. After getting R_U from $\mathcal{A}$, set $h_i = H_2(ID_I||W_I||0||P_I||R_S \cdot R_U||m)$.
4. Send $\langle R_S, t_i \rangle$ to $\mathcal{A}$.

Embedding h_i as the response of H_2 is not possible if $\mathcal{A}_\mathcal{I}$ has queried the H_2 value of $(ID_I||W_I||0||P_I||R_S \cdot R_U||m)$ beforehand. We consider the case that $H_0(R_S)$ has previously queried and the case that it was not. In the first case, $\mathcal{A}_\mathcal{I}$ probably knows R_S and may have deliberately queried such value. However, since t_i is chosen randomly by $\mathcal{B}$ independent of $\mathcal{A}_\mathcal{I}$'s view, the probability that $\mathcal{A}_\mathcal{I}$ made such H_0 query is at most $(q_H + q_S)/2^k$. In the second case, the view of $\mathcal{A}_\mathcal{I}$ is completely independent of R_S. The probability that $R_S \cdot R_U$ appeared (by chance) in a previous H_2 query is against at most $(q_H + q_S)/2^k$.

`User-Sign`: Note that at any time during the simulation, if equipped with those user private keys for any ID, $\mathcal{A}_\mathcal{I}$ is able to generate partial signatures on any message. For replaced public key P_I, the simulation is as follows.

1. An ℓ-bit commitment c_I is obtained from the adversary, which models the first message that SEM should be sent to initiate the `User-Sign` process.
2. Pick $\upsilon_i, \xi_i \in Z_q^*$ at random and set $R_U = g^{\upsilon_i} {P_I}^{(-\xi_i)}$.
3. Search for H_0^{list} to find R'_S such that $H_0(R'_S) = c_I$.
4. If found, set $\xi_i = H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$.
5. After obtained $\langle R_S, t_i \rangle$, check if $H_0(R_S) = c_I$, stop `User-Sign`.
6. If R'_S is previously found, but $R'_S \neq R_S$, declare failure and abort.
7. If R'_S was not found, and $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m) \neq \xi_i$, declare failure and abort since $H_0(R_S) = c_I$ but $\mathcal{B}$ cannot program H_2 accordingly.
8. If $\mathcal{B}$ is lucky enough to reach this step, send $\langle R_U, \upsilon_i + t_i \rangle$ to $\mathcal{A}$.

The above simulation fails if $H_0(R_S) = c_I$, but no R'_S can be found or $R'_S \neq R_S$. For the first case, the probability that $\mathcal{A}_\mathcal{I}$ can predict $H_0(R_S) = c_I$ without asking the random oracle is at most $1/2^\ell$. For the second case, collision must have occurred and the probability for this is at most $((q_H + q_S)(q_H + q_S + 1)/2)/2^\ell \leq (q_H + q_S + 1)^2/2^\ell$. We just assume $\mathcal{A}$ asked for $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$ if R'_S was not found since $\mathcal{A}$ knew the value of R_S before $\mathcal{B}$.

`Complete-Sign`: If both the SEM private key and the user private key are available, signing is trivial. If either one of them is unavailable, this request can be simulated faithfully as a combination of the above two simulation, but much easier since we no longer need the technicality to solve the problem that either one of the R_S and R_U is unknown.

`Forgery`: The next step of the simulation is to apply the general forking lemma in [7]: Let $\langle R^*, V^* \rangle$ be a forgery of a signature on message m^* with respect to $\langle ID^*, P_{ID^*}, W^* \rangle$ that is output by $\mathcal{A}_{\mathcal{I}}$ at the end of the attack. If $\mathcal{A}_{\mathcal{I}}$ does not output $ID^* = ID_I$ as a part of the forgery then $\mathcal{B}$ aborts (the probability that $\mathcal{B}$ does not abort the simulation is $O(1/q_{H_1})$).

Consider the case that W^* has not been replaced, i.e. $W^* = W_I = g^a Y^{-e_I}$ where $e_I = H_1(ID_I||W_I)$. $\mathcal{B}$ then replays $\mathcal{A}_{\mathcal{I}}$ with the same random tape but different H_2 after the point $(ID^*||W^*||0||P_{ID^*}||R^*||m^*)$. Suppose H_2 outputs h_S and h'_S in the first round and the second round respectively, where $h_S \neq h'_S$. Note the special step in the simulation of H_2 ensures $H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)$ remains the same after forking. Moreover, since $e_I = H_1(ID^*||W^*)$ is defined at the very beginning of the game, it remains the same as well.

So we get another valid forgery $\langle R^*, V' \rangle$, i.e.

$$R^* = g^{V^*} {P_{ID}}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-h_S}$$
$$R^* = g^{V'} {P_{ID}}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-h'_S}$$

$\mathcal{B}$ thus gets $V^* - ah_S = V' - ah'_S$. DLP's solution is $a = (V^* - V')/(h_S - h'_S)$.

In the second case, W^* is replaced. We would like to apply forking lemma so that $H_1(ID^*||W^*)$ changes from h to h' after forking, but H_2 queries related to W^* remain the same. Since W^* never appears in `Sign` query of any kind, it is thus safe to rearrange all those H_2 queries before the forking, without affecting the adversary's view. After forking, we get another valid forgery $\langle R^*, V' \rangle$ where

$$R^* = g^{V^*} {P_{ID}}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{h})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$
$$R^* = g^{V'} {P_{ID}}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{h'})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$

$\mathcal{B}$ solves the DLP by $a = (V^* - V')/(H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)(h - h'))$.

Theorem 2. *Our SMC signature scheme is existential unforgeable against the Type II adversary in the random oracle model assuming the DLP in $\mathbb{G}$ is hard.*

Proof. Let $\mathcal{A}_{\mathcal{II}}$ be a forger that breaks our scheme under adaptive chosen message attack. We show how $\mathcal{B}$ can use $\mathcal{A}_{\mathcal{II}}$ to solve the DLP instance (p, q, g, g^a).

$\mathcal{B}$ first randomly picks $s \in Z_q^*$, gives the system parameters as $(p, q, g, Y = g^s)$ and the master key s to $\mathcal{A}_{\mathcal{II}}$. It also maintains a list of tuples $(ID_i, W_i, e_i, d_i, P_i, x_i)$ which is denoted as H_1^{list}. Next, $\mathcal{B}$ randomly selects an index I such that $1 \leq I \leq q_{H_1}$, where q_{H_1} denotes the maximum number of queries to the random oracle H_1. $\mathcal{B}$ sets $P_I = g^a$, picks $z_I, e_I \in Z_q^*$ at random and computes $W_I = g^{w_I}$ and $d_I = z_I + se_I$ where $e_I = H_1(ID_I||W_I)$.

Adversary $\mathcal{B}$ interacts with $\mathcal{A}_{\mathcal{II}}$ in the `Attack` phase of the game as follows:

H_0 `Queries`: When $\mathcal{A}_{\mathcal{II}}$ queries H_0, $\mathcal{B}$ checks the corresponding H_0^{list} and outputs c_i if such query has already been made. Otherwise, $\mathcal{B}$ picks $c_i \in \{0,1\}^\ell$ at random, updates the H_0^{list} and outputs c_i as answer.

H_1 `Queries`: When $\mathcal{A}_{\mathcal{II}}$ queries H_1 on input $(ID_i||W_i)$, $\mathcal{B}$ checks the corresponding H_1^{list} and outputs e_i if such value is defined. Otherwise, $\mathcal{B}$ picks $e_i \in Z_q^*$ at random and outputs e_i as answer. H_1^{list} is also updated, such that $(ID_i, W_i, e_i, \perp, \tilde{\Delta}, \tilde{\Delta})$ is stored, where $\tilde{\Delta}$ means the old value (if exists) is kept.

H_2 `Queries`: When $\mathcal{A}_{\mathcal{II}}$ issues a query on these hash values, $\mathcal{B}$ parses the input as $(ID||\mu||b||\nu||R_i||m_i)$. If the query cannot be parsed as this form, simply returns a random value $h_i \in Z_q^*$. If $b = 0$, $\mathcal{B}$ sets $W_i = \mu$ and $P_i = \nu$. If $b = 1$, $\mathcal{B}$ sets $W_i = \nu$ and $P_i = \mu$. After that, $\mathcal{B}$ checks whether the value $H_1(ID||W_i)$ has been defined. If no, $\mathcal{B}$ runs the simulation of H_1 as usual to define it.

$\mathcal{B}$ then checks the corresponding H_2^{list}. If an entry for the query is found, the same answer will be given to $\mathcal{A}_{\mathcal{II}}$. Otherwise, $\mathcal{B}$ randomly selects h_i and h_i' from Z_q^*, assigns $h_i = H_2(ID||\mu||0||\nu||R_i||m_i)$ and $h_i' = H_2(ID||\nu||1||\mu||R_i||m_i)$. By symmetry, $H_2(ID||\nu||1||\mu||R_i||m_i)$ must have not been defined beforehand if $H_2(ID||\mu||0||\nu||R_i||m_i)$ has not. Finally, all queries and the corresponding answer will be stored in the H_2^{list}.

`SEM-Extract`: Note that at any time during the simulation, equipped with the master key s, $\mathcal{A}_{\mathcal{II}}$ is able to generate SEM private key for any ID.

`User-Extract`: Suppose the query is on ID_i. $\mathcal{B}$ responds as follow.

1. $i \neq I$: If the public key does not exist, $\mathcal{B}$ picks $x_i \in Z_q^*$ at random and updates the H_1^{list} (i.e. stores x_i and g^{x_i}). Otherwise, returns the x_i stored.
2. $i = I$: $\mathcal{B}$ aborts.

`SEM-Sign`: Note that at any time during the simulation, equipped with the master key s, $\mathcal{A}_{\mathcal{II}}$ is able to generate partial signatures $\langle R_S, t_i \rangle$ on any message.

`User-Sign`: Note that at any time during the simulation, equipped with those user private keys for any $ID_i \neq ID_I$, $\mathcal{A}_{\mathcal{II}}$ is able to generate partial signatures on any message. For $ID_i = ID_I$, the simulation is as follows.

1. An ℓ-bit commitment c_I is obtained from the adversary, which models the first message that SEM should be sent to initiate the `User-Sign` process.
2. Pick $\upsilon_i, \xi_i \in Z_q^*$ at random and set $R_U = g^{\upsilon_i} {P_I}^{(-\xi_i)}$.
3. Search for H_0^{list} to find R_S' such that $H_0(R_S') = c_I$.
4. If found, set $\xi_i = H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$.
5. After obtained $\langle R_S, t_i \rangle$, check if $H_0(R_S) = c_I$, stop `User-Sign`.
6. If R_S' is previously found, but $R_S' \neq R_S$, declare failure and abort.
7. If R_S' was not found, and $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m) \neq \xi_i$, declare failure and abort since $H_0(R_S) = c_I$ but $\mathcal{B}$ cannot program H_2 accordingly.
8. If $\mathcal{B}$ is lucky enough to reach this step, send $\langle R_U, \upsilon_i + t_i \rangle$ to $\mathcal{A}$.

The above simulation fails if $H_0(R_S) = c_I$, but no R'_S can be found or $R'_S \neq R_S$. For the first case, the probability that $\mathcal{A}_{\mathcal{I}}$ can predict $H_0(R_S) = c_I$ without asking the random oracle is at most $1/2^\ell$. For the second case, collision must have occurred and the probability for this is at most $((q_H + q_S)(q_H + q_S + 1)/2)/2^\ell \leq (q_H + q_S + 1)^2/2^\ell$. We just assume $\mathcal{A}$ asked for $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$ if R'_S was not found since $\mathcal{A}$ knew the value of R_S before $\mathcal{B}$.

`Complete-Sign`: If the user private key is available, signing is trivial. Otherwise, this request can be simulated faithfully similar to above, but much easier since we no longer need the technicality to solve the problem that R_S is unknown.

`Forgery`: The next step of the simulation is to apply the general forking lemma in [7]: Let $\langle R^*, V^* \rangle$ be a forgery of a signature on message m^* with respect to $\langle ID^*, P_{ID^*}, W^* \rangle$ that is output by $\mathcal{A}_{\mathcal{II}}$ at the end of the attack. If $\mathcal{A}_{\mathcal{II}}$ does not output $ID^* = ID_I$ as a part of the forgery then $\mathcal{B}$ aborts (the probability that $\mathcal{B}$ does not abort the simulation is $O(1/q_{H_1})$).

$\mathcal{B}$ then replays $\mathcal{A}_{\mathcal{II}}$ with the same random tape but different H_2 after the point $(ID^*||P_{ID^*}||1||W^*||R^*||m^*)$. Suppose H_2 outputs h_U and h'_U in the first round and the second round respectively, where $h_U \neq h'_U$. Since $e_I = H_1(ID^*||W^*)$ and $H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)$ are defined before $H_2(ID^*||P_{ID^*}$ $||1||W^*||R^*||m^*)$ outputs h_U and h'_U, they remain the same as well.

So we get another valid forgery $\langle R^*, V' \rangle$, i.e.

$$R^* = g^{V^*} {P_{ID}}^{-h_U} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$
$$R^* = g^{V'} {P_{ID}}^{-h'_U} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$

Since $P_{ID} = g^a$, $\mathcal{B}$ thus gets $V^* - ah_U = V' - ah'_U$. DLP's solution is $a = (V^* - V')/(h_U - h'_U)$. It works even the discrete logarithm of W^* is unknown.

5 Conclusion

We presented a formal study of security mediated certificateless signatures (SMC signatures). We showed that Ju *et al.*'s construction [17] is insecure. We formalize the security model of SMC signature, and discussed the subtleties arose from a public component introduced by the probabilistic SEM private key generation. We then proposed the first provable secure SMC signature scheme. Our scheme is efficient in the sense that no bilinear pairing is involved. It is provable secure in the random oracle model based on the intractability of the discrete logarithm problem. We also extended our scheme to achieve trust level 3 by adopting the technique used in [1,26]. It is worth noting that our scheme can be easily extended to be an efficient CLS scheme without pairing. Distributing the power of the SEM by threshold signature technique [25] will be our future work.

Acknowledgement

Sherman Chow is grateful for the comments of Alexander W. Dent, Joonsang Baek, Guilin Wang and Guomin Yang. This research is supported by the Malaysia IRPA grant (04-99-01-00003-EAR) and e-Science fund (01-02-01-SF0032).

References

1. Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless Public Key Cryptography. In *Proceedings of Asiacrypt 2003*, LNCS 2894, pp. 452-473. Full version available at Cryptology ePrint Archive, 2003/126.
2. Joonsang Baek, Reihaneh Safavi-Naini and Willy Susilo. Certificateless Public Key Encryption Without Pairing. In *Proc. of ISC 2005*, LNCS 3650, pp. 134-148.
3. Joonsang Baek and Yuliang Zheng. Identity-based Threshold Decryption. *Proc. of PKC 2004*, LNCS 2947, pp. 262–276.
4. Dan Boneh, Xuhua Ding and Gene Tsudik. Fine-Grained Control of Security Capabilities. *ACM Transactions on Internet Technology 2004*, 4/1, pp. 60-82.
5. Dan Boneh, Xuhua Ding, Gene Tsudik and Chi-Ming Wong. A Method for Fast Revocation of Public Key Certificates and Security Capabilities. In *USENIX 2001*.
6. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32/3, pp. 586–615, 2003.
7. Mihir Bellare and Gregory Neven. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma In *Proceedings of ACM-CCS 2006*, pp. 390-399.
8. Xuefei Cao, Kenneth G. Paterson and Weidong Kou. An Attack on a Certificateless Signature Scheme. Cryptology ePrint Archive, Report 2006/367.
9. Sherman S.M. Chow, Colin Boyd and Juan Manuel González Nieto. Security Mediated Certificateless Cryptography. In *Proceedings of PKC 2006*, LNCS 3958, pp. 508-524.
10. Alexander W. Dent. Personal communication, April 26, 2006.
11. Xuhua Ding and Gene Tsudik. Simple Identity-Based Cryptography with Mediated RSA. In *Proceedings of CT-RSA 2003*, LNCS 2612, pp. 193-210.
12. Marc Girault. Self-Certified Public Keys. In *Proceedings of Eurocrypt 1991*, LNCS 547, pp. 490-497.
13. M. Choudary Gorantla and Ashutosh Saxena. An Efficient Certificateless Signature Scheme. In *Computational Intelligence and Security 2005*, LNAI 3802, pp. 110-116.
14. Florian Hess. Efficient Identity Based Signature Schemes based on Pairings. In *Proceedings of SAC 2002*, LNCS 2595, pp. 310-324.
15. Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang and Xiaotie Deng. Key Replacement Attack Against a Generic Construction of Certificateless Signature. In *Proceedings of ACISP 2006*, LNCS 4058, pp. 235-246.
16. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. On the Security of Certificateless Signature Schemes from Asiacrypt 2003. In *Proceedings of CANS 2005*, LNCS 3810, pp. 13-25.
17. Hak Soo Ju, Dae Youb Kim, Dong Hoon Lee, Jongin Lim and Kilsoo Chun. Efficient Revocation of Security Capability in Certificateless Public Key Cryptography. In *Knowledge-Based Intelligent Information and Engineering Systems 2005*, LNAI 3682, pp. 453-459.
18. Joseph K. Liu, Man Ho Au and Willy Susilo. Self-Generated-Certificate Public Key Cryptography and Certificateless Signature / Encryption Scheme in the Standard Model. In *Proceedings of ASIACCS 2007*.

19. X. Li, K. Chen and L. Sun. Certificateless Signature and Proxy Signature Schemes from Bilinear Pairings. *Lithuanian Mathematical Journal*, 45/1, pp. 76–83, 2005.
20. Benoît Libert and Jean-Jacques Quisquater. Efficient Revocation and Threshold Pairing Based Cryptosystems. In *Proceedings of PODC 2003*, pp. 163-171.
21. Silvio Micali. Novomodo: Scalable Certificate Validation and Simplified PKI Management. In *Proceedings of 1st Annual PKI Research Workshop 2002*, pp. 15-25.
22. M. Myers, R. Ankney, A. alpani, S. Galperin and C. Adams. X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol (OCSP), *RFC 2560*.
23. Antonio Nicolosi, Maxwell Krohn, Yevgeniy Dodis, David Mazières. Proactive Two-Party Signatures for User Authentication. In *Proceedings of 10th NDSI 2003*.
24. Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In *Proceedings of CRYPTO 1989*, LNCS 435, pp. 239-252.
25. Jun Shao, Zhenfu Cao, and Licheng Wang. Efficient ID-based Threshold Signature Schemes without Pairings. Cryptology ePrint Archive, Report 2006/308.
26. Dae Hyun Yum and Pil Joong Lee. Generic Construction of Certificateless Signature. In *Proceedings of ACISP 2004*, LNCS 3108, pp. 200-211.
27. Zhenfeng Zhang, Duncan S. Wong, Jing Xu and Dengguo Feng. Certificateless Public-Key Signature: Security Model and Efficient Construction. In *Proceedings of ACNS 2006*, LNCS 3989, pp. 293-308. Journal version appeared in *Designs, Codes and Cryptography*, 42/2, pp. 109–126, 2007.

A Chow *et al.* vs. Ju *et al.*

Chow *et al.* [9] provided two constructions of encryption schemes: a generic construction in the standard model (which is earlier than [18]), with the restriction that decryption may not work after public key replacement, and a concrete scheme assuming random oracles, with the aforementioned restriction removed.

Ju *et al.*'s [17] security model for the encryption scheme does not allow the adversary to ask for the private key of the target user (*weak semantic security against insider attacks* in the term of [20]). Obviously, such assumption is too strong. They suggested to strengthen their scheme by using the technique of [3], but Chow *et al.*'s scheme [9] took a step further with a solution more elegant and efficient than such modification. Moreover, it is easy to show the hierarchical variant in [17] is insecure against chosen ciphertext attack.

B Errata of Chow *et al.*'s Generic SMC Encryption

Below give two errata of the generic SMC encryption in [9]. Both are about the encryption algorithm. Decryption algorithms should be revised accordingly.

1. Instead of $\alpha = H(C_1, C_2, \ell)$, α should be $H(C_1, C_2, \ell, s_1)$; otherwise, a chosen ciphertext attack is possible [10].
2. Instead of $(C_1, C_2) = (\mathsf{IBE.Enc}_{\mathsf{params}}(\mathsf{ID}_A, s_1), C_2 = \mathsf{PKE.Enc}^{\ell}_{\mathsf{EK}}(s_2))$, it should be $(C_1, C_2) = (\mathsf{IBE.Enc}_{\mathsf{params}}(\mathsf{ID}_A, s_1||H(\mathsf{VK})), C_2 = \mathsf{PKE.Enc}^{\mathsf{VK}}_{\mathsf{EK}}(s_2))$, i.e. the IBE encryption also encrypts the hash value of VK, and the label as the input of the public key encryption should be VK instead of the label ℓ of the SMC encryption. This amendment has been made in the PKC '06 presentation. Without such change, a chosen ciphertext attack is possible.

Gradually Convertible Undeniable Signatures (Michels-Petersen-Horster Convertible Undeniable Signatures Revisited)

Laila El Aimani and Damien Vergnaud

b-it COSEC - Bonn/Aachen International Center for Information Technology
Computer Security Group, Dahlmannstr. 2, D-53113 Bonn, Germany
{elaimani,vergnaud}@bit.uni-bonn.de

Abstract. In 1990, Boyar, Chaum, Damgård and Pedersen introduced *convertible undeniable signatures* which limit the self-authenticating property of digital signatures but can be converted by the signer to ordinary signatures. Michels, Petersen and Horster presented, in 1996, an attack on the Elgamal-based seminal scheme of Boyar *et al.* and proposed a repaired version without formal security analysis. In this paper, we modify their protocol so that it becomes a generic one and it provides an advanced feature which permits the signer to universally convert *achronously* all signatures pertaining to a specific time period. We supply a formal security treatment of the modified scheme: we prove, in the generic group model, that the protocol is existentially unforgeable and anonymous under chosen message attacks, assuming new assumptions (though reasonable) on the underlying hash function.

1 Introduction

In 1996, Michels, Petersen and Horster [14] proposed a convertible undeniable signature protocol whose security relies on the difficulty of the discrete logarithm problem in the multiplicative group of a finite field. This scheme has received little attention from the cryptographic community whereas we are convinced that it deserves better than oblivion. This paper focuses on the security treatment and on the proposal of an additional functionality for Michels-Petersen-Horster convertible undeniable signatures. Our analysis points out new security properties for the underlying hash functions which may be of independent interest.

Related work. A property of conventional digital signature schemes is that once a signature is released, everybody can check its validity. However there are numerous situations where this *self-authenticating* property is not desirable. In 1989 Chaum and van Antwerpen [7] introduced the concept of *undeniable signatures* whose purpose is to perform public key digital signatures which cannot be verified without interacting with the signer. In addition to the confidentiality and privacy concerns in themselves, this primitive finds applications in such different fields as electronic payment systems, certificate management or cyberdemocracy.

In 1991, the concept has been refined by giving the possibility to transform an undeniable signature into an ordinary digital signature. These *convertible undeniable signatures*, proposed in [4] by Boyar, Chaum, Damgård and Pedersen, provide individual and universal conversions of the signatures. Unfortunately, this Elgamal-like scheme has been broken in 1996 by Michels, Petersen, and Horster [14] who proposed a repaired version with heuristic security.

The universal conversion of all convertible undeniable signature protocols proposed before 2005, consists in revealing a part of the signer's secret key. This conversion makes all signatures, *past as well as future*, be universally verifiable. This property may be undesirable in some context since the corresponding keys cannot be used to generate undeniable signatures any more. To overcome this problem, Laguillaumie and the second author introduced and formalized, in 2005 [13], the *time-selective convertible undeniable signatures* which supports signers in gradually converting the undeniable signatures in a controlled fashion. They proposed a scheme which permits the signer to universally convert *chronologically* signatures pertaining only to a specific time period: given a time-selective convertible undeniable signature σ for a time period t, it is computationally infeasible to determine which signing secret key was used to generate σ; but with the knowledge of a matching universal receipt for some time period $p' \geq p$, it is easy to determine whether σ is a valid time-selective convertible undeniable signature or not. A tantalizing challenge is to generalize the concept of time-selective convertible signature to *event-selective* convertible signature where a signature becomes universally verifiable if a specific event happens that makes the signer publish the corresponding receipt information. This primitive will enable the signer to gradually convert signatures *achronously* (*i.e.* with time periods made completely independent of each other). Up to now, no concrete realization of this concept has been proposed in the literature.

Our contributions. In this paper, we revisit the Michels-Petersen-Horster convertible undeniable signature scheme. First of all, we modify it such that it becomes a generic algorithm. This point of view allows to look at cryptographic constructions in an abstract way and "move" them to other groups without the original restriction of subgroup of the multiplicative group of a finite field. In addition, we suggest a slight modification of this scheme which gives the first realization of *achronous* gradually convertible undeniable signatures.

The security of many cryptographic tools relies on assumptions about the hardness of certain algorithmic problems. Techniques from [17] suggest that it is highly improbable to reduce the security of the Michels-Petersen-Horster signatures to the discrete logarithm problem in the standard security model. Therefore, we investigate their security in the so-called *generic group model*, following previous work from [5,21] where the security of a generic version of the protocol DSA was analyzed. However, it is worth noting that the real, non-generic security of the scheme may be completely different in different groups [8].

This security analysis points out new sufficient security properties for the underlying hash functions. These new notions of *random affine preimage resistance* and *random linear collision resistance* are satisfied by generic hash functions (*i.e.* in the random oracle model [3]). The former property is necessary for our scheme to be secure, while the latter is for the RSA-FDH signature scheme [3].

Notations. The set of n-bit strings is denoted by $\{0,1\}^n$ and the set of all finite binary strings (or messages) is denoted by $\{0,1\}^*$. Let $\mathcal{A}$ be a probabilistic Turing machine running in polynomial time (a PPTM, for short), and let x be an input for $\mathcal{A}$. The probability space that assigns to a string σ the probability that $\mathcal{A}$, on input x, outputs σ is denoted by $\mathcal{A}(x)$. The support of $\mathcal{A}(x)$ is denoted by $\mathcal{A}[x]$. Given a probability space S, a PPTM that samples a random element according to S is denoted by $x \overset{R}{\longleftarrow} S$. For a finite set X, $x \overset{R}{\longleftarrow} X$ denotes a PPTM that samples a random element uniformly at random from X. A *two-party protocol* is a pair of interactive PPTMs $(\mathsf{Prove}, \mathsf{Verify})$.

2 Gradually Convertible Undeniable Signatures

2.1 Definition

As in ordinary digital signatures, undeniable signature schemes establish two complimentary algorithms: one for signing (Sign) and the other for controlling the signature at some later time (Cont), but this algorithm is not publicly available since it requires the knowledge of the signer's secret key to be executed. Besides, the signer can prove his authorship of an undeniable signature by running a confirmation protocol (Conf) with a verifier and a falsely implicated signer may deny his involvement by running a denial protocol (Deny) with a verifier.

Designated verifier proofs were introduced by Jakobsson, Sako and Impagliazzo in 1996 [11] and have been widely used for undeniable signature schemes. In [12], Kudla and Paterson present a security model for these signatures where the confirmation and denial protocols are actually implemented with such proofs. They proposed non-interactive designated verifier proofs suited to combination with Chaum-van Antwerpen original undeniable signature scheme resulting in a secure[1] and efficient undeniable signature scheme. Unfortunately, we cannot use these non-interactive non-transferable proofs, to obtain the security results without the random oracle model. Therefore, in this paper, we will use interactive version of the designated verifier proofs described in [12].

In addition, the signer has at its disposal an algorithm (Conv) which permits to:

- convert a given undeniable signature into a regular, universally verifiable signature. This operation does not affect other undeniable signatures.
- publish a universal trapdoor relative to a specific time period p by the means of which all undeniable signatures for the time period p become universally verifiable.

[1] In the random oracle model, assuming the intractability of the decisional Diffie-Hellman problem in the underlying group [9,15,16].

The verification of the converted signatures is performed thanks to the algorithm Vf.

Definition 1 (Gradually Convertible Undeniable Signature). *Let* $\pi \in \mathbb{N}$. *A* gradually convertible undeniable signature scheme *with* π *time periods* Σ *is a 9-tuple* $\Sigma = (\mathsf{Setup}, \mathsf{SKg}, \mathsf{VKg}, \mathsf{Sign}, \mathsf{Cont}, \mathsf{Conf}, \mathsf{Deny}, \mathsf{Conv}, \mathsf{Vf})$ *such that:*

– Σ.*Setup, the* common parameter generation algorithm, *is a PPTM which takes an integer* k *as input. The output are the* public parameters $\mathcal{P}$. k *is called the* security parameter.
– Σ.*SKeyGen, the* signer key generation algorithm, *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathbf{sk_s}, \mathbf{pk_s})$ *where* $\mathbf{sk_s}$ *is called a* signing secret key *and* $\mathbf{pk_s}$ *a* signing public key.
– Σ.*VKeyGen, the* verifier key generation algorithm, *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathbf{sk_v}, \mathbf{pk_v})$ *where* $\mathbf{sk_v}$ *is called a* verifying secret key *and* $\mathbf{pk_v}$ *a* verifying public key.
– Σ.*Sign, the* signing algorithm, *is a PPTM which takes the public parameters, a message, an integer in* $[\![1, \pi]\!]$ *and a signing secret key as inputs and outputs a bit string.*
– Σ.*Cont, the* controlling algorithm, *is a PPTM which takes the public parameters, a message* m, *a bit string* σ, *an integer* $p \in [\![1, \pi]\!]$ *and a signing key pair* $(\mathbf{sk_s}, \mathbf{pk_s})$ *as inputs and outputs a bit. If the bit output is 1 then the bit string* σ *is said to be a* signature *on* m *for* $\mathbf{pk_s}$ *for the time period* p.
– $\Sigma.\{$*Conf.Deny*$\}$, *the* confirming/denying protocols *(respectively), are two-party protocols* (*Prove*, *Verify*) *such that:*
  - *Prove and Verify take as input a message* m, *an integer* $p \in [\![1, \pi]\!]$, *a bit-string* σ, *a signing public key* $\mathbf{pk_s}$ *and a verifying public key* $\mathbf{pk_v}$ *and the public parameters;*
  - *Prove takes as input* $\mathbf{sk_s}$ *the signing secret key corresponding to* $\mathbf{pk_s}$*;*
  - *Verify takes as input* $\mathbf{sk_v}$ *the verifying secret key corresponding to* $\mathbf{pk_v}$*;*

  *Conf.Verify (*resp. *Deny.Verify) outputs an element in* $\{\perp, 1\}$ *(*resp. $\{\perp, 0\}$*).*
– Σ.*Conv, the* conversion algorithm, *is a PPTM which takes as input the public parameters, an integer in* $[\![1, \pi]\!]$, *a signing key pair and a bit string* Υ *(either a pair message/signature or the empty string) and outputs a bit string.*
– Σ.*Vf, the* verifying algorithm for converted signature, *is a PPTM which takes as input the public parameters, a message* m, *and a bit string* σ, *an integer* $p \in [\![1, \pi]\!]$, *a signing public key* $\mathbf{pk_s}$ *and a bit string* Λ *and outputs a bit. If the bit output is* 1 *then the bit string* Λ *is said to be a* receipt *of the validity of* σ.

where the protocols Σ.*Conf and* Σ.*Deny are a designated verifier proof of membership system for the languages (respectively):*

$$\{(\mathcal{P}, m, \sigma, p, \mathbf{pk_s}) \in \Sigma.\mathit{Setup}[k] \times \{0,1\}^{*2} \times [\![1, \pi]\!] \times \Sigma.\mathit{SKg}[\mathcal{P}] \big| \Sigma.\mathit{Vf}[\mathcal{P}, m, \sigma, p]\} = \{1\}$$
$$\{(\mathcal{P}, m, \sigma, p, \mathbf{pk_s}) \in \Sigma.\mathit{Setup}[k] \times \{0,1\}^{*2} \times [\![1, \pi]\!] \times \Sigma.\mathit{SKg}[\mathcal{P}] \big| \Sigma.\mathit{Vf}[\mathcal{P}, m, \sigma, p]\} = \{0\}$$

and for all $k \in \mathbb{N}$, for all $\mathcal{P} \in \Sigma.\mathsf{Setup}[k]$, for all $\mathcal{S} = (\mathbf{pk_s}, \mathbf{sk_s}) \in \Sigma.\mathsf{SKg}[\mathcal{P}]$, for all $m \in \{0,1\}^*$ and for all $p \in [\![1, \pi]\!]$, we have:

$$\forall \sigma \in \Sigma.\mathsf{Sign}[\mathcal{P}, m, p, \mathbf{sk_s}], \Sigma.\mathsf{Cont}[\mathcal{P}, m, \sigma, p, (\mathbf{sk_s}, \mathbf{pk_s})] = \{1\}$$
$$\forall \sigma \in \Sigma.\mathsf{Sign}[\mathcal{P}, m, p, \mathbf{sk_s}], \forall \Lambda \in \Sigma.\mathsf{Conv}[\mathcal{P}, p, \mathcal{S}, (m, \sigma)], \Sigma.\mathsf{Vf}[\mathcal{P}, m, \sigma, p, \mathbf{pk_s}, \Lambda] = \{1\}$$
$$\forall \sigma \in \Sigma.\mathsf{Sign}[\mathcal{P}, m, p, \mathbf{pk_s}], \forall \Lambda \in \Sigma.\mathsf{Conv}[\mathcal{P}, p, \mathcal{S}, \varepsilon], \Sigma.\mathsf{Vf}[\mathcal{P}, m, \sigma, p, \mathbf{pk_s}, \Lambda] = \{1\}$$
$$\forall \sigma, \Lambda \in \{0,1\}^*, \Sigma.\mathsf{Vf}[\mathcal{P}, m, \sigma, p, \mathbf{pk_s}, \Lambda] = \{1\} \Rightarrow \Sigma.\mathsf{Cont}[\mathcal{P}, m, \sigma, p, (\mathbf{sk_s}, \mathbf{pk_s})] = \{1\}.$$

Remark 1. The first two properties capture the validity and the non-transferable property of the protocols Conf and Deny (*i.e.* the use of designated verifier proofs insures that a verifier will gain no information in an execution of one of these protocols [12]). The three last properties are the properties of *correctness*:

– a well-formed signature is always accepted by the algorithm Cont;
– a receipt correctly constructed is always accepted by the algorithm Vf;
– and if there exists a bit-string Λ which makes accepted a bit-string σ by the algorithm Vf, then σ is a valid signature.

2.2 Security Model

Registered public key model. In public key cryptography, the notion of anonymity is to be handled with great attention. For instance, in order to ensure anonymity, it is important that users register their public key by a certifying authority. Hence, in our security analysis, it is assumed that the users' keys have been already registered to an authority. The registration procedure would always contain a proof of knowledge of the associated private key. To further simplify the security analysis, we will assume that this procedure will be the *direct registration of the keys*[2].

Security against existential forgery under chosen message attack. The standard notion of security for digital signatures was defined by Goldwasser, Micali and Rivest [10] as *existential forgery against adaptive chosen message attacks* (EF-CMA). In [13], the corresponding notion for time-selective convertible undeniable signatures is defined along the same lines. The definition of *resistance to forgery* for gradually convertible undeniable signatures that we propose is similar. In fact, we suppose that the adversary has access to the universal receipts for every time period $p \in [\![1, \pi]\!]$ and is allowed to query a converting oracle $\mathfrak{Cv}$, a confirming oracle $\mathfrak{C}$ amd a denying oracle $\mathfrak{D}$ on any couple message/signature of its choice. As usual, in the adversary answer, there is the natural restriction that the returned message/signature has not been obtained from the signing oracle.

Definition 2 (Unforgeability - EF-CMA). *Let π be a positive integer, let $\Sigma = (\mathsf{Setup}, \mathsf{SKg}, \mathsf{VKg}, \mathsf{Sign}, \mathsf{Cont}, \mathsf{Conf}, \mathsf{Deny}, \mathsf{Conv}, \mathsf{Vf})$ be a gradually convertible*

[2] It is often necessary to require the security of the schemes even if the adversary is the key registration center. In this case, one must replace the proof of knowledge associated to the key registration by a zero-knowledge one.

undeniable signature scheme with π time periods and let $\mathcal{A}$ be an PPTM. We consider the following random experiment, where k is a security parameter:

```
$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ef-cma}}_{\Sigma,\mathcal{A}}(k)}$
$\mathcal{P} \overset{R}{\longleftarrow} \Sigma.\mathsf{Setup}(k),$
$(\mathbf{pk_s}, \mathbf{sk_s}) \overset{R}{\longleftarrow} \Sigma.\mathsf{SKg}(\mathcal{P})$
$\mathsf{for}\ j = 1\ \mathsf{to}\ \pi\ \mathsf{do}\ \Lambda_j \leftarrow \Sigma.\mathsf{Conv}(\mathcal{P}, j, (\mathbf{sk_s}, \mathbf{pk_s}), \varepsilon)$
$(m^\star, \sigma^\star, p^\star) \overset{R}{\longleftarrow} \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{C},\mathfrak{D}}(\mathcal{P}, \mathbf{pk_s}, \{\Lambda_j\}_{j\in[\![1,\pi]\!]})$
        | $\mathfrak{S} : (m, p) \longrightarrow \Sigma.\mathsf{Sign}(\mathcal{P}, m, p, \mathbf{sk_s})$
        | $\mathfrak{Cv} : (m, p, \sigma) \longrightarrow \Sigma.\mathsf{Conv}(\mathcal{P}, p, (\mathbf{sk_s}, \mathbf{pk_s}), (m, \sigma))$
        | $\mathfrak{C} : (m, p, \sigma, \mathbf{pk_v}) \longrightarrow \Sigma.\mathsf{Conf}(m, p, \sigma, \mathbf{pk_v}, \mathbf{pk_s})$
        | $\mathfrak{D} : (m, p, \sigma, \mathbf{pk_v}) \longrightarrow \Sigma.\mathsf{Deny}(m, p, \sigma, \mathbf{pk_v}, \mathbf{pk_s})$
return 1 if and only if the following properties are satisfied:
    - $\Sigma.\mathsf{Vf}[\mathcal{P}, \mathbf{pk_s}, m^\star, \sigma^\star, \Lambda_{p^\star}] = \{1\}$
    - $m$ was not queried to $\mathfrak{S}$
```

We define the success *of $\mathcal{A}$,* via $\mathbf{Succ}^{\textit{ef-cma}}_{\Sigma,\mathcal{A}}(k) \Pr\left[\mathbf{Exp}^{\textit{ef-cma}}_{\Sigma,\mathcal{A}}(k) = 1\right]$.
Given $(k, t) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, the scheme Σ is said to be (k, t, ε)-EF-CMA secure, if no EF-CMA-adversary $\mathcal{A}$ running in time t has $\mathbf{Succ}^{\textit{ef-cma}}_{\Sigma,\mathcal{A}}(k) \geq \varepsilon$. The scheme Σ is said to be EF-CMA secure if, for any security parameter $k \in \mathbb{N}$, any polynomial function $t : \mathbb{N} \to \mathbb{N}$, and any negligible function $\varepsilon : \mathbb{N} \to [0, 1]$, it is $(k, t(k), \varepsilon(k))$-EF-CMA secure.

Anonymity. We state the precise definition of *anonymity* under a chosen message attack (Ano-CMA) which captures the notion that an attacker cannot determine under which key a signature was performed [9]. We consider a Ano-CMA-adversary $\mathcal{A}$ that runs in two stages. In the find stage, it takes as input two signing public keys $\mathbf{pk_{s0}}$ and $\mathbf{pk_{s1}}$ and outputs a message $m^\star$, a time period $p^\star$ together with some state information $\mathcal{I}$. In the guess stage, $\mathcal{A}$ gets a challenge gradually convertible undeniable signature $\sigma^\star$ formed by signing at random the message $m^\star$ under one of the two keys for the time period $p^\star$ and it must say which key was chosen. In both stages, the adversary has access to a signing oracle $\mathfrak{S}$ for both signing key pairs, to a converting oracle $\mathfrak{Cv}$, to a confirming oracle $\mathfrak{C}$ and to a denying oracle $\mathfrak{D}$. The attacker is also given the universal receipts of both potential signers for all[3] time period $p \in [\![1, \pi]\!] \setminus \{p^\star\}$. The only restriction on $\mathcal{A}$ is that it cannot query the triple $(m^\star, \sigma^\star, p^\star)$ on the converting and confirming/denying oracles.

Definition 3 (Anonymity - Ano-CMA). *Let π be a positive integer, let $\Sigma = (\mathsf{Setup}, \mathsf{SKg}, \mathsf{VKg}, \mathsf{Sign}, \mathsf{Cont}, \mathsf{Conf}, \mathsf{Deny}, \mathsf{Conv}, \mathsf{Vf})$ be a gradually convertible undeniable signature scheme with π time periods and let $\mathcal{A}$ be an PPTM. We consider the following random experiment, for $r \in \{0, 1\}$, where k is a security parameter:*

[3] This is the main difference with time-selective convertible undeniable signatures from [13] where this universal receipts was given only for $p \in [\![1, p^\star - 1]\!]$.

```
$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{ano-cma}-r}_{\Sigma,\mathcal{A}}(k)}$
$\mathcal{P} \xleftarrow{R} \Sigma.\textsf{Setup}(k)$
$(\mathbf{pk_{s}}_0, \mathbf{sk_{s}}_0) \xleftarrow{R} \Sigma.\textsf{SKeyGen}(\mathcal{P})$,
$(\mathbf{pk_{s}}_1, \mathbf{sk_{s}}_1) \xleftarrow{R} \Sigma.\textsf{SKeyGen}(\mathcal{P})$
$(m^\star, p^\star, \mathcal{I}) \xleftarrow{R} \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{C},\mathfrak{D}}(\textsf{find}, \mathcal{P}, \mathbf{pk_{s}}_0, \mathbf{pk_{s}}_1)$
                  | $\mathfrak{S} : (m, p, i) \longrightarrow \Sigma.\textsf{Sign}(\mathcal{P}, m, p, \mathbf{sk_{s}}_i)$
                  | $\mathfrak{Cv} : (m, p, \sigma, i) \longrightarrow \Sigma.\textsf{Conv}(\mathcal{P}, p, (\mathbf{sk_{s}}_i, \mathbf{pk_{s}}_i), (m, \sigma))$
                  | $\mathfrak{C} : (m, p, \sigma, \mathbf{pk_v}, i) \longrightarrow \Sigma.\textsf{Conf}(m, p, \sigma, \mathbf{pk_v}, \mathbf{pk_{s}}_i)$
                  | $\mathfrak{D} : (m, p, \sigma, \mathbf{pk_v}, i) \longrightarrow \Sigma.\textsf{Deny}(m, p, \sigma, \mathbf{pk_v}, \mathbf{pk_{s}}_i)$
$\sigma^\star \xleftarrow{R} \Sigma.\textsf{Sign}(\mathcal{P}, m, \mathbf{sk_{s}}_r, p^\star)$
for $j$ from $1$ to $\pi$ do
    $\Lambda_j^0 \leftarrow \Sigma.\textsf{Conv}(\mathcal{P}, j, \mathbf{pk_{s}}_0, \mathbf{sk_{s}}_0, \varepsilon)$ and $\Lambda_j^1 \xleftarrow{R} \Sigma.\textsf{Conv}(\mathcal{P}, j, \mathbf{pk_{s}}_1, \mathbf{sk_{s}}_1, \varepsilon)$
$d \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{C},\mathfrak{D}}(\textsf{guess}, \mathcal{I}, \{\Lambda_j^0, \Lambda_j^1\}_{j \in [\![1,\pi]\!] \setminus \{p^\star\}})$
Return $d$
```

We define the advantage *of $\mathcal{A}$,* via

$$\mathbf{Adv}^{\textsf{ano-cma}}_{\Sigma,\mathcal{A}}(k) \left| \Pr\left[\mathbf{Exp}^{\textsf{ano-cma}-1}_{\Sigma,\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\textsf{ano-cma}-0}_{\Sigma,\mathcal{A}}(k) = 1\right]\right| .$$

Given $(k,t) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$, the scheme Σ is said to be (k,t,ε)-Ano-CMA secure, if no Ano-CMA-adversary $\mathcal{A}$ running in time t has $\mathbf{Adv}^{\textsf{ano-cma}}_{\Sigma,\mathcal{A}}(k) \geq \varepsilon$. The scheme Σ is said to be Ano-CMA secure if, for any security parameter $k \in \mathbb{N}$, any polynomial function $t : \mathbb{N} \rightarrow \mathbb{N}$, and any negligible function $\varepsilon : \mathbb{N} \rightarrow [0,1]$, it is $(k, t(k), \varepsilon(k))$-Ano-CMA secure.

3 Hash Functions and New Security Properties

Hash functions take messages of arbitrary length and outputs a fixed length string. In cryptographic uses of a hash function $\mathcal{H} : \{0,1\}^* \longrightarrow H$, these properties are considered prerequisites:

– *Preimage resistance*: given $h \in H$, it should be computationally intractable to find a message m such that $\mathcal{H}(m) = h$.
– *Collision-resistant:* it should be computationnally intractable to find two different messages m_1 and m_2 such that $\mathcal{H}(m_1) = \mathcal{H}(m_2)$.

In this section, we formulate generalization of these security notions and study their properties.

Definitions. The proof of security of our variant of Michels-Petersen-Horster signatures makes use of new non-standard variations of the preimage resistance and the collision resistance assumptions for hash functions. These assumptions are of independent interest as they have interesting relations with the classical ones. We call them *random affine preimage resistance* and *random linear collision resistance.* Although stronger than the standard assumptions, they are quite realistic.

According to [18], an hash function family is a family of functions $(\mathcal{H}_k : \mathcal{K}_k \times \{0,1\}^* \longrightarrow \{0,1\}^k)_{k\in\mathbb{N}}$, where $\mathcal{K}_k$ is a finite non-empty set. We will write the first argument of $\mathcal{H}_k$ as a subscript, so that $\mathcal{H}_{K,k}(m) = \mathcal{H}_k(K,m)$. In the following, we denote elements from $\{0,1\}^k$ as the corresponding k-bits integers in binary representation and we will denote for every integer $N \in \mathbb{Z}$, $\mathcal{H}^N_{K,k}$ the map defined by: $\mathcal{H}^N_{K,k} : \begin{cases} \{0,1\}^* \longrightarrow \mathbb{Z}_N \\ \quad m \longmapsto \mathcal{H}_{K,k}(m) \mod N. \end{cases}$

The new security definitions can be quantified as follows:

Definition 4 (Random affine preimage resistance). *Let n be an integer, let $(\mathcal{H}_k : \mathcal{K}_k \times \{0,1\}^* \longrightarrow \{0,1\}^k)_{k\in\mathbb{N}}$ be an hash function family and let $\mathcal{A}$ be a PPTM. The success $\mathbf{Succ}^{raPre(n)}_{\mathcal{H},\mathcal{A}}(k)$ of $\mathcal{A}$ against the n-random affine preimage resistance of $\mathcal{H} = (\mathcal{H}_k)_{k\in\mathbb{N}}$ is defined by:*

$$\max_{\substack{2^{k-1}\le N<2^k \\ \alpha_1,\dots,\alpha_n\in\mathbb{Z}_N^* \\ \beta_1,\dots,\beta_n\in\mathbb{Z}_N^*}} \left\{ \Pr\left[\begin{array}{c} K \overset{R}{\longleftarrow} \mathcal{K}_k; (m,i,j) \overset{R}{\longleftarrow} \mathcal{A}(K,\alpha_1,\dots,\alpha_n,\beta_1,\dots,\beta_n) \\ m \in \{0,1\}^*, (i,j) \in [\![1,n]\!]^2, i \neq j \\ \alpha_i + \beta_j \mathcal{H}^N_{K,k}(m) = 0 \mod N \end{array} \right] \right\}.$$

An adversary $\mathcal{A}$ against the n-random affine preimage resistance of a hash function family $(\mathcal{H}_k)_{k\in\mathbb{N}}$ can be transformed easily into an adversary against the classical preimage resistance of $(\mathcal{H}_k)_{k\in\mathbb{N}}$ with success probability greater than $\mathbf{Succ}^{raPre(n)}_{\mathcal{H},\mathcal{A}}(k)/n^2$ and time-complexity of $\mathcal{A}$ increased by the time necessary to compute n modular multiplications modulo N. In particular, the 1-random affine preimage resistance is equivalent to the classical preimage resistance.

Definition 5 (Random linear collision resistance). *Let n be an integer, let $(\mathcal{H}_k : \mathcal{K}_k \times \{0,1\}^* \longrightarrow \{0,1\}^k)_{k\in\mathbb{N}}$ be an hash function family and let $\mathcal{A}$ be a PPTM. The success $\mathbf{Succ}^{rlColl(n)}_{\mathcal{H},\mathcal{A}}(k)$ of $\mathcal{A}$ against the n-random affine preimage resistance of $\mathcal{H} = (\mathcal{H}_k)_{k\in\mathbb{N}}$ is defined by:*

$$\max_{\substack{2^{k-1}\le N<2^k \\ \lambda_1,\dots,\lambda_n\in\mathbb{Z}_N^*}} \left\{ \Pr\left[\begin{array}{c} K \overset{R}{\longleftarrow} \mathcal{K}_k; (m,m',i,j) \overset{R}{\longleftarrow} \mathcal{A}(K,\lambda_1,\dots,\lambda_n) \\ m, m' \in \{0,1\}^*, (i,j) \in [\![1,n]\!]^2, m \neq m' \\ \lambda_i \cdot \mathcal{H}_{K,N}(m) = \lambda_j \cdot \mathcal{H}_{K,N}(m') \mod N \end{array} \right] \right\}.$$

As for random affine preimage resistance, the 1-random linear collision resistance is equivalent to the classical collision resistance. Unfortunately, it is impossible to prove that the n-random linear collision resistance can be reduced generically to the collision resistance for $n \geq 2$.

Remark 2. This security requirement is however reasonable since if the hash function family underlying the protocol RSA-FDH [3] does not satisfy it, then it is existential forgeable against a *one* chosen-message attack: given an RSA public key (N,e), the adversary can simply pick at random $r_1,\dots,r_n \in \mathbb{Z}_N$, compute $\lambda_i = r_i^e \mod N$ for all $i \in [\![1,n]\!]$, and try to find a random linear collision with parameters $N, \lambda_1,\dots,\lambda_n$. If a collision $m, m' \in \{0,1\}^*$, $(i,j) \in [\![1,n]\!]^2$, (such that

$\lambda_i \cdot \mathcal{H}_{K,N}(m) = \lambda_j \cdot \mathcal{H}_{K,N}(m') \mod N$ is found), then the adversary queries the signature σ on m to the signing oracle and can compute the signature of m' as $\sigma' = r_i \cdot \sigma \cdot r_j^{-1} \mod N$.

Generic security. The best known general collision-finding attack against a hash function family is the so-called birthday-attack. If we assume that the values of the hash-function family $(\mathcal{H}_k)_{k\in\mathbb{N}}$ are uniformly distributed over $\{0,1\}^k$ and that the generalisation of the birthday attack[4] against the random affine preimage resistance and the random linear collision resistance of $(\mathcal{H}_k)_{k\in\mathbb{N}}$ is the best possible attack (which is true in the random oracle model), then it is possible to give exponential lower bounds on the minimum of n and of the number of hash functions evaluation required to have non-negligible probability of success. Indeed, for any integer $N \geq 2$, and for $(i,k) \in \mathbb{Z}_N$, it is straightforward [20] that

$$\#\{j \in \mathbb{Z}_N | i \cdot j \mod N \leq k\} = \gcd(i,N) \times \left(\left\lfloor \frac{k}{\gcd(i,N)} \right\rfloor + 1 \right).$$

Therefore if D denotes the product of two independent random variables uniformly distributed over $\mathbb{Z}_N$, we have $\forall k \in \mathbb{Z}_N$

$$\Pr(D \leq k) = \frac{1}{N^2} \sum_{i=0}^{N-1} \gcd(i,N) \left(\left\lfloor \frac{k}{\gcd(i,N)} \right\rfloor + 1 \right),$$

and consequently, D is close to the uniform distribution ove $\mathbb{Z}_N$. The results from [2] are sufficient to conclude; details will appear elsewhere.

4 Michels-Petersen-Horster Convertible Undeniable Signatures Revisited

4.1 Description of the Scheme

Let π be an integer. Following the notations from § 2.1, we describe in this section our variant of Michels-Petersen-Horster scheme. It is parameterized by a prime order group generator [1], an hash function family and two pseudo-random function families [18].

Let $\mathbb{G}$ be a group of prime order q generated by the prime order group generator. A *reduction function* is a map that sends an element of the group $\mathbb{G}$ [5,21] to an integer in $\mathbb{Z}_q$. In our security analysis, the reduction function must satisfy the so called *almost-invertibility*: given an arbitrary integer in $\mathbb{Z}_q$, then with nonnegligeable probability one can efficiently find one preimage.

[4] These attacks consist in picking messages $m_1, \ldots, m_r$, computing $h_i = \mathcal{H}_k(m_i) \mod N$ for $i \in [\![1,r]\!]$ and $\gamma_{i,j} = -h_i\beta_j \mod N$ (*resp.* $\gamma_{i,j} = h_i\lambda_j \mod N$) for $j \in [\![1,n]\!]$. They are successful if there is a triple $(i,j,\ell) \in [\![1,r]\!] \times [\![1,n]\!]^2$ (*resp.* a 4-tuple $(i,i',j,j') \in [\![1,r]\!]^2 \times [\![1,n]\!]^2$) s. t. $\gamma_{i,j} = \alpha_\ell$ (*resp.* $\gamma_{i,j} = \gamma_{i',j'}$ and $j \neq j'$).

Definition 6. *Let F be a reduction function $F : \mathbb{G} \to \mathbb{Z}_q$. An almost-inverse of F is a probabilistic algorithm G, possibly outputting $\perp$, such that:*

$$\Pr_{b \in_R \mathbb{Z}_q}[G(b) \in S \wedge F(G(b)) = b] \geq \frac{1}{3}.$$

A reduction function F is (δ, t)-almost-invertible, with almost-inverse G, if furthermore: $\mathcal{D} \approx_\delta \mathcal{U}$ where $\mathcal{D} = \{G(b) \mid b \xleftarrow{R} \mathbb{Z}_q \wedge G(b) \in \mathbb{G}\}$ and $\mathcal{U} = \{a \mid a \xleftarrow{R} \mathbb{G}\}$. The notation $\mathcal{D} \approx_\delta \mathcal{U}$ means that no distinguisher with running time t can get an advantage greater than δ.

Description of the scheme

– Σ.Setup: on input a security parameter k, the underlying generators output a group $\mathbb{G}$ of prime order q generated by an element P, a reduction function $F : \mathbb{G} \to \mathbb{Z}_q$, a hash function $h : \{0,1\}^* \to \mathbb{Z}_q$ and two pseudo-random functions $H^1 : \mathbb{Z}_q \times [\![1, \pi]\!] \to \{0,1\}^k$ and $H^2 : \{0,1\}^k \times \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_q$. The public parameters are $(q, \mathbb{G}, P, h, H^1, H^2)$.

– Σ.SKg: The signer picks at random its secret key $u, v \xleftarrow{R} [\![1, q-1]\!]$, computes $U \leftarrow uP$ and $V \leftarrow vP$ and sets (U, V) as its public key.

– Σ.VKg: The verifier picks at random its secret key $w \xleftarrow{R} [\![1, q-1]\!]$, computes $W \leftarrow wP$. and set it as its public key.

Protocol EDL.Prove

Common input: (U_1, U_2, V_1, V_2), Y
$\mathcal{P}$'s input: x
$\mathcal{V}$'s output: b

① $\mathcal{P} \xrightarrow{\quad C_1, C_2, C_3 \quad} \mathcal{V}$
$(a, b, k) \xleftarrow{R} [\![1, q-1]\!]^3$
$C_1 \leftarrow [k] \cdot U_1$; $C_2 \leftarrow [k] \cdot U_2$
$C_3 \leftarrow [a] \cdot U_1 + [b] \cdot Y$
❶ $\mathcal{V} \xrightarrow{\quad r \quad} \mathcal{P}$
$r \xleftarrow{R} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{\quad a, b, c \quad} \mathcal{V}$
$c \leftarrow k - x(r+b) \mod q$

• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow [c] \cdot U_1 + [r+b] \cdot V_1$
$\widetilde{C_2} \leftarrow [c] \cdot U_2 + [r+b] \cdot V_2$
$\widetilde{C_3} \leftarrow [a] \cdot U_1 + [b] \cdot Y$
if $(C_1, C_2, C_3)(\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3})$
  then $b \leftarrow$ Accept **else** $b \leftarrow \perp$

Protocol EDL.Fake

Common input: (U_1, U_2, V_1, V_2), Y
$\mathcal{P}$'s input: y
$\mathcal{V}$'s output: b

① $\mathcal{P} \xrightarrow{\quad C_1, C_2, C_3 \quad} \mathcal{V}$
$(c, d, k) \xleftarrow{R} [\![1, q-1]\!]^3$
$C_1 \leftarrow [c] \cdot U_1 + [d] \cdot V_1$; $C_2 \leftarrow [c] \cdot U_2 + [d] \cdot V_2$
$C_3 \leftarrow [k] \cdot U_1$
❶ $\mathcal{V} \xrightarrow{\quad r \quad} \mathcal{P}$
$r \xleftarrow{R} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{\quad a, b, c \quad} \mathcal{V}$
$b \leftarrow d - r \mod q$; $a \leftarrow k - by \mod q$

• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow [c] \cdot U_1 + [r+b] \cdot V_1$
$\widetilde{C_2} \leftarrow [c] \cdot U_2 + [r+b] \cdot V_2$
$\widetilde{C_3} \leftarrow [a] \cdot U_1 + [b] \cdot Y$
if $(C_1, C_2, C_3)(\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3})$
  then $b \leftarrow$ Accept **else** $b \leftarrow \perp$

Fig. 1. Interactive designated verifier proof of membership of the language EDL($\mathbb{G}$)

```
Protocol IDL.Prove
Common input: $(U_1, U_2, V_1, V_2)$, $Y$
$\mathcal{P}$'s input : $x$
$\mathcal{V}$'s output : $b$
① $\mathcal{P} \xrightarrow{C_0, C_1, C_2, C_3} \mathcal{V}$
$(a, b, k_0, k_1, k_2) \overset{R}{\longleftarrow} [\![1, q-1]\!]^5$
$C_0 \leftarrow [k_0] \cdot (V_2 - [x] \cdot U_2)$
$C_1 \leftarrow [k_1] \cdot U_1 - [k_2] \cdot V_1$
$C_2 \leftarrow [k_1] \cdot U_2 - [k_2] \cdot V_2$
$C_3 \leftarrow [a] \cdot U_1 + [b] \cdot Y$
❶ $\mathcal{V} \xrightarrow{r} \mathcal{P}$
$r \overset{R}{\longleftarrow} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{a, b, c, d} \mathcal{V}$
$c \leftarrow k_1 - xk_0(r+b) \mod q$
$d \leftarrow k_2 - k_0(r+b) \mod q$

• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow [c] \cdot U_1 - [d] \cdot V_1$
$\widetilde{C_2} \leftarrow C_0 + [c] \cdot U_2 - [r+b] \cdot V_2$
$\widetilde{C_3} \leftarrow [a] \cdot U_1 + [b] \cdot Y$
if $(C_1, C_2, C_3)(\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3}) \wedge C_0 \neq \mathbb{O}_{\mathbb{G}_2}$
    then $b \leftarrow$ Accept else $b \leftarrow \perp$
```

```
Protocole IDL.Fake
Common input: $(U_1, U_2, V_1, V_2)$, $Y$
$\mathcal{P}$'s input: $y$
$\mathcal{V}$'s output: $b$
① $\mathcal{P} \xrightarrow{C_0, C_1, C_2, C_3} \mathcal{V}$
$(c, d, k_1, k_2) \overset{R}{\longleftarrow} [\![1, q-1]\!]^4$
$C_0 \overset{R}{\longleftarrow} \mathbb{G} \setminus \{\mathbb{O}_{\mathbb{G}}\}$ ; $C_1 \leftarrow [c] \cdot U_1 - [d] \cdot V_1$
$C_2 \leftarrow C_0 + [c] \cdot U_2 - [k_1] \cdot V_2$
$C_3 \leftarrow [k_2] \cdot U_1$
❶ $\mathcal{V} \xrightarrow{r} \mathcal{P}$
$r \overset{R}{\longleftarrow} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{a, b, c, d} \mathcal{V}$
$b \leftarrow k_1 - r \mod q$ ; $a \leftarrow b - k_2 y \mod q$

• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow [c] \cdot U_1 - [d] \cdot V_1$
$\widetilde{C_2} \leftarrow C_0 + [c] \cdot U_2 - [r+b] \cdot V_2$
$\widetilde{C_3} \leftarrow [a] \cdot U_1 + [b] \cdot Y$
if $(C_1, C_2, C_3)(\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3}) \wedge C_0 \neq \mathbb{O}_{\mathbb{G}_2}$
    then $b \leftarrow$ Accept else $b \leftarrow \perp$
```

Fig. 2. Interactive designated verifier proof of membership to the language $\mathsf{IDL}(\mathbb{G})$

- Σ.Sign: on message m and period p, the signer does the following:
  - $r \overset{R}{\longleftarrow} [\![1, q-1]\!]$, $R \leftarrow rP$. If $F(R) = 0$ it tries with another value r.
  - $e_p \leftarrow H_v^1(p)$, $d \leftarrow H_{e_p}^2(m, R)$, $T \leftarrow dP$
  - $s \leftarrow (F(T) \cdot d \cdot h(m) \cdot v - u \cdot F(R) - 1)r^{-1} \bmod q$

  The signature is the tuple (R, T, s).
- Σ.Cont: to control the validity of a signature (R, T, s), the signer checks that: $(v \cdot F(T) \cdot h(m)) \cdot T = F(R) \cdot U + s \cdot R + P$ using its private key v.
- Σ.{Conf/Deny}: the signer provides a designated verifier proof of the equality/inequality of two discrete logarithms, namely, $F(R) \cdot U + s \cdot R + P$ to the base $(F(T).h(m)) \cdot T$ and V to the base P (see § 4.2).
- Σ.Conv: there exist two types of conversions, namely
  - The gradual conversion for the signature corresponding to the time period p could be done by releasing the value e_p.
  - The individual conversion can be achieved by releasing the value of d.
- Σ.Vf: The signature corresponding to the period p, once e_p or d is revealed, could be checked by any verifier using the equations: $(d \cdot F(T) \cdot h(m))V = F(R) \cdot U + s \cdot R + P$ and $T = dP$.

4.2 Proofs of Equality/Inequality of Discrete Logarithms

Let $\mathbb{G}$ be a group. To confirm or deny that a bit string is a signature in our undeniable signature scheme, it is necessary to prove that a given quadruple $(U_1, V_1, U_2, V_2) \in \mathbb{G}^4$ is a Diffie-Hellman quadruple (or not), *i.e.* belongs to the set $\mathsf{EDL}(\mathbb{G}) = \{(U_1, V_1, U_2, V_2) \in \mathbb{G}^4, \log_{U_1}(V_1) = \log U_2(V_2)\}$, (or to the set $\mathsf{IDL}(\mathbb{G}) = \mathbb{G}^4 \setminus \mathsf{EDL}(\mathbb{G})$).

To face *blackmailing* or *mafia* attacks against our undeniable signatures, we use interactive designated verifier proofs, as introduced in [11] by Jakobsson, Sako, and Impagliazzo, in Chaum's proofs of equality (*cf.* Fig. 1) and inequality (*cf.* Fig. 2) of discrete logarithm of [6]. The idea is to replace the generic commitment scheme by a *trapdoor commitment* [11] and using classical techniques, the proofs are readily seen to be complete, sound, and above all non-transferable. The protocols, involve a point $Y = yU_1$ where y is the secret key of the verifier, and the prover must be convinced that Y is well-formed (in the registered public key model, the registration procedure is used to force the users to know the secret-key corresponding to their public key).

5 Security Analysis

We note first that the property of non-transferability is fulfilled by our scheme as a direct consequence of the use of designated-verifier proofs in the confirm/deny protocols. Further, we state that our scheme resists existential forgeries and that signatures are anonymous. Both security reductions stand in the generic group model [19].

5.1 Resistance to Forgery

The theorem below states that our variant of Michels-Petersen-Horster scheme is $\mathsf{EF\text{-}CMA}$-secure in the generic group model assuming the preimage resistance, the random affine preimage resistance and the random linear collision resistance of the underlying hash function family.

Theorem 1. *Let $\mathcal{A}$ be an $\mathsf{EF\text{-}CMA}$-adversary in the generic group model, operating in time t, after n group queries and m signing queries, such that $m \ll n^2$ and $n \gg 1$, with success probability $\mathbf{Succ}^{\mathsf{ef\text{-}cma}}_{\Sigma,\mathcal{A}}$.*

There exist adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ operating in time t' against the n-random affine preimage resistance, the n-random linear collision resistance and the preimage resistance of the underlying hash function (respectively) such that:

$$t' \leq t + 5n\tau_G \ln n + 5m \ln n(2\tau_G + \tau_{H^1} + \tau_{H^2} + \tau_F + \tau_h)$$

and

$$6 \cdot \mathbf{Succ}^{raPre(n)}_{h,\mathcal{B}} + 2 \cdot \mathbf{Succ}^{rlColl(n)}_{h,\mathcal{C}} + 3 \cdot n^2 \mathbf{Succ}^{Pre(n)}_{h,\mathcal{D}} \geq \frac{\mathbf{Succ}^{\mathsf{ef\text{-}cma}}_{\Sigma,\mathcal{A}}}{8} - 5n^4/q - 3mn^3$$

where δ is the advantage of an adversary playing a distinguisher for G, τ_g, τ_F, τ_{H^1}, τ_{H^2} and τ_h are the running time for G, F, H^1, H^2 and h respectively.

The $\mathsf{EF\text{-}CMA}$-adversary $\mathcal{A}$ will output a valid signature $\sigma^\star = (R^\star, T^\star, s^\star)$ on a message $m^\star$ for the time period $p^\star$ with success probability $\mathbf{Succ}^{\mathsf{ef\text{-}cma}}_{\Sigma,\mathcal{A}}$. In our security analysis, this event is divided into subevents according to whether $R^\star$ and $T^\star$ are created during the simulation by a signature query or a group query.

In the list used to maintain the group oracle, a group element created during a group query will have a "group" tag, while the tag "sign" will correspond to elements created in a signature query. Moreover, a signature query on a message m_i for the time period p_i will be answered by a triple (R_i, T_i, s_i), where $R_i, T_i \in \mathbb{G}$. Hence, in addition we will specify the type of an element that has the tag sign: we denote $\mathsf{Type}(R_i) = 0$ and $\mathsf{Type}(T_i) = 1$. The different forgeries output by $\mathcal{A}$ will be classified as follows:

– **Type 0:** $\mathsf{Tag}(R^\star) = \mathsf{group}$, $\mathsf{Tag}(T^\star) = \mathsf{group}$
– **Type 1:** $\mathsf{Tag}(R^\star) = \mathsf{group}$, $\mathsf{Tag}(T^\star) = \mathsf{sign}$ and $\mathsf{Type}(T^\star) = 0$
– **Type 2:** $\mathsf{Tag}(R^\star) = \mathsf{group}$, $\mathsf{Tag}(T^\star) = \mathsf{sign}$ and $\mathsf{Type}(T^\star) = 1$
– **Type 3:** $\mathsf{Tag}(R^\star) = \mathsf{sign}$, $\mathsf{Tag}(T^\star) = \mathsf{group}$ and $\mathsf{Type}(R^\star) = 0$
– **Type 4:** $\mathsf{Tag}(R^\star) = \mathsf{sign}$, $\mathsf{Tag}(T^\star) = \mathsf{group}$ and $\mathsf{Type}(R^\star) = 1$
– **Type 5:** $\mathsf{Tag}(R^\star) = \mathsf{sign}$, $\mathsf{Tag}(T^\star) = \mathsf{sign}$, $\mathsf{Type}(R^\star) = 0$ and $\mathsf{Type}(T^\star) = 0$
– **Type 6:** $\mathsf{Tag}(R^\star) = \mathsf{sign}$, $\mathsf{Tag}(T^\star) = \mathsf{sign}$, $\mathsf{Type}(R^\star) = 0$ and $\mathsf{Type}(T^\star) = 1$
– **Type 7:** $\mathsf{Tag}(R^\star) = \mathsf{sign}$, $\mathsf{Tag}(T^\star) = \mathsf{sign}$, $\mathsf{Type}(R^\star) = 1$ and $\mathsf{Type}(T^\star) = 0$
– **Type 8:** $\mathsf{Tag}(R^\star) = \mathsf{sign}$, $\mathsf{Tag}(T^\star) = \mathsf{sign}$, $\mathsf{Type}(R^\star) = 1$ and $\mathsf{Type}(T^\star) = 1$

We denote ε_i the probability that the forgery $\sigma^\star = (R^\star, T^\star, s^\star)$ output by $\mathcal{A}$ is of type **Type i** (for $i \in \{1, \ldots, 8\}$). We have:

$$\sum_{i=1}^{8} \varepsilon_i = \mathbf{Succ}^{\mathsf{ef\text{-}cma}}_{\Sigma,\mathcal{A}}$$

The adversaries, $\mathcal{B}, \mathcal{C}$ and $\mathcal{D}$ will simulate the group and signing oracles according to the alleged kind of forgery returned by $\mathcal{A}$. More precisely, adversary $\mathcal{C}$ will use the forgery to find a random linear collision if it is of type **Type 6**, $\mathcal{D}$ will exploit a forgery of the type **Type 0** to break the preimage resistance and finally, the adversary $\mathcal{B}$ will utilize all the remaining cases to find a random affine preimage.

The group oracle. In this model [19], one assumes that group operations can be perfomed only by means of an oracle. More specifically, suppose that $\mathbb{G}$ is an (additive) group of prime order q generated by P. Then $\mathbb{G}$ is isomorphic to the additive group $\mathbb{Z}_q$ and in the generic model, one assumes that instead of having explicit formulas for the group element iP, the adversary has only access to an "encoding" $\sigma(i) \in S \subset \{0,1\}^*$, that represents the element iP. The generic algorithm $\mathcal{A}$ will then consult the oracle for two types of queries:

– $\mathcal{A}$ requests the encoding of i: the oracle will select randomly a value $\sigma(i)$, to represent the element iP, from the given set of bit-srings.
– Given two encodings $\sigma(i)$ and $\sigma(j)$, $\mathcal{A}$ requests (without knowing necessarily i and j) the encoding of $(\sigma(i \pm j)$. Again the oracle responds with a randomly chosen bit-string.

The only condition on the oracle responses is that if the same group element is queried a second time, the same corresponding encoding must be returned. We will group the above queries in a single type of query, namely, $(\overrightarrow{i}, \overrightarrow{\alpha})$ where $\overrightarrow{i}$ refers to the set of indices of the group elements whereas $\overrightarrow{\alpha}$ denotes the set of exponents. The answers to such queries are elements z_i of $S \subset \{0.1\}^*$. Let $\mathcal{L} = \{z_1, z_2, z_3, \ldots, z_{n+2}\}$ be the sequence of queries' answers where n denotes the total number of queries to the group oracle. We use an interpretation similar to the one in [21], using polynomials $F_i(X)$ over $\mathbb{F}_q$:

- Polynomials F_1 and F_2 are set to $F_1 = 1$ and $F_2 = X$, which correspond to the generator and the public key U respectively. The corresponding bit-strings are z_1 and z_2 respectively.
- At the ℓ-th query $(\overrightarrow{i}, \overrightarrow{\alpha})$, the polynomial F_ℓ is defined as $\sum_{j=1}^{|\overrightarrow{\alpha}|} \alpha_j \mathcal{F}_{\overrightarrow{i}_j}$. If F_ℓ is already listed[5] as F_h, then F_ℓ is marked and the corresponding answer to F_h is returned. Otherwise, z_ℓ is selected at random from S, recorded[6] using $\mathtt{Record}(z_\ell \| F_\ell \| \mathsf{group} \| \mathsf{notype})$ and then returned to $\mathcal{A}$.

It is easy to see that the simulation driven by this interpretation is similar to the one of the regular algorithm provided that all the answers corresponding to unmarked polynomials are distinct and no polynomial F_ℓ vanishes at $X = u$. In these conditions, we call the sequence of encodings a safe sequence. The probability of such a sequence is given by the following lemma [21]:

Lemma 1. *Assume $n^2 \leq q$. The probability of unsafe sequence is upper-bounded by $5(n+1)^2/q$.*

The signing oracle. Basically, the signing oracle Σ will receive queries, of the form (m, p) and will respond with a valid signature $\sigma = (R, T, s)$ according to the following simulation:

Simulation of Σ: on query (m, p) do the following:

```
– $R \xleftarrow{R} S$, $e_p \leftarrow H_v^1(p)$, $d \leftarrow H_{e_p}^2(m, R)$,
– Repeat: $a, b \xleftarrow{R} \mathbb{Z}_q$, $t \leftarrow (a - b \cdot F(R))a^{-1}d^{-1}v^{-1}h(m)^{-1} \bmod q$
  Until $T = g(t) \neq \perp$,
– Record$(R \| aX + b \| \mathsf{sign} \| 0)$, Record$(T \| d \| \mathsf{sign} \| 1)$,
– $s \leftarrow (d \cdot v \cdot t \cdot h(m) - 1)b^{-1} \bmod q$,
– Return $(R, T, s)$.
```

The confirming/denying oracles. The use of designated verifier proofs of membership and of the registered public key model makes these oracles useless for the attacker. Therefore, we do not describe them in our security proof.

[5] The adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ will maintain, in addition to the outputs' list $\mathcal{L}$, three further lists, namely, the list of corresponding polynomials, denoted $\mathcal{F}$, the list of tags $\mathcal{T}$ and the list of types $\mathcal{S}$.

[6] The command $\mathtt{Record}(R \| F \| t \| s)$ will abort in some cases, namely when $(R, F', ?, ?)$ already exists and $F' \neq F$. The probability that this event happens can be upper-bounded by n/q, where n is the number of queries to the group oracle.

Proof. Let $(R^\star, T^\star, s^\star)$ be the forgery output by $\mathcal{A}$ on the message $m^\star$ for the time period $p^\star$. Due to space limitations, we will detail only the reduction in the case where this forgery is of type **Type 0**, **Type 2** and **Type 6**.

Description of $\mathcal{B}$. $\mathcal{B}$ picks uniformly at random an integer $i \in \{1, 2, 3, 4, 5, 7, 8\}$ which is its guess for the type of the forgery output by $\mathcal{A}$. In the following simulation, we suppose that $\mathcal{A}$ returns a forgery of type **Type 2** and that $i = 2$ (the other cases are treated similarly).

The forgery produced by A satisfies the following equation[7]

$$a - b \cdot F(R^\star) = (ad - bc) \cdot v \cdot F(T^\star) \cdot h(m),$$

where $R^\star = aU + bP$ and $T^\star = cU + dP$. Since $T^\star$ was generated during a signature query as a "T" ($\mathsf{Type}(T) = 1$) we have $c = 0$ (the adversary must know the discrete logarithm of T in base P in case the attacker asks for the signature conversion). Hence, the equation turns out to be $a - b \cdot F(R) = a \cdot d \cdot v \cdot F(T) \cdot h(m)$ or

$$1 - \frac{a}{b} F(R) = d \cdot v \cdot v \cdot F(T) \cdot h(m).$$

Thus, in order to find a random affine preimage, $\mathcal{B}$ must plug the values α_i and β_j in answers to the group and the signature queries (respectively). More precisely, he must answer group queries (a, b) by R such that $1 - aF(R)/b = \alpha$, similarly, signature queries must be answered by (R, T, s), such that $-d \cdot v \cdot v \cdot F(T) = \beta$:

Game 0: We consider an $\mathsf{EF\text{-}CMA}$-adversary $\mathcal{A}$ in the generic group model. In any game **Game i**, we denote S_i the event "$(R^\star, T^\star, s^\star)$ is a valid forgery of type **Type 2** and $i = 2$". By definition, we have $\Pr[S_0] = \varepsilon_2/7$.

Game 1: We use the interpretation described above for the generic oracle which considers a safe sequence $\mathcal{L}$. This event' s probability $\Pr[S_1]$ satisfies

$$|\Pr[S_1] - \Pr[S_0]| \leq 5(n+1)^2/q.$$

Game 2: In this game we modify the simulation of the group oracle. On query (a, b) such that the corresponding polynomial $F = aX + b$ is new, $\mathcal{B}$ does the following:

```
• Repeat
      pick $\alpha$ from the corresponding oracle
      compute $r \leftarrow (1 - \alpha)ab^{-1}$
      compute $\tilde{R} \leftarrow g(r)$
• Until $\tilde{R} \neq$ Fail.
• Return $\tilde{R}$.
```

However, $\mathcal{B}$ stops after $5 \ln n$ trials. The event S_2 differs from the previous one if $\tilde{R}$ remains undefined. Since the experiments are mutually independent (a and b are uniformly distributed), we may use

[7] This follows from the verification equation $(v \cdot F(T^\star) \cdot h(m))T^\star = F(R^\star) \cdot U + s^\star \cdot R^\star + P$.

a lemma from elementary probability theory [21, Lemma 5] to bound the corresponding probability by $1/n^2$. The overall probability when l ranges the set of queries indices is then $1/n$. Hence, we have

$$\Pr[S_2] \geq (1 - 1/n)\Pr[S_1].$$

Game 3: In this simulation, the groupe oracle replaces $\tilde{R}$ from the previous game by R a new random encoding. It executes Record$(R\|aX + b\|\mathsf{group})$ and return the value R as the response to the oracle query. Since the inputs to G are uniformly distributed (α is picked at random), we can use n times the almost-invertibility of F (the so-called *Hybrid Technique*) to bound the probability of S_3:

$$|\Pr[S_3] - \Pr[S_2]| \leq n\delta_G.$$

Game 4: In this game, $\mathcal{B}$ changes the simulation of the signing oracle. On query (m, p) it does the following:

```
• Compute $e_p \leftarrow H_v^1(p)$
• Pick the next $\alpha$ in the instance of the random affine preimage problem
• Pick $a \in \mathbb{Z}_q^*$,
• Compute $b \leftarrow a(\alpha h(m) - 1)F(R)^{-1}$
• Record$(R\|aX + b\|\mathsf{sign}\|0)$
• Compute $d \leftarrow H_{e_p}^2(m, R)$.
• Repeat
      Pick the next $\beta$ in the instance of the random affine preimage problem
      Compute $t \leftarrow -d^{-1}v^{-1}\beta$
  Until $\tilde{T} = G(t) \neq \text{Fail}$.
• Compute $s \leftarrow -F(R)a^{-1}$.
• Return $(R, \tilde{T}, s)$
```

Here again, $\mathcal{B}$ aborts after $5\ln n$ trials. Using the hybrid technique as above, we have

$$\Pr[S_4] \geq (1 - m/n^2)\Pr[S_{3,}].$$

Game 5: In this game, $\mathcal{B}$ replaces $\tilde{T}$ by T and executes Record$(T\|d\|\mathsf{sign}\|1)$. Applying the same technique, we get

$$|\Pr[S_5] - \Pr[S_4]| \leq m\delta_G + mn/q.$$

Game 6: In this game, $\mathcal{B}$ exploits the forgery $(R^\star, T^\star, s^\star)$ returned by $\mathcal{A}$. Since $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{group}, \mathsf{sign})$ and $\mathsf{Type}(R^\star, T^\star) = (0, 1)$ and $\mathcal{B}$ generated the correct i, there exist i, j such that $R^\star = R_i, T^\star = T_j$ and $1 - \frac{a_i}{b_i}F(R_i) = \alpha_i$ and $-d_j \cdot v \cdot F(T_j) = \beta_j$, the equation satisfied by the forgery turns out to be $\alpha_i + \beta_j h(m) = 0$. $\mathcal{B}$ would then find a random affine preimage with success probability greater than

$$\epsilon_2/7 + 5n^2/q - n\delta - m\delta - 2mn/q$$

and time

$$t' \leq t + 5n\ln n + m(\tau_{H^1} + \tau_{H^2} + 5\tau_g \ln n + \tau_h + 2\tau_F).$$

Description of $\mathcal{C}$. $\mathcal{C}$ will simulate Γ and Σ such that the simulation exploits a forgery $(R^\star, T^\star, s^\star)$ of the type **Type 6**. Hence $\mathcal{C}$ will simulate Γ in the standard way described in 5.1. Furthermore, he will have to plug the λ's in answers to signature queries in a way that the returned signature $(R^\star, T^\star, s^\star)$ satisfies $1 - \frac{b}{a}F(R^\star) = \lambda$. More precisely, on (m, p), $\mathcal{C}$ does the following:

```
– Pick the next $\lambda$ in the instance of the random affine preimage problem -
– Compute $e_p = H^1_v(p)$
– Repeat
      Pick $\alpha \in_R \mathbb{Z}_q$,
      Compute $r \leftarrow \frac{\lambda-1}{\alpha}$
  Until $R = g(r) \neq Fail$
– Compute $d = H^2_{e_p}(m, R)$
– Repeat
      Pick $a \in_R \mathbb{Z}_q$, Compute $b = \alpha a$ and $t = (a - bF(R))(a \cdot d \cdot v \cdot h(m))^{-1}$
  Until $T = g(t) \neq Fail$
– Record $(R||aX + b||sign, 0)$ - Record $(T||d||sign||1)$
– Compute $s = (d \cdot v \cdot h(m) \cdot F(T) - 1) \cdot b^{-1}$
– Return $(R, T, s)$.
```

It is easy to conclude that this simulation, together with the above forgery returned by the attacker will lead to a find a random linear collision.

Description of $\mathcal{D}$. $\mathcal{D}$ will attempt to exploit a forgery $(R^\star, T^\star, s^\star)$ such that $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{group}, \mathsf{group})$ (**Type 0**) to find a preimage of a certain value, say a. The equation satisfied by the forgery is $a_i - b_iF(R_i) = (a_ib_j - a_jb_i)F(R_j) \cdot v \cdot h(m)$. For this, $\mathcal{D}$ will simulate the signing oracle in the standard way given in 5.1. To simulate Γ, $\mathcal{D}$ selects in advance $i, j \in_R [\![1, n]\!]$. If $i < j$, then on the i-th query (a_i, b_i), $\mathcal{D}$ will select $R_i \in_R S$ and record it using $\mathtt{Record}(R_i \| a_iX + b_i \| \mathsf{group})$. On the j–th query (a_j, b_j), compute $T_j \leftarrow g(a \cdot (a_i - b_iF(R))(a_ib_j - a_jb_i)^{-1}v^{-1})$. With probability at least $1/n^2$, $\mathcal{D}$ would have chosen the correct i, j and the success of having $T_j \neq \perp$ is at least $1/3$ (almost invertibility of F and randomness of a). If $j \leq i$, $\mathcal{D}$ will proceed in a similar manner. □

5.2 Anonymity

Theorem 2. *Let $\mathcal{A}$ be an* Ano-CMA*-adversary operating in time t, after n group queries and m signing queries, with success advantage ε, such that $m \ll n^2$, $m \ll q$ and $n \gg 2$, then there exist adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{C}$, operating in time t' and attempting to break the pseudo-randomness property of H^1, the pseudo-randomness of H^2 and the random linear collision of h (respectively) with success probability $\mathbf{Succ}^{prf}_{H^1,\mathcal{B}_1}$, $\mathbf{Succ}^{prf}_{H^2,\mathcal{B}_2}$ and $\mathbf{Succ}^{rlColl(n)}_{h,\mathcal{C}}$ such that:*

$$t' \leq t + 5n\tau_g \ln n + 5m \ln n(\tau_{H^2} + \tau_h + \tau_g) + m\tau_{H^1}$$

and

$$\mathbf{Succ}^{prf}_{H^1,\mathcal{B}_1} + \mathbf{Succ}^{prf}_{H^2,\mathcal{B}_2} + 2\frac{\mathbf{Succ}^{rlColl(n)}_{h,\mathcal{C}}}{n} \geq \frac{\varepsilon}{n} + \frac{18n}{q} - n\delta + \delta + \frac{3m\delta}{n}$$

where δ is the advantage of an adversary playing a distinguisher for g, τ_g, τ_F, τ_{H^1}, τ_{H^2} and τ_h are the running time for g, F, H^1, H^2 and h respectively.

Proof. The proof is similar to the previous one and will be given in the full version of the paper. □

6 Conclusion

We properly defined security notions for convertible undeniable signatures that support the additional property of *achronous* gradual conversion. Adapting the scheme proposed by Michels, Petersen and Horster in 1996, we realized the first scheme featuring this usefull notion of conversion. In addition, we gave the first security analysis of the Michels-Petersen-Horster protocol, thereby addressing a problem left open since 1996. We have modified this scheme such that it becomes a generic one, which allows to use it for instance in the setting of elliptic curves (and therefore offers attractive practical advantages in terms of signature length and performances). In this context, in comparison with the time-selective convertible undeniable signatures from [13], the computational costs for the confirmation/disavowal protocols and the conversion algorithms, are much smaller.

References

1. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, *Key-Privacy in Public-Key Encryption.*, Advances in Cryptology - Asiacrypt 2001 (C. Boyd, ed.), Lect. Notes Comput. Sci., vol. 2248, Springer, 2001, pp. 566–582.
2. M. Bellare and T. Kohno, *Hash Function Balance and its Impact on Birthday Attacks.*, Advances in Cryptology - Eurocrypt 2004 (C. Cachin and J. Camenisch, eds.), Lect. Notes Comput. Sci., vol. 3027, Springer, 2004, pp. 401–418.
3. M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.*, Proceedings of the First ACM Conference on Computer and Communications Security (D. Denning, R. Pyle, R. Ganesan, R. Sandhu, and V. Ashby, eds.), ACM Press, 1993, pp. 62–73.
4. J. Boyar, D. Chaum, I. B. Damgård, and T. B. Pedersen, *Convertible undeniable signatures.*, Advances in Cryptology - Crypto'90 (A. J. Menezes and S. A. Vanstone, eds.), Lect. Notes Comput. Sci., vol. 537, Springer, 1991, pp. 189–205.
5. D. R. L. Brown, *Generic Groups, Collision Resistance, and ECDSA.*, Des. Codes Cryptography **35** (2005), no. 1, 119–152.
6. J. Camenisch and V. Shoup, *Practical Verifiable Encryption and Decryption of Discrete Logarithms.*, Advances in Cryptology - Crypto 2003 (D. Boneh, ed.), Lect. Notes Comput. Sci., vol. 2729, Springer, 2003, pp. 126–144.
7. D. Chaum and H. van Antwerpen, *Undeniable Signatures.*, Advances in Cryptology - Crypto'89 (G. Brassard, ed.), Lect. Notes Comput. Sci., vol. 435, Springer, 1990, pp. 212–216.
8. A. W. Dent, *Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model.*, Advances in Cryptology - Asiacrypt 2002 (Y. Zheng, ed.), Lect. Notes Comput. Sci., vol. 2501, Springer, 2002, pp. 100–109.

9. S. D. Galbraith and W. Mao, *Invisibility and Anonymity of Undeniable and Confirmer Signatures.*, Topics in Cryptology - CT-RSA 2003 (M. Joye, ed.), Lect. Notes Comput. Sci., vol. 2612, Springer, 2003, pp. 80–97.
10. S. Goldwasser, S. Micali, and R. L. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.*, SIAM J. Comput. **17** (1988), no. 2, 281–308.
11. M. Jakobsson, K. Sako, and R. Impagliazzo, *Designated Verifier Proofs and Their Applications.*, Advances in Cryptology - Eurocrypt'96 (U. M. Maurer, ed.), Lect. Notes Comput. Sci., vol. 1070, Springer, 1996, pp. 143–154.
12. C. Kudla and K. G. Paterson, *Non-interactive Designated Verifier Proofs and Undeniable Signatures.*, Cryptography and Coding, 10th IMA International Conference (N. P. Smart, ed.), Lect. Notes Comput. Sci., vol. 3796, Springer, 2005, pp. 136–154.
13. F. Laguillaumie and D. Vergnaud, *Time-Selective Convertible Undeniable Signatures.*, Topics in Cryptology - CT-RSA 2005 (A. J. Menezes, ed.), Lect. Notes Comput. Sci., vol. 3376, Springer, 2005, pp. 154–171.
14. M. Michels, H. Petersen, and P. Horster, *Breaking and Repairing a Convertible Undeniable Signature Scheme.*, Proceedings of the Third ACM Conference on Computer and Communications Security (L. Gong and J. Stern, eds.), ACM Press, 1996, pp. 148–152.
15. W. Ogata, K. Kurosawa, and S.-H. Heng, *The Security of the FDH Variant of Chaum's Undeniable Signature Scheme*, IEEE Trans. Inf. Theory **52** (2006), no. 5, 2006 – 2017.
16. T. Okamoto and D. Pointcheval, *The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes.*, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001 (K. Kim, ed.), Lect. Notes Comput. Sci., vol. 1992, Springer, 2001, pp. 104–118.
17. P. Paillier and D. Vergnaud, *Discrete-Log Based Signatures May Not Be Equivalent to Discrete-Log.*, Advances in Cryptology - Asiacrypt 2005 (B. Roy, ed.), Lect. Notes Comput. Sci., vol. 3788, Springer, 2005, pp. 1–20.
18. P. Rogaway and T. Shrimpton, *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance.*, Fast Software Encryption, 11th International Workshop, FSE 2004 (B. K. Roy and W. Meier, eds.), Lect. Notes Comput. Sci., vol. 3017, Springer, 2004, pp. 371–388.
19. V. Shoup, *Lower Bounds for Discrete Logarithms and Related Problems.*, Advances in Cryptology - Eurocrypt'97 (W. Fumy, ed.), Lect. Notes Comput. Sci., vol. 1233, Springer, 1997, pp. 256–266.
20. W. Stadje, *The Residues modulo m of Products of Random Integers.*, Comb. Probab. Comput. **11** (2002), no. 5, 529–540.
21. J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart, *Flaws in applying proof methodologies to signature schemes.*, Advances in Cryptology - Crypto 2002 (M. Yung, ed.), Lect. Notes Comput. Sci., vol. 2442, Springer, 2002, pp. 93–110.

Author Index