Joonsang Baek
Feng Bao
Kefei Chen
Xuejia Lai (Eds.)

LNCS 5324

Provable Security

Second International Conference, ProvSec 2008
Shanghai, China, October/November 2008
Proceedings

Springer

Lecture Notes in Computer Science 5324

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Joonsang Baek Feng Bao Kefei Chen
Xuejia Lai (Eds.)

Provable Security

Second International Conference, ProvSec 2008
Shanghai, China, October 30 - November 1, 2008
Proceedings

Springer

Volume Editors

Joonsang Baek
Feng Bao
Institute for Infocomm Research
1 Fusionopolis Way, 21-01 Connexis
Singapore 138632, Singapore
E-mail: {jsbaek, baofeng}@i2r.a-star.edu.sg

Kefei Chen
Xuejia Lai
Shanghai Jiao Tong University
Department of Computer Science and Engineering
800 Dong Chuan Road
Shanghai, 200240, China
E-mail: {chen-kf, lai-xj}@cs.sjtu.edu.cn

Library of Congress Control Number: 2008937905

CR Subject Classification (1998): E.3, K.6.5, I.1, I.2.2

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-88732-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-88732-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12551750 06/3180 5 4 3 2 1 0

Preface

The second international conference on provable security, ProvSec 2008, was held in Shanghai, China, during October 30th – November 1st, 2008. The conference was sponsored by Shanghai Jiao Tong University (SJTU) in cooperation with the Chinese Association for Cryptologic Research (CACR) and the Natural Science Foundation of China (NSFC).

The aim of ProvSec is to provide a platform for researchers, scholars, and practitioners to exchange ideas and extend knowledge on provable security, which is an important research area in cryptography. The first ProvSec was held in Wollongong, Australia, in 2007.

This year, the conference received 79 papers and the program committee selected 25 papers during eight weeks' thorough reviewing process. The authors of the selected papers are from 12 different countries: Australia, Belgium, China, Estonia, France, Germany, India, Japan, Norway, Singapore, the UK, and the USA. We are grateful to the members of the program committee for their many hours of valuable time and hard work.

In addition to the regular conference program, the conference hosted two invited talks:

- Kenny Paterson (University of London, Royal Holloway): Non-interactive Key Distribution and Identity-Based Encryption: A Historical Perspective
- Phillip Rogaway (University of California, Davis): Blockcipher Modes of Operation: Culture and Counter-Culture in Modern Cryptography.

The conference was also one of the special events for the 50th anniversary of the Department of Computer Science at SJTU.

We extend our gratitude to all the people involved in organizing ProvSec from the Department of Computer Science and Engineering and the Lab for Information Security of Shanghai Jiao Tong University, in particular to Yu Long, Yanfei Zheng, Meiju Chen, Zhihua Su, and Bo Zhu for their great efforts in making the conference run smoothly.

Last but not least, we thank all the authors for submitting their interesting papers to ProvSec 2008, many of which were of good quality but only a few of which could be accepted due to lack of space in the program.

October 2008

Joonsang Baek
Feng Bao
Kefei Chen
Xuejia Lai

Second International Conference on Provable Security 2008 (ProvSec 2008)

General Chair

Xuejia Lai	Shanghai Jiao Tong University, China

Program Chairs

Feng Bao	Institute for Infocomm Research, Singapore
Kefei Chen	Shanghai Jiao Tong University, China

Publicity Chair

Joonsang Baek	Institute for Infocomm Research, Singapore

Program Committee

Joonsang Baek	Institute for Infocomm Research, Singapore
Tom Berson	Anagram Laboratories, USA
Lily Chen	NIST, USA
Liqun Chen	Hewlett-Packard Laboratories, UK
Robert Deng	Singapore Management University, Singapore
Dengguo Feng	Chinese Academy of Sciences, China
David Galindo	University of Malaga, Spain
Akinori Kawachi	Tokyo Institute of Technology, Japan
Aggelos Kiayias	University of Connecticut, USA
Kwangjo Kim	ICU, Korea
Miroslaw Kutylowski	Wroclaw University of Technology, Poland
Fabien Laguillaumie	University of Caen, France
Ninghui Li	Purdue University, USA
Helger Lipmaa	University College London, UK
Shengli Liu	Shanghai Jiao Tong University, China
Javier Lopez	University of Malaga, Spain
Yi Mu	University of Wollongong, Australia
David Naccache	ENS, France
Antonio Nicolosi	Stevens Institute of Technology, USA
Juanma Gonzalez Nieto	Queensland University of Technology, Australia
Tatsuaki Okamoto	NTT Labs, Japan
Kenny Paterson	Royal Holloway, UK
Olivier Pereira	UCL, Belgium
Giuseppe Persiano	Università di Salerno, Italy

Raphael C.W. Phan	EPFL, Switzerland
Kui Ren	Illinois Institute of Technology, USA
Kouichi Sakurai	Kyushu University, Japan
Alice Silverberg	U.C. Irvine, USA
Ron Steinfeld	Macquarie University, Australia
Willy Susilo	University of Wollongong, Australia
Damien Vergnaud	ENS, France
Jorge Villar	Universitat Politecnica de Catalunya, Spain
Huaxiong Wang	Nanyang Technological University, Singapore
Duncan Wong	City University of Hong Kong, China
Tzong-Chen Wu	NTUST, Taiwan
Shouhuai Xu	University of Texas at San Antonio, USA
Fangguo Zhang	Sun Yat-sen University, China
Rui Zhang	AIST, Japan
Yunlei Zhao	Fudan University, China
Huafei Zhu	Institute for Infocomm Research, Singapore

External Referees

Man Ho Au	Qiong Huang	Ray Perlner
Przemysław Błaśkiewicz	Xinyi Huang	Yi Qian
Colin Boyd	Emeline Hufschmitt	Ying Qui
Shaoying Cai	Takashi Kitagawa	Peter van Rossum
Debrup Chakraborty	Marek Klonowski	Chunhua Su
Kim-Kwang Raymond Choo	Divyan M. Konidala	Hirotoshi Takebe
	Michał Koza	Xiao Tan
Dang Nguyen Duc	Hyunrok Lee	Qian Wang
Nelly Fazio	David Lefranc	Puwen Wei
Zheng Gong	Jin Li	Jian Weng
Choudary Gorantla	Vo Duc Liem	Paweł Wlaź
Jens Groth	Yu Long	Wei Wu
Fuchun Guo	Olivier de Marneffe	Guomin Yang
Hua Guo	Giacomo de Meulenaer	Yanjiang Yang
Ryotaro Hayashi	Paul Morrissey	Tsz Hon Yuen
Takato Hirano	Mridul Nandi	Yao Zai
Dennis Hofheinz	Kazuto Ogawa	Xingwen Zhao
Xuan Hong	Ayoub Otmani	Hong-Sheng Zhou

Organizing Committee

Zhihua Su	Shanghai Jiao Tong University, China
Meiju Chen	Shanghai Jiao Tong University, China
Yanfei Zheng	Shanghai Jiao Tong University, China
Yu Long	Shanghai Jiao Tong University, China
Bo Zhu	Shanghai Jiao Tong University, China

Table of Contents

Encryption

Signature

Analysis

Application of Hash Functions

Universal Composability

Applications

Generalized ElGamal Public Key Cryptosystem Based on a New Diffie-Hellman Problem*

Huawei Huang[1,2], Bo Yang[1], Shenglin Zhu[1], and Guozhen Xiao[2]

[1] College of Information, South China Agricultural University, Guangzhou 510642, China
[2] State Key Lab. of Integrated Service Networks, Xidian University, Xi'an, 710071, China
hwhuang7809@gmail.com, byang@scau.edu.cn
zhusl@scau.edu.cn, gzxiao@xidian.edu.cn

Abstract. This paper proposes a new generalized ElGamal public key encryption scheme based on a new Diffie-Hellman problem, so-called EDDH problem, which DDH problem can be reduced to. This scheme is one-way if and only if ECDH assumption holds and it is semantically secure in the standard model if and only if EDDH assumption holds. Since EDDH assumption still holds for generic bilinear groups, this encryption scheme adds to the growing toolkit of provable security primitives that can be used by the protocol designer looking to build complex secure systems with a sound basis.

Keywords and phrases: public key cryptosystem, matrix semigroup action, DDH problem, semantically secure.

1 Introduction

Diffie and Hellman introduced the concept of public key cryptography in their landmark paper of 1976 [4]. They showed how to construct a public key cryptosystem (PKC) using a trapdoor one-way function. In 1985, ElGamal [5] proposed the renowned ElGamal cryptosystem which has been widely used and studied for the last two decades.

The discrete logarithm (DL) problem is the basic ingredient of many cryptography protocols such as Diffie-Hellman key agreement protocol, ElGamal public key cryptosystem, digital signature algorithm (DSA) and ElGamal's signature scheme. The CDH assumption was introduced informally by [4]. Since then, there have been many papers that deal with the DL and CDH assumptions, and cryptographic applications based on them. The DDH assumption appears to have first surfaced in the cryptographic literature in [2], although as that paper notes, the DDH assumption is actually needed to prove the security of a number of previously proposed protocols. Indeed, the famous Diffie-Hellman key exchange cannot be proved secure in any reasonable and standard way just

* Project supported by the nature science foundation of China (No. 60573043; 60773175; 60773003).

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 1–21, 2008.

based on the CDH assumption: the DDH assumption (or some variant thereof) is required. And the well-known encryption scheme of ElGamal relies on the DDH for its security against passive attacks (i.e., semantic security).

Although Shoup [14] shows that the DDH problem is hard in a generic model of computation, Joux and Nguyen [8] proves that the DDH assumption is a much stronger hypothesis than the CDH assumption. Joux and Nguyen show that there are some very special families of elliptic curves for which the DDH assumption does not hold, but for which the CDH assumption still appears to stand. In fact, in all bilinear group for which there exists a computable pairing, DDH problem is solvable in polynomial time, hence in bilinear group DDH assumption does not hold. These facts cast some doubt on the DDH assumption in general. Many cryptographers propose other Diffie-Hellman problem. For example, Boneh and Boyen [1] proposed linear DDH assumption and proved that linear DDH assumption is weaker than DDH assumption in generic bilinear groups. Shacham [13] studied further the linear DDH assumption and presented a generalization of it into a family of progressively weaker assumptions.

In this paper we present a new generalized ElGamal PKC that can also be proven semantically secure in the standard model under a new decisional Diffie-Hellman problem which we call EDDH problem. The DDH problem can be reduced to EDDH problem and EDDH assumption is weaker than DDH assumption in generic bilinear groups. For generic groups in Shoup model, the lower bound on the computational complexity of EDDH problem is less than that of linear DDH problem.

This paper is structured as follows. Firstly, in Section 2, we cover some cryptographic preliminaries. Section 3 defines the complexity assumptions on which the security of the new PKC relies. Section 4 describes the new scheme. Finally, Section 5 analyses its security..

2 Preliminaries

Throughout this paper, we employ some terminologies and notions in [3].

Z denotes the ring of integers. $Z_{\geq 0}$ denotes the set of non-negative integers. For a prime q, F_q denotes the finite field of integers modulo q.

We write $\nu \leftarrow \alpha$ to denote the algorithmic action of assigning the value of α to the variable ν.

Let X be a finite probability space, i.e., a probability space on a finite set S. For $\alpha \in S$, we let $\Pr_X[\alpha]$ denote the probability that X assigns to α, and for $S' \subset s$, we let $\Pr_X[S']$ denote the probability that X assigns to S'.

We write $\nu \stackrel{R}{\leftarrow} X$ to denote the algorithmic action of sampling an element of S according to the distribution X, and assigning the result of this sampling experiment to the variable ν. We sometimes write $\nu_1, ..., \nu_k \stackrel{R}{\leftarrow} X$ as a shorthand for $\nu_1 \stackrel{R}{\leftarrow} X$;...;$\nu_k \stackrel{R}{\leftarrow} X$.

For any finite set S, $\mathbf{U}(S)$ denotes the uniform distribution on S. We write $\nu \stackrel{R}{\leftarrow} S$ as a shorthand for $\nu \stackrel{R}{\leftarrow} \mathbf{U}(S)$.

For any probability space X on a finite set S, we denote by $[X]$ the subset of elements of S that are assigned non-zero probability by X, i.e., the "support" of X.

If $X_1, X_2, ..., X_k$ are finite probability spaces, and ϕ is a k-ary predicate, then we write

$$\Pr[\phi(\nu_1, ..., \nu_k) : \nu_1 \stackrel{R}{\leftarrow} X_1; ...; \nu_k \stackrel{R}{\leftarrow} X_k]$$

to denote the probability that $\phi(\nu_1, ..., \nu_k)$ holds when ν_1 is sampled from X_1, ν_2 is sampled from X_2, etc.

For $\lambda \in Z_{\geq 0}$, 1^λ denotes the bit string consisting of λ copies of the bit 1. The string 1^λ will often be an input to an algorithm: this is a technical device that allows a polynomial-time algorithm to run in time bounded by a polynomial in λ, even if there are no other inputs to the algorithm, or those inputs happen to be very short.

A function F mapping non-negative integers to non-negative reals is called *negligible* if for all positive numbers c, there exists an integer $\lambda_0(c) \geq 0$ such that for all $\lambda > \lambda_0(c)$, we have $F(\lambda) < 1/\lambda^c$.

To understand some intractability assumptions of this paper in a general but precise way, we introduce the notion of a computational group scheme of [3].

Definition 2.1. *A computational group scheme $\mathcal{G}$ specifies a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of group distributions. For every value of a security parameter $\lambda \in Z_{\geq 0}$, S_λ is a probability distribution of group descriptions. A group description Γ specifies a finite abelian group $\hat{G}$, along with a prime-order subgroup G, a generator g of G, and the order q of G. We will write $\Gamma[\hat{G}; G; g; q]$ to indicate that Γ specifies $\hat{G}$, G, g, and q as above. As usual, mathematical objects like a group description Γ and elements of a group $\hat{G}$ are represented for computational purposes as bit strings bounded in length by a polynomial in λ.*

The group scheme should also provide several algorithms:

- *a deterministic, polynomial-time algorithm for computing the group operation that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [S_\lambda]$, along with $h_1, h_2 \in \hat{G}$, and outputs the group element $h_1 \cdot h_2 \in \hat{G}$;*
- *a deterministic, polynomial-time algorithm for computing the group inversion operation that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [S_\lambda]$, and $h \in \hat{G}$, and outputs $h^{-1} \in \hat{G}$;*
- *a deterministic, polynomial-time algorithm that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [S_\lambda]$, and $\alpha \in \{0,1\}^*$, and determines if α is a valid binary encoding of an element of $\hat{G}$;*
- *a deterministic, polynomial-time algorithm that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [S_\lambda]$, and $h \in \hat{G}$, and determines if $h \in G^n$;*
- *a deterministic, polynomial-time algorithm that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [S_\lambda]$, and outputs g, q and A;*
- *a probabilistic, polynomial-time approximate sampling algorithm $\hat{S}$ that on input 1^λ approximately samples S_λ. The distributions S_λ and $\hat{S}(1^\lambda)$ should be statistically close; that is, the statistical distance $\Delta(S_\lambda, \hat{S}(1^\lambda))$ should be a negligible function in λ.*

Let $\mathcal{G}$ be a computational group scheme, specifying a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of group distributions. For all $\lambda \in Z_{\geq 0}$, and for all $\Gamma[\hat{G}, G, g, q] \in [S_\lambda]$, we define the sets $\mathcal{P}_{\lambda,\Gamma}$ and $\mathcal{Q}_{\lambda,\Gamma}$ as follows:

$$\mathcal{P}_{\lambda,\Gamma} := \{(g^x, g^y, g^{xy}) \in G^3 | x, y \in F_q\};$$
$$\mathcal{Q}_{\lambda,\Gamma} := G^3.$$

For all 0/1-valued, probabilistic, polynomial-time algorithms $\mathscr{A}$, and for all $\lambda \in Z_{\geq 0}$, we define the *DDH advantage of $\mathscr{A}$ against $\mathcal{G}$ at λ* as

$$\begin{aligned}&\mathrm{AdvDDH}_{\mathcal{G},\mathscr{A}}(\lambda) := \\ &|\Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{P}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)] - \\ &\Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{Q}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)]|.\end{aligned}$$

The DDH assumption for $\mathcal{G}$ is this: *For every probabilistic, polynomial-time, 0/1-valued algorithm $\mathscr{A}$, the function $AdvDDH_{\mathcal{G},\mathscr{A}}(\lambda)$ is negligible in λ.*

For all 0/1-valued, probabilistic, polynomial-time algorithms $\mathscr{A}$, for all $\lambda \in Z_{\geq 0}$, and for all $\Gamma[\hat{G}, G, g, q] \in [S_\lambda]$, we define the *DDH advantage of $\mathscr{A}$ against $\mathcal{G}$ at λ given Γ* as

$$\begin{aligned}&\mathrm{AdvDDH}_{\mathcal{G},\mathscr{A}}(\lambda|\Gamma) := |\Pr[\tau = 1 : \rho \xleftarrow{R} \mathcal{P}_{\lambda\Gamma} \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)] - \\ &\Pr[\tau = 1 : \rho \xleftarrow{R} \mathcal{Q}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)]|.\end{aligned}$$

Definition 2.2. *An encryption scheme $(\mathscr{G}, \mathscr{E}, \mathscr{D})$ consists of the following three polynomial-time (in k) algorithms:*

1. *A key generation algorithm $\mathscr{G}$ that on input 1^λ outputs a pair of binary strings (e, d). The string e is called the encryption or public key, and d the corresponding decryption or secret key. $\mathscr{G}$ is probabilistic.*
2. *An encryption algorithm $\mathscr{E}$ that takes as inputs 1^λ, an encryption key e from $\mathscr{G}$, a message $m \in \{0,1\}^*$, and outputs a ciphertext c. $\mathscr{E}$ can be probabilistic.*
3. *A decryption algorithm $\mathscr{D}$ that takes as inputs 1^λ, a decryption key d and a ciphertext c, and outputs a binary string m', such that for every $(e, d) \leftarrow \mathscr{G}(1^\lambda)$,*

$$Pr[m' = \mathscr{D}(1^\lambda, d, \mathscr{E}(1^\lambda, e, m)) \neq m]$$

is negligible. $\mathscr{D}$ is deterministic.

For simplicity, in what follows we drop the term 1^λ from the input to the algorithms.

Definition 2.3. *An encryption scheme $(\mathscr{G}, \mathscr{E}, \mathscr{D})$ is said to be one-way if for all efficient algorithms $\mathscr{A}$, for every $\alpha > 0$ and sufficiently large λ,*

$$Pr[\mathscr{A}(e, c) = \mathscr{D}(d, c)] < \frac{1}{\lambda^\alpha},$$

where $c \leftarrow \mathscr{E}(e, m)$, $(e, d) \leftarrow \mathscr{G}(1^\lambda)$ and $m \leftarrow M$.

In 1984, Goldwasser and Micali [7] proposed a practical definition of security, semantic security, also known as polynomial indistinguishability.

Definition 2.4. *A public key cryptosystem $(\mathscr{G}, \mathscr{E}, \mathscr{D})$ is polynomial time indistinguishable if for every pair of efficient algorithms $\mathscr{A}, \mathscr{M}$, for every $\alpha > 0$ and sufficiently large λ,*

$$Pr[\mathscr{A}(e, m_a, m_b, c) = m] < \frac{1}{2} + \frac{1}{\lambda^\alpha},$$

where $(e, d) \leftarrow \mathscr{G}(1^\lambda)$, $\{m_a, m_b\} \leftarrow \mathscr{M}(1^\lambda)$, $m \leftarrow \{m_a, m_b\}$ and $c \leftarrow \mathscr{E}(e, m)$.

Fujisaki and Okamoto [6] described a generic transformation to achieve chosen-ciphertext security in the random oracle model from any semantically secure scheme.

3 Complexity Assumptions

3.1 A Class of Vector Space over Finite Field F_{q^2}

In general, the *companion matrix* of a monic polynomial $f(x) = a_0 + a_1 x + x^2$ of positive degree 2 over a field is defined to be the 2×2 matrix

$$A = \begin{pmatrix} 0 & -a_0 \\ 1 & -a_1 \end{pmatrix}.$$

It is well known in linear algebra that A satisfies the equation $f(A) = 0$; that is, $a_0 I + a_1 A + A^2 = 0$, where I is the 2×2 identity matrix.

Let F_p be a finite field with $p = q^2$, where q is the characteristic of F_p. If A is the companion matrix of a monic irreducible polynomial f over F_q of degree 2, then $f(A) = 0$, and therefore A can play the role of a root of f. The polynomials in A over F_q of degree less than 2 yield a representation of the elements of F_p, that is,

$$F_p \cong F_q[A] = \{x_0 I + x_1 A | x_0, x_1 \in F_q\}.$$

For more details, see [9].

Let $\hat{G}$ be a finite cyclic group and G be a q-order subgroup of $\hat{G}$ where q is a prime. Then $\hat{G}^2$ is also a finite group and G^2 is a subgroup of $\hat{G}^2$. For any $a = (a_1, a_2) \in \hat{G}^2$, $a^{-1} = (a_1^{-1}, a_2^{-1})$.

Let A be the companion matrix of a monic irreducible polynomial over finite field F_q of degree 2. Let $g = (g_1, g_2) \in G^n$ and $B = (b_{ij})_{2\times 2} \in F_q[A]$. We define an action of the finite field $F_q[A]$ on the group G^2 as follows

$$B * g = (\prod_{j=1}^{2} g_j^{b_{1j}}, \prod_{j=1}^{2} g_j^{b_{2j}}).$$

The operation "$*$" is an action of $F_q[A]$ on G^2 (see [10,11]). It is easy to verify that G^2 is a vector space over finite filed $F_q[A]$ with respect to the action.

Proposition 3.1. *Let A be the companion matrix of a monic irreducible polynomial over finite field F_q of degree n. Let G^2 be a vector space over finite field $F_q[A]$ as above. Then for any $b \in G^2$ and $g = (g_1, g_2) \in G^2$ where g_1, g_2 are generators of G, there exists a unique element $X \in F_q[A]$ such that $X * g = b$. That is, G^2 is a 1-dimensional vector space over finite field $F_q[A]$.*

Proof. Let $g = (g_1, g_2) \in G^2$ and g_1, g_2 are generators of G. Consider the map $\psi : (F_q[A], +) \to G^2; \psi(X) = X * g$.

Let $X, Y \in F_q[A]$. Then by the definition of vector space, $\psi(X + Y) = (X + Y) * g = (X * g)(Y * g) = \psi(X)\psi(Y)$.

Assume that $X * g = Y * g$ for some $X, Y \in F_q[A]$. By the definition of vector space, $(X - Y) * g = 1_{G^2}$. It follows that $X - Y = 0_{F_q[A]}$. (Assume that $X - Y \neq 0_{F_q[A]}$. Then $(X - Y)^{-1}[(X - Y) * g] = (X - Y)^{-1} * 1_{G^2} = 1_{G^2}$. On the other hand, $(X - Y)^{-1}[(X - Y) * g] = [(X - Y)^{-1}(X - Y)] * g = 1_{F_q[A]} * g = g$. So we deduce $g = (g_1, g_2) = 1_{G^2}$, which contradicts the condition that g_1, g_2 are generators of G.) Hence $X = Y$. Therefore, ψ is an injective homomorphism.

Since A is the companion matrix of a monic irreducible polynomial over finite field F_q of degree 2, we have $F_{q^2} \cong F_q[A]$. So $|F_q[A]| = |G^2| = q^2$. Hence $(F_q[A], +) \cong G^2$ and the proposition holds.

3.2 Generalized Computational Group Schemes

In this subsection, we propose the notion of a *generalized computational group scheme*. It is the natural generalization of computational group scheme in [3].

Definition 3.2. *A generalized computational group scheme $\mathcal{G}$ specifies a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of generalized group distributions. For every value of a security parameter $\lambda \in Z_{\geq 0}$, S_λ is a probability distribution of generalized group descriptions. A generalized group description Γ specifies*

(1) a finite abelian group $\hat{G}$, along with a prime-order subgroup G,
(2) a generator g of G^2 with $g = (g_1, g_2)$ such that g_1, g_2 are not equal each other,
(3) the order q of G,
(4) and the companion matrix A of a monic irreducible polynomial over finite field F_q of degree 2.

We will write $\Gamma[\hat{G}, G, g, q, A]$ to indicate that Γ specifies G, G, g, q and A as above. The generalized computational group scheme should also provide several algorithms:

- a deterministic, polynomial-time algorithm for computing the group operation that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$, along with $h_1, h_2 \in \hat{G}^2$, and outputs the group element $h_1 \cdot h_2 \in \hat{G}^2$;
- a deterministic, polynomial-time algorithm for computing the group inversion operation that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$, and $h \in \hat{G}^2$, and outputs $h^{-1} \in \hat{G}^2$;

- a deterministic, polynomial-time algorithm that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$, and $\alpha \in \{0,1\}^*$, and determines if α is a valid binary encoding of an element of $\hat{G}^2$;
- a deterministic, polynomial-time algorithm that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$, and $h \in \hat{G}^2$, and determines if $h \in G^2$;
- a deterministic, polynomial-time algorithm that takes as input 1^λ for $\lambda \in Z_{\geq 0}$, $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$, and outputs g, q and A;
- a probabilistic, polynomial-time approximate sampling algorithm $\hat{S}$ that on input 1^λ approximately samples S_λ. The distributions S_λ and $\hat{S}(1^\lambda)$ should be statistically close; that is, the statistical distance $\Delta(S_\lambda, \hat{S}(1^\lambda))$ should be a negligible function in λ.

It is clear that for a generalized computational group scheme $\mathcal{G}$ specifying a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of generalized group distributions Γ, there is a corresponding computational group scheme $\mathcal{G}'$ specifying a sequence $(S'_\lambda)_{\lambda \in Z_{\geq 0}}$ of group distributions Γ'. And any computational group scheme can be easily extended to a generalized computational group scheme. For example, we can obtain some generalized computational group schemes from the computational group schemes in [3].

3.3 ECDH Problem and ECDH Assumption

Definition 3.3. *(ECDH problem (ECDHP)). Let $\mathcal{G}$ be a generalized computational group scheme, specifying a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of generalized group distributions. Let $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$. Given $(X * g), (Y * g) \in G^2$ for some $X, Y \in F_q[A]$, find $(XY) * g$.*

For all probabilistic, polynomial-time algorithms $\mathscr{A}$, and for all $\lambda \in Z_{\geq 0}$, we define the *ECDH advantage of $\mathscr{A}$ against $\mathcal{G}$ at λ* as

$$\begin{aligned}\mathrm{AdvECDH}_{\mathcal{G},\mathscr{A}}(\lambda) := \Pr[c = (XY) * g : \Gamma[\hat{G}, G, g, q, A] \xleftarrow{R} [S_\lambda];\\ X, Y \xleftarrow{R} F_q[A]; c \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, X * g, Y * g)].\end{aligned}$$

Definition 3.4. *(ECDH assumption). The ECDH assumption for $\mathcal{G}$ is this: For every probabilistic, polynomial-time algorithm $\mathscr{A}$, the function $AdvECDH_{\mathcal{G},\mathscr{A}}(\lambda)$ is negligible in λ.*

Theorem 3.5. *CDH assumption (for a computational group scheme $\mathcal{G}'$) $\Rightarrow$ ECDH assumption (for the generalized computational group scheme $\mathcal{G}$).*

Proof. Assume that the ECDH assumption does not hold for the generalized computational group scheme $\mathcal{G}'$. Then there exists a probabilistic algorithm $\mathscr{A}$ such that for some fixed $c > 0$ and for all sufficiently large λ

$$\Pr[c = (XY) * g : \Gamma[\hat{G}, G, g, q, A] \xleftarrow{R} [S_\lambda]; X, Y \xleftarrow{R} F_q[A]; c \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, X * g, Y * g)] > 1/\lambda^c.$$

Let $\Gamma'[\hat{G}, G, \gamma, q] \xleftarrow{R} [S'_\lambda]$, $x, y \xleftarrow{R} F_q$. For the input $(1^\lambda, \Gamma', \gamma^x, \gamma^y)$, we define $\mathscr{A}'$ as follows:

1. Generate a companion matrix A of a random irreducible polynomial over finite field F_q of degree 2.
2. Generate randomly a generator $g_2 \in G$ such that γ, g_2 are not equal each other. Select $x_2, y_2 \in F_q$ uniformly at random.
3. Denote $g = (\gamma, g_2)$. Denote

$$\dot{h} = (\gamma^x, g_2^{x_2}), \qquad \ddot{h} = (\gamma^y, g_2^{y_2}).$$

   Select $R, S \in F_q[A]$ uniformly at random. Compute

$$h_1 = \dot{h} \cdot (R * g), h_2 = \ddot{h} \cdot (R * g).$$

4. Invoke $\mathscr{A}$ on input $(1^\lambda, \Gamma[\hat{G}, G, g, q, A], h_1, h_2)$ and get output h. Compute

$$h' = h \cdot [(-S) * \dot{h}] \cdot [(-R) * \ddot{h}] \cdot [(-SR) * g].$$

5. Output $c' = (h')_1$, where $(h')_1$ denotes the first component of h'.

By the definition of $\mathscr{A}'$, we know that when $\Gamma'[\hat{G}, \gamma, g, q]$ is distributed uniformly in $[S'_\lambda]$, the corresponding $\Gamma[\hat{G}, G, g, q, A]$ is distributed uniformly in $[S_\lambda]$. And the matrices $X = diag(x, x_2) + R$ and $Y = diag(y, y_2) + S$ are also distributed uniformly in $F_q[A]$. Hence we have

$$\begin{aligned}\mathrm{AdvCDH}_{\mathcal{G}',\mathscr{A}'}(\lambda) :&= \Pr[c' = \gamma^{xy} : \Gamma'[\hat{G}, G, \gamma, q] \xleftarrow{R} [S'_\lambda]; x, y \xleftarrow{R} F_q; c' \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma', \gamma^x, \gamma^y)] \\ &= \Pr[h = (XY) * g : h \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma[\hat{G}, G, g, q, A], h_1, h_2)] \\ &> \frac{1}{\lambda^c}\end{aligned}$$

3.4 EDDH Problem and EDDH Assumption

In this subsection, we propose a new generalized DDH problem and analyze the relationship between it and DDH problem.

Definition 3.6. *(Extended DDH problem (EDDHP)) Let $\mathcal{G}$ be a generalized computational group scheme, specifying a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of generalized group distributions. Let $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$. Given $(X*g), (Y*g), (Z*g) \in G^2$ for some $X, Y, Z \in F_q[A]$, decide whether $(XY) * g = Z * g$.*

For all $\lambda \in Z_{\geq 0}$, and for all $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$, we define the sets $\mathcal{D}_{\lambda,\Gamma}$ and $\mathcal{T}_{\lambda,\Gamma}$ as follows:

$$\begin{aligned}&\mathcal{D}_{\lambda,\Gamma} := \{(X * g, Y * g, (XY) * g) \in (G^2)^3 | X, Y \in F_q[A]\}; \\ &\mathcal{T}_{\lambda,\Gamma} := (G^2)^3.\end{aligned}$$

The set $\mathcal{D}_{\lambda,\Gamma}$ is called the set of "EDH triples." Also, for $\rho \in (G^2)^3$, define $\mathrm{EDHP}_{\lambda,\Gamma}(\rho) = 1$ if $\rho \in \mathcal{D}_{\lambda,\Gamma}$, and otherwise, define $\mathrm{EDHP}_{\lambda,\Gamma}(\rho) = 0$.

For all 0/1-valued, probabilistic, polynomial-time algorithms $\mathscr{A}$, and for all $\lambda \in Z_{\geq 0}$, we define the *EDDH advantage of $\mathscr{A}$ against $\mathcal{G}$ at λ* as

$$\begin{aligned}&\mathrm{AdvEDDH}_{\mathcal{G},\mathscr{A}}(\lambda) := \\ &|\Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)] - \\ &\Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)]|.\end{aligned}$$

Definition 3.7. *(EDDH assumption) The EDDH assumption for $\mathcal{G}$ is this: For every probabilistic, polynomial-time, 0/1-valued algorithm $\mathscr{A}$, the function $AdvEDDH_{\mathcal{G},\mathscr{A}}(\lambda)$ is negligible in λ.*

For all 0/1-valued, probabilistic, polynomial-time algorithms $\mathscr{A}$, for all $\lambda \in Z_{\geq 0}$, and for all $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$, we define the *EDDH advantage of $\mathscr{A}$ against $\mathcal{G}$ at λ given Γ* as

$$\begin{aligned}&\mathrm{AdvEDDH}_{\mathcal{G},\mathscr{A}}(\lambda|\Gamma) := |\Pr[\tau = 1 : \rho \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)] - \\ &\Pr[\tau = 1 : \rho \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)]|.\end{aligned}$$

Now we discuss the random self-reducibility property of the EDDH problem, and its implications. The following lemma states the random self-reducibility property for the EDDH problem.

Lemma 3.8. *There exists a probabilistic, polynomial-time algorithm RSR such that for all $\lambda \in Z_{\geq 0}$, for all $\Gamma \in [S_\lambda]$, and for all $\rho \in \mathcal{T}_{\lambda,\Gamma}$, the distribution $RSR(1^\lambda, \Gamma, \rho)$ is $\mathbf{U}(\mathcal{D}_{\lambda,\Gamma})$* if $\rho \in \mathcal{D}_{\lambda,\Gamma}$, and is $\mathbf{U}(\mathcal{T}_{\lambda,\Gamma})$ *if $\rho \notin \mathcal{D}_{\lambda,\Gamma}$.*

Proof. The algorithm RSR is very simple. Given 1^λ, the generalized group description $\Gamma[\hat{G}, G, g, q, A]$, and $\rho = (X * g, Y * g, Z * g) \in (G^2)^3$, the algorithm chooses S_1, S_2 and R uniformly in $F_q[A]$ and computes $(X' * g, Y' * g, Z' * g) \in (G^2)^3$ as follows:

$$\begin{aligned}X' * g &= [R * (X * g)](S_1 * g) \\ Y' * g &= (Y * g)(S_2 * g) \\ Z' * g &= [R * (Z * g)][(RS_2) * (X * g)][S_1 * (Y * g)][(S_1 S_2) * g].\end{aligned}$$

Let $Z = XY + E$ for some $E \in F_q[A]$ then:

$$X' = RX + S_1, \quad Y' = Y + S_2, \quad Z' = X'Y' + ER.$$

If $E = 0_{F_q[A]}$ we have that X' and Y' are uniformly distributed in $F_q[A]$ and $Z' = X'Y'$. If $E \neq 0_{F_q[A]}$, then X', Y' and Z' are all uniformly distributed in $F_q[A]$ (since $F_q[A]$ is a finite field).

The implication of the random self-reduction of EDDH problem is that if EDH tuples can be efficiently distinguished from random tuples with a non-negligible advantage, then EDH tuples can be efficiently recognized with negligible error probability. More formally, we have the following:

Theorem 3.9. *For every be a 0/1-valued, probabilistic, polynomial-time algorithm $\mathscr{A}$, and every polynomial P (with integer coefficients, taking positive values on $Z_{\geq 0}$), there exists a 0/1-valued, probabilistic, polynomial-time algorithm $\mathscr{A}_1$ such that for all $\lambda \in Z_{\geq 0}$, for all $\Gamma \in [S_\lambda]$, for all $\rho \in \mathcal{T}_{\lambda,\Gamma}$, and for all $\kappa \in Z_{\geq 0}$,*

if $\mathrm{AdvEDDH}_{\mathcal{G},\mathscr{A}}(\lambda|\Gamma) \geq 1/P(\lambda)$, *then* $\Pr[\tau \neq \mathrm{EDHP}_{\lambda,\Gamma}(\rho) : \tau \xleftarrow{R} \mathscr{A}_1(1^\lambda, \Gamma, \rho, 1^\kappa)] \leq 2^{-\kappa}$.

Proof. Let $\lambda \in Z_{\geq 0}$ and $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$. Assume that $\mathrm{AdvEDDH}_{\mathcal{G},\mathscr{A}}(\lambda|\Gamma) \geq 1/P(\lambda)$. Then

$$|\Pr[\tau = 1 : \rho \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)] - \Pr[\tau = 1 : \rho \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho)]| \geq 1/P(\lambda).$$

By Lemma 3.10, there exists a probabilistic polynomial-time algorithm $\mathscr{R}$ such that for *any* $\delta = (X * g, Y * g, Z * g) \in \mathcal{T}_{\lambda,\Gamma} \setminus \mathcal{D}_{\lambda,\Gamma}$:

$$|\Pr[\tau = 1 : \tau \xleftarrow{R} \mathscr{A}(\mathscr{R}(1^\lambda, \Gamma, \delta'))] - \Pr[\tau = 1 : \tau \xleftarrow{R} \mathscr{A}(\mathscr{R}(1^\lambda, \Gamma, \delta))]| \geq 1/P(\lambda),$$

where $\delta' = (X * g, Y * g, (XY) * g)$. Now the probabilities are only taken over the random bits of $\mathscr{A}$ and $\mathscr{R}$. Therefore, by standard methods of amplification we can define a probabilistic algorithm $\mathscr{A}_1$ such that for *any* $\delta = (X * g, Y * g, Z * g) \in \mathcal{T}_{\lambda,\Gamma} \setminus \mathcal{D}_{\lambda,\Gamma}$:

$$|\Pr[\tau = 1 : \tau \xleftarrow{R} \mathscr{A}_1(1^\lambda, \Gamma, \delta', 1^\kappa)] - \Pr[\tau = 1 : \tau \xleftarrow{R} \mathscr{A}_1(1^\lambda, \Gamma, \delta, 1^\kappa)]| \geq 1 - 2^{-\kappa}.$$

On any input $1^\lambda, \Gamma, \delta$ and 1^κ, the out put of $\mathscr{A}_1$ is essentially a threshold function of $\mathcal{O}(P(\lambda)^2\kappa)$ independent values - $\mathscr{A}(\mathscr{R}(1^\lambda, \Gamma, \delta))$. It is clear that $\mathscr{A}_1$ satisfies the conditions required in Theorem.

Theorem 3.10. *DDH assumption (for a computational group scheme $\mathcal{G}'$) $\Rightarrow$ EDDH assumption (for the generalized computational group scheme $\mathcal{G}$).*

Proof. Assume that the EDDH assumption does not hold for the generalized group scheme $\mathcal{G}$. Then there exists a 0/1-valued probabilistic algorithm $\mathscr{A}$ such that for some fixed $c > 0$ and for all sufficiently large λ

$$\begin{aligned}|\Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho_1 \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho_1)] - \\ \Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho_2 \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \rho_2)]| > 1/\lambda^c.\end{aligned}$$

For all $\lambda \in Z_{\geq 0}$, and for all $\Gamma'[\hat{G}, G, \gamma, q] \in [S'_\lambda]$, we define

$$\begin{aligned}\mathcal{P}_{\lambda,\Gamma'} &:= \{(\gamma^x, \gamma^y, \gamma^{xy}) | x, y \in F_q, x \neq 0\}; \\ \mathcal{Q}_{\lambda,\Gamma'} &:= \{(\gamma^a, \gamma^b, \gamma^c) | a, b, c \in F_q, a \neq 0, c \neq ab\}.\end{aligned}$$

Let $\Gamma'[\hat{G}, G, \gamma, q] \in [S'_\lambda]$. For the input $(1^\lambda, \Gamma', \gamma^x, \rho)$ (where $\rho = (\rho_1, \rho_2, \rho_3)$ is in either $\mathcal{P}_{\lambda,\Gamma'}$ or $\mathcal{Q}_{\lambda,\Gamma'}$), we define $\mathscr{A}'$ as follows:

1. Generate a companion matrix A of a random irreducible polynomial over finite field F_q of degree 2.

2. Generate randomly a generator $g_2 \in G$ such that γ, g_2 are not equal each other. Select $u_2, v_2 \in F_q$ uniformly at random. Denote $g = (\gamma, g_2)$. Denote

$$\rho'_1 =: (\rho_1, g_2^{u_2}), \rho'_2 =: (\rho_2, g_2^{v_2}), \rho'_3 =: (\rho_3, g_2^{u_2 v_2}).$$

Denote $\rho' = (\rho'_1, \rho'_2, \rho'_3)$.

4. Invoke the random self-reducibility algorithm RSR on input $(1^\lambda, \Gamma[\hat{G}, G, g, q, A], \rho')$ and get output ϱ.

5. Invoke $\mathscr{A}$ on input $(1^\lambda, \Gamma[\hat{G}, G, g, q, A], \varrho)$.

6. Output whatever $\mathscr{A}$ outputs.

By the definition of $\mathscr{A}'$, we know that when $\Gamma'[\hat{G}, \gamma, g, q]$ is distributed uniformly in $[S'_\lambda]$, the corresponding $\Gamma[\hat{G}, G, g, q, A]$ is distributed uniformly in $[S_\lambda]$. By Lemma 3.3, we know that ϱ is distributed uniformly in $\mathcal{D}_{\lambda,\Gamma}$ when ρ' is distributed uniformly in $\mathcal{P}_{\lambda,\Gamma'}$ and ϱ is distributed uniformly in $\mathcal{T}_{\lambda,\Gamma}$ when ρ' is distributed uniformly in $\mathcal{Q}_{\lambda,\Gamma'}$. Hence we

$$\begin{aligned}\mathrm{AdvDDH}'_{\mathcal{G}',\mathscr{A}'}(\lambda) &= |\Pr[\tau = 1 : \Gamma' \xleftarrow{R} [S'_\lambda]; \rho \xleftarrow{R} \mathcal{P}_{\lambda,\Gamma'}; \tau \xleftarrow{R} \mathscr{A}'(1^\lambda, \Gamma', \rho)] - \\ &\qquad \Pr[\tau = 1 : \Gamma' \xleftarrow{R} [S'_\lambda]; \hat{\rho} \xleftarrow{R} \mathcal{Q}_{\lambda,\Gamma'}; \tau \xleftarrow{R} \mathscr{A}'(1^\lambda, \Gamma', \hat{\rho})]| \\ &= |\Pr[\tau = 1 : \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \varrho)] - \Pr[\tau = 1 : \tau \xleftarrow{R} \mathscr{A}(1^\lambda, \Gamma, \hat{\varrho})]| \\ &> \frac{1}{\lambda^c}\end{aligned}$$

By Lemma 1 of [3], this implies that the DDH assumption for the computational group scheme $\mathcal{G}'$ does not hold.

4 New Generalized ElGamal Public Key Cryptosystem

New scheme makes use of a generalized computational group scheme $\mathcal{G}$ as described in section 3, defining a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of distributions of generalized group descriptions, and providing a sampling algorithm $\hat{S}$, where the output distribution $\hat{S}(1^\lambda)$ closely approximates S_λ.

(1) Key generation algorithm $\mathscr{G}$

On input 1^λ for $\lambda \in Z_{\geq 0}$, compute

$$\Gamma[\hat{G}, G, g, q, A] \xleftarrow{R} \hat{S}(1^\lambda); \quad B \xleftarrow{R} F_q[A]; \quad h = B * g$$

and output the public key $e = (\Gamma, h)$ and the secret key $d = (\Gamma, B)$.

(2) Encryption algorithm $\mathscr{E}$

Given 1^λ for $\lambda \in Z_{\geq 0}$, a public key $e = (\Gamma, h)$ along with a message $m \in G^2$, compute

$$R \xleftarrow{R} F_q[A]; \quad c_1 = R * g; \quad c_2 = (R * h)m$$

and output the ciphertext $c = (c_1, c_2)$.

(3) Decryption algorithm $\mathscr{D}$

Given 1^λ for $\lambda \in Z_{\geq 0}$, a secret key $d = (\Gamma, B)$ along with a ciphertext c, compute $f = (B * c_1)^{-1}$, $m = fc_2$ and output m.

Remark. Note that this encryption scheme has a restricted message space: messages are elements of the group G^2. It is straightforward to verify that this encryption scheme satisfies the basic requirements that any public key encryption scheme should satisfy. We can verify that decryption algorithm actually decrypts to the original message m. By the fact that "$*$" is an action of the commutative semigroup $(F_q[A], \cdot)$ on the group G^2, we have

$$\begin{aligned}(B * c_1)^{-1}c_2 &= (B * (R * g))^{-1}(R * h)m \\ &= ((BR) * g)^{-1}(R * (B * g))m \\ &= ((BR) * g)^{-1}((RB) * g)m \\ &= m.\end{aligned}$$

5 Security of the Scheme

In this section we show that the new PKC is one-way if and only if ECDH assumption holds and it is semantically secure EDDH assumption holds.

Theorem 5.1. *The new generalized ElGamal public-key encryption scheme is one-way if and only if ECDH assumption holds.*

Proof. Assume that ECDH assumption does not hold. Then there is a probabilistic polynomial algorithm $\mathcal{A}$ such that for some fixed $\theta > 0$, for all sufficiently large λ,

$$AdvECDH_{\mathcal{G},\mathcal{A}}(\lambda) > \frac{1}{\lambda^\theta}.$$

This means that

$$Pr[h = (XY)*g | \Gamma[\hat{G}, G, g, q, A] \xleftarrow{R} \hat{S}(1^\lambda); X, Y \xleftarrow{R} F_q[A]; h \xleftarrow{R} \mathcal{A}(1^\lambda, \Gamma, X*g, Y*g)] > \frac{1}{\lambda^\theta}.$$

Given public key $e = (\Gamma, h)$ and ciphertext $c = (c_1, c_2)$, where $h = B * g$, $c_1 = R * g$, we construct an algorithm $\mathcal{B}$ as follows

$$\mathcal{B}(e, c) = [\mathcal{A}(1^\lambda, \Gamma, h, c_1)]^{-1} \cdot c_2.$$

If $e = (\Gamma[\hat{G}, G, g, q, A], h) \xleftarrow{R} \mathscr{G}(1^\lambda)$, $d = (\Gamma, B) \xleftarrow{R} \mathscr{G}(1^\lambda)$, then $\Gamma[\hat{G}, G, g, q, A] \xleftarrow{R} [S_\lambda]$, $B, R \xleftarrow{R} F_q[A]$. It follows that for the fixed $\theta > 0$ and all sufficiently large λ

$$Pr[B(e, c) = D(d, c)] = Pr[\mathcal{A}(1^\lambda, \Gamma, h, c_1) = (RB) * g] > \frac{1}{\lambda^\theta},$$

where $e = (\Gamma[\hat{G}, G, g, q, A], h) \xleftarrow{R} \mathscr{G}(1^\lambda)$, $d = (\Gamma, B) \xleftarrow{R} \mathscr{G}(1^\lambda)$, $m \xleftarrow{R} G^2$ and $c = (c_1, c_2) \xleftarrow{R} \mathscr{E}(e, m)$.

Conversely, assume that this new generalized ElGamal encryption scheme is not one-way. Then by the definition 1.3 there is a probabilistic polynomial algorithm $\mathcal{C}$ such that for some fixed $\theta > 0$, for all sufficiently large λ,

$$Pr[\mathcal{C}(e, c) = m] > \frac{1}{\lambda^\theta} \tag{1}$$

where $e = (\Gamma[\hat{G}, G, g, q, A], h) \overset{R}{\leftarrow} \mathscr{G}(1^\lambda)$, $m \overset{R}{\leftarrow} G^2$ and $c = (c_1, c_2) \overset{R}{\leftarrow} \mathscr{E}(e, m)$.

Let $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$ and $X, Y \in F_q[A]$. For the input $(1^\lambda, \Gamma, X * g, Y * g)$, we construct an algorithm $\mathcal{D}$ as follows:

1. Select $R, S \in F_q[A]$ uniformly at random. Compute $h_1 = (X + R) * g$ and $h_2 = (Y + S) * g$.
2. Denote $e = (\Gamma, h_2)$. Invoke $\mathcal{C}$ on input $(1^\lambda, e, (h_1, 1))$ and get output w.
3. Compute and output $w^{-1} \cdot [(-S) * (X * g)] \cdot [(-R) * (Y * g)] \cdot [(-SR) * g]$.

Since

$$(h_1, 1) = ((X + R) * g, [(X + R) * ((Y + S) * g)] \cdot [((X + R)(Y + S)) * g]^{-1}),$$

$(h_1, 1)$ is a ciphertext of $m = [((X + R)(Y + S)) * g]^{-1}$ on the public key $e = (\Gamma, h_2)$. If the output of $\mathcal{C}$ is $m = [((X + R)(Y + S)) * g]^{-1}$, then

$$\begin{aligned} & w^{-1} \cdot [(-S) * (X * g)] \cdot [(-R) * (Y * g)] \cdot [(-SR) * g] \\ &= [((X + R)(Y + S)) * g] \cdot [(-S) * (X * g)] \cdot [(-R) * (Y * g)] \cdot [(-SR) * g] \\ &= (XY) * g \end{aligned}$$

When $\Gamma[\hat{G}, G, g, q, A] \overset{R}{\leftarrow} [S_\lambda]$ and $X, Y \overset{R}{\leftarrow} F_q[A]$, the distributions of $e = (\Gamma, h_2)$, $m = [((X + R)(Y + S)) * g]^{-1}$ and $(h_1, 1)$ are the same as that of the e, m, c in equation (1). Therefore, for the same fixed θ as above, for sufficiently large λ, we have

$$\begin{aligned} & Pr[\mathcal{D}(1^\lambda, \Gamma, X * g, Y * g) = (XY) * g] \\ &= Pr[\mathcal{C}((\Gamma, h_2), (h_1, 1)) = [((X + R)(Y + S)) * g]^{-1}] \\ &> \tfrac{1}{\lambda^\theta}, \end{aligned}$$

where $\Gamma[\hat{G}, G, g, q, A] \overset{R}{\leftarrow} [S_\lambda]$ and $X, Y \overset{R}{\leftarrow} F_q[A]$. This contradicts ECDH assumption.

Theorem 5.2. *The new generalized ElGamal public-key encryption scheme is semantically secure if and only if EDDH assumption holds.*

Proof. ($\Rightarrow$) Assume that the scheme is not semantically secure. According to Definition 1.4, this means that there exist a probabilistic polynomial algorithm $\mathscr{M}$ and $\mathcal{A}_D$ such that for some fixed $\theta > 0$, for all sufficiently large λ,

$$Pr[\mathcal{A}_D(e, m_a, m_b, c) = m] > \frac{1}{2} + \frac{1}{\lambda_\theta}, \tag{2}$$

where $e = (\Gamma[\hat{G}, G, g, q, A], h) \overset{R}{\leftarrow} \mathscr{G}(1^\lambda)$, $\{m_a, m_b\} \overset{R}{\leftarrow} \mathscr{M}(1^\lambda)$, $m \overset{R}{\leftarrow} \{m_a, m_b\}$ and $c = (c_1, c_2) \overset{R}{\leftarrow} \mathscr{E}(e, m)$. Then we can construct another efficient algorithm $\mathcal{A}_S$ that uses $\mathcal{A}_D$ to contradict the EDDH assumption.

By the key generation algorithm, we know that if $\Gamma \xleftarrow{R} [S_\lambda]$ and $\rho = (\rho_1, \rho_2, \rho_3) \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma}$ then (Γ, ρ_2) has the same distribution as $e = (\Gamma, h)$. Let $\Gamma[\hat{G}, G, g, q, A] \in [S_\lambda]$. It is easy to verify that for any $m \in G^2$ and $\rho = (\rho_1, \rho_2, \rho_3) \in \mathcal{T}_{\lambda,\Gamma}$, $(\rho_1, \rho_3 \cdot m)$ is a valid encryption of m on the public key (Γ, ρ_2) if and only if $\rho \in \mathcal{D}_{\lambda,\Gamma}$.

Let $\Gamma \xleftarrow{R} [S_\lambda]$ and $\rho = (\rho_1, \rho_2, \rho_3) \in \mathcal{T}_{\lambda,\Gamma}$. We define $\mathscr{A}_S$ as

$$\mathscr{A}_S(1^\lambda, \Gamma, \rho) = \begin{cases} 1 \text{ if } \mathscr{A}_D((\Gamma, \rho_2), m_a, m_b, (\rho_1, \rho_3 \cdot m)) = m; \\ 0 \text{ otherwise,} \end{cases}$$

where $m \in \{m_a, m_b\}$. Then, for the same fixed θ, for all sufficiently large λ,

$$\begin{aligned} \mathrm{AdvEDDH}_{\mathcal{G},\mathscr{A}_S}(\lambda) :&= |\Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}_S(1^\lambda, \Gamma, \rho)] - \\ &\quad \Pr[\tau = 1 : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma}; \tau \xleftarrow{R} \mathscr{A}_S(1^\lambda, \Gamma, \rho)]| \\ &= |\Pr[\nu = m : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma}; \nu \xleftarrow{R} \mathscr{A}_D((\Gamma, \rho_2), m_a, m_b, (\rho_1, \rho_3 \cdot m))] \\ &\quad - \Pr[\nu = m : \Gamma \xleftarrow{R} [S_\lambda]; \rho' \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma}; \nu \xleftarrow{R} \mathscr{A}_D((\Gamma, \rho'_2), m_a, m_b, (\rho'_1, \rho'_3 \cdot m))]| \\ &> (\tfrac{1}{2} + \tfrac{1}{k^\alpha}) - \tfrac{1}{2}, \end{aligned} \tag{3}$$

where $\{m_a, m_b\} \xleftarrow{R} \mathscr{M}(1^\lambda)$ and $m \xleftarrow{R} \{m_a, m_b\}$.

Now we prove (3) holds. Since the distribution of $(\rho_1, \rho_3 m)$ is the same as that of $c = (c_1, c_2)$ in (2), we have

$$|\Pr[\nu = m : \Gamma \xleftarrow{R} [S_\lambda]; \rho \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma}; \nu \xleftarrow{R} \mathscr{A}_D((\Gamma, \rho_2), m_a, m_b, (\rho_1, \rho_3 \cdot m))] > \frac{1}{2} + \frac{1}{k^\alpha}.$$

On the other hand, $(\rho'_1, \rho'_3 \cdot m)$ represents the encryption of a random message in $G^2 \setminus \{m\}$, and hence, except with negligible probability, it does not reveal any information about m. Therefore the probability that $\mathscr{A}_D$ outputs the correct m in such case is essentially $1/2$. So (3) holds. It contradict the EDDH assumption.

($\Leftarrow$) Assume that EDDH assumption does not holds. By Theorem **, there is a 0/1-valued, probabilistic, polynomial-time algorithm $\mathscr{B}$ such that for all $\lambda \in Z_{\geq 0}$, $\Gamma \in [S_\lambda]$, for all $\rho \in \mathcal{T}_{\lambda,\Gamma}$, and $\kappa \in Z_{\geq 0}$,

$$\Pr[\tau \neq \mathrm{EDHP}_{\lambda,\Gamma}(\rho) : \tau \xleftarrow{R} \mathscr{B}(1^\lambda, \Gamma, \rho, 1^\kappa)] \leq 2^{-\kappa}.$$

Let a public key $e = (\Gamma[\hat{G}, G, g, q, A], h)$, where $h = B * g$. Select at random two messages $m_0, m_1 \in G^2$. Suppose that their ciphertexts are $(R_0 * g, (R_0 * h)m_i)$ and $(R_1 * g, (R_1 * h)m_{1-i})$, where $i \xleftarrow{R} \{0, 1\}$ and $R_0, R_1 \xleftarrow{R} F_q[A]$. We will efficiently distinguish the ciphertexts of m_0 and m_1 by $\mathscr{B}$.

Select at random $V \xleftarrow{R} F_q[A]$, and compute

$$(V * g) \cdot h, \qquad [V * (R_0 * g)] \cdot (R_0 * h) \cdot m_i m_0^{-1}.$$

Since G^2 is a vector space on $F_q[A]$, we have

$$(V * g) \cdot h = (V * g)(B * g) = (V + B) * g$$

and

$$\begin{aligned}[V * (R_0 * g)] \cdot (R_0 * h) \cdot m_i m_0^{-1} &= [(VR_0) * g] \cdot (R_0 * (B * g)) \cdot m_i m_0^{-1} \\ &= [(VR_0) * g] \cdot ((R_0 B) * g) \cdot m_i m_0^{-1} \\ &= [(VR_0 + R_0 B) * g] \cdot m_i m_0^{-1} \\ &= [(R_0(V + B)) * g] \cdot m_i m_0^{-1}.\end{aligned}$$

If $m_i = m_0$, then $i = 0$ and it means that the first ciphertext is corresponding to the first message. In this case,

$$Pr[\mathscr{B}(1^\lambda, e, \{R_0 * g, (V * g)h, [V * (R_0 * g)] \cdot (R_0 * h) \cdot m_i m_0^{-1}\}) \neq 1] < \frac{1}{2^\kappa}.$$

If $m_i \neq m_0$, then $i = 1$ and it means that the first ciphertext is corresponding to the second message. In this case,

$$Pr[\mathscr{B}(1^\lambda, e, \{R_0 * g, (V * g)h, [V * (R_0 * g)] \cdot (R_0 * h) \cdot m_i m_0^{-1}\}) \neq 0] < \frac{1}{2^\kappa}.$$

This means that we can efficiently distinguish the corresponding ciphertexts of m_0, m_1 and it contradicts the semantic security of the new generalized ElGamal PKC.

Let p, q be prime numbers and $q|p-1$. If we adopt $\hat{G} = Z_p^*$ and G is the q-order subgroup of Z_p^*, then we can compare the original ElGamal scheme, the linear ElGamal scheme [13] and our new generalized ElGamal scheme as follows:

The comparison of ElGamal encryption schemes

Scheme	ElGamal [5]	Linear ElGamal [13]	This generalized ElGamal
Secret-key length	1	2	2
Encryption	2E	3E	4ME
Decryption	1E	1ME	2ME
Message expansion	double	triplicity	double
Assumption	CDH, DDH	CDH, Linear DDH	ECDH, EDDH

(An element of Z_q denotes a unit of secret-key length; E denotes exponentiation; ME denotes multi-exponentiations.)

We can obtain chosen ciphertext security by applying the generic scheme of Fujisaki and Okamoto [6] at practically no extra computational cost in the random oracle model.

References

1. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
2. Brands, S.: An efficient off-line electronic cash system based on the representation problem. CWI Technical Report, CS-R9323 (1993)
3. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. of Computing 33, 167–226 (2003)

4. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inform. Theory 22(6), 644–654 (1976)
5. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory 31(4), 469–472 (1985)
6. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)
7. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
8. Joux, A., Nguyen, K.: Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. J. Cryptology 16(4), 239–247 (2003)
9. Lidl, R., Niederreiter, H.: Introduction to finite fields and their applications. Cambridge University Press, Cambridge (1986)
10. Maze, G., Monico, C., Rosenthal, J.: A public key cryptosystem based on actions by semigroups. In: Proceedings of the 2002 IEEE International Symposium on Information Theory, Lausanne, Switzerland, p. 484 (2002)
11. Maze, G., Monico, C., Rosenthal, J.: Public key cryptography based on simple modules over simple rings. In: Proceedings of the 2002 Mathematical Theory of Networks and System, pp. 8–16. University of Notre Dame, Notre Dame (2002)
12. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton (1997)
13. Schacham, H.: A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants, http://eprint.iacr.org/2007/074.pdf
14. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: McCurley, K.S., Ziegler, C.D. (eds.) Advances in Cryptology 1981 - 1997. LNCS, vol. 1440, pp. 256–266. Springer, Heidelberg (1999)

Appendix: EDDH Problem in Generic Bilinear Groups

To provide more confidence in the EDDH assumption, we prove a lower bound on the computational complexity of the EDDH problem for generic groups in the sense of Shoup [14]. In this model, elements of group appear to be encoded as unique random strings, so that no property other than equality can be directly tested by the adversary.

We first review a few concepts related to bilinear maps.
1. G_1 and G_2 are two (multiplicative) cyclic groups of prime order q;
2. g_1 is a generator of G_1 and g_2 is a generator of G_2;
3. ψ is a computable isomorphism from G_2 to G_1, with $\psi(g_2) = g_1$; and
4. e is a computable map $e : G_1 \times G_2 \to G_T$ with the following properties:
**Bilinearity: for all $u \in G_1$, $v \in G_2$ and $a, b \in Z$, $e(u^a, v^b) = e(u, v)^{ab}$.
** Non-degeneracy: $e(g_1, g_2) \neq 1$.

Throughout the paper, we consider bilinear maps $e : G_1 \times G_2 \to G_T$ where all groups G_1, G_2, G_T are multiplicative and of prime order q.

We say that two groups (G_1, G_2) as above are a bilinear group pair if the group action in G_1 and G_2, the map ψ, and the bilinear map e are all efficiently computable. In what follows, we consider only the case of $G_1 = G_2 = G$. It is

clear that we can chose ψ to be identical map and $g_1 = g_2 = g \in G$. It is known that DDH assumption does not hold in the bilinear groups.

Proposition A. *For a generalized computational group scheme $\mathcal{G}$, the EDDH problem is equivalent to the following problem: Let G be a cyclic group of prime order q, g a generator of G, $\lambda_1, \lambda_2 \in F_q^*$, $t(x) = x^2 - ax - b$ a irreducible polynomial over F_q. Given g^{λ_1}, g^{λ_2}, $g^{\lambda_1 u_1 + b\lambda_2 v_1}$, $g^{\lambda_2 u_1 + (a\lambda_2 + \lambda_1)v_1}$, $g^{\lambda_1 u_2 + b\lambda_2 v_2}$, $g^{\lambda_2 u_2 + (a\lambda_2 + \lambda_1)v_2}$, g^{c_1}, $g^{c_2} \in G$, where $u_1, v_1, u_2, v_2, c_1, c_2 \in F_q^*$, decide whether $c_1 = u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1)$ and $c_2 = v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2)$ hold simultaneously.*

Proof. Let $\mathcal{G}$ be a generalized computational group scheme, specifying a sequence $(S_\lambda)_{\lambda \in Z_{\geq 0}}$ of generalized group distributions. Then EDDH problem is as follows: Let $\Gamma[\hat{G}, G, \hat{g}, q, A] \in [S_\lambda]$. Given $(X * \hat{g}), (Y * \hat{g}), (Z * \hat{g}) \in G^2$ for some $X, Y, Z \in F_q[A]$, decide whether $(XY) * \hat{g} = Z * \hat{g}$.

Let g be a generator of G. Assume that $\hat{g} = (g^{\lambda_1}, g^{\lambda_2})$, where $\lambda_1, \lambda_2 \in F_q^*$. Let A be the companion matrix of an irreducible polynomial $t(x) = x^2 - ax - b$ over F_q. Let $X = u_1 I + v_1 A$, $Y = u_2 I + v_2 A$, where $u_1, v_1, u_2, v_2 \in F_q^*$. Then it is easy to verify that $X * \hat{g} = (g^{\lambda_1 u_1 + b\lambda_2 v_1}, g^{\lambda_2 u_1 + (a\lambda_2 + \lambda_1)v_1})$, $Y * \hat{g} = (g^{\lambda_1 u_2 + b\lambda_2 v_2}, g^{\lambda_2 u_2 + (a\lambda_2 + \lambda_1)v_2})$,

$$(XY) * \hat{g} = (g^{u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1)}, g^{v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2)}).$$

Assume that $Z * \hat{g} = (g^{c_1}, g^{c_2})$, where $c_1, c_2 \in F_q^*$. Then to decide whether $Z * \hat{g} = (XY) * \hat{g}$ is equivalent to decide whether $c_1 = u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1)$ and $c_2 = v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2)$ hold simultaneously.

Let G be a bilinear group, g a generator of G and $e : G \times G \to G_T$ is a bilinear map. In the generic group model, elements of G and G_T appear to be encoded as unique random strings, so that no property other than equality can be directly tested by the adversary. Three oracles are assumed to perform operations between group elements, such as computing the group action in each of the groups G, G_T, as well as the bilinear pairing $e : G \times G \to G_T$.

The opaque encoding of the elements of G is modeled as an injective function $\xi_1 : F_q \to \Xi_1$, where $\Xi_1 \subset 0, 1^*$, which maps all $x \in F_q$ to the string representation $\xi_1(g^x)$ of $g^x \in G$. Analogous maps $\xi_T : F_q \to \Xi_T$ for G_T is also defined. The attacker $\mathcal{A}$ communicates with the oracles using the ξ-representations of the group elements only.

Let $\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, c_2 \xleftarrow{R} F_q^*, T_0 = g^{u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1)}$, $T_1 = g^{c_1}$, $S_0 = g^{v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2)}$, $S_1 = g^{c_2}$, and $d_1, d_2 \xleftarrow{R} \{0, 1\}$. We show that no generic algorithm $\mathcal{A}$ that is given the encodings of g^{λ_1}, g^{λ_2}, $g^{\lambda_1 u_1 + b\lambda_2 v_1}$, $g^{\lambda_2 u_1 + (a\lambda_2 + \lambda_1)v_1}$, $g^{\lambda_1 u_2 + b\lambda_2 v_2}$, $g^{\lambda_2 u_2 + (a\lambda_2 + \lambda_1)v_2}$, $T_{d_1}, T_{1-d_1}, S_{d_2}$, S_{1-d_2} and makes up to r oracle queries can guess the value of d_1, d_2 with probability greater than $1/4 + \mathcal{O}(r^4/q^2)$.

Theorem B. *Let $\mathcal{A}$ be an algorithm that solves EDDH problem in the generic group model. Assume that ξ_1, ξ_T are random encoding functions for G, G_T. If $\mathcal{A}$ makes a total of at most r queries to the oracles computing the group action in G, G_T, and the bilinear pairing e, then*

$$\left| Pr\left[\begin{array}{c} \mathcal{A}\begin{pmatrix} q, \xi_1(1), \xi_1(\lambda_1), \xi_1(\lambda_2), \xi_1(\lambda_1 u_1 + b\lambda_2 v_1) \\ \xi_1(\lambda_2 u_1 + (a\lambda_2 + \lambda_1)v_1) \\ \xi_1(\lambda_1 u_2 + b\lambda_2 v_2), \xi_1(\lambda_2 u_2 + (a\lambda_2 + \lambda_1)v_2) \\ \xi_1(t_0), \xi_1(t_1), \xi_1(s_0), \xi_1(s_1) \end{pmatrix} = (d_1, d_2) : \\ \lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, c_2 \xleftarrow{R} F_q^*; d_1, d_2 \xleftarrow{R} \{0,1\}; t_{1-d_1} = c_1; s_{1-d_2} = c_2 \\ t_{d_1} = u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1) \\ s_{d_2} = v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2) \end{array} \right] - \frac{1}{4} \right| \leq \frac{16(r+11)^4}{q^2}$$

Proof. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$.
$\mathcal{B}$ maintains three lists of pairs, $L_1 = \{(F_{1,i}, \xi_{1,i}) : i = 0, ..., \tau_1 - 1\}$, $L_2 = \{(F_{2,i}, \xi_{2,i}) : i = 0, ..., \tau_2 - 1\}$, under the invariant that, at step τ in the game, $\tau_1 + \tau_2 = \tau + 11$. Here, the $F_{*,*} \in F_q[\Lambda_1, \Lambda_2, U_1, V_1, U_2, V_2, T_0, T_1, S_0, S_1]$ are polynomials in the indeterminates $\Lambda_1, \Lambda_2, U_1, V_1, U_2, V_2, T_0, T_1, S_0, S_1$ with coefficients in F_q. The $\xi_{*,*} \in \{0,1\}^*$ are arbitrary distinct strings.

The lists are initialized at step $\tau = 0$ by initializing $\tau_1 = 11, \tau_2 = 0$ and setting $F_{1,0} = 1, F_{1,1} = \Lambda_1, F_{1,2} = \Lambda_2, F_{1,3} = \Lambda_1 U_1 + b\Lambda_2 V_1, F_{1,4} = \Lambda_2 U_1 + (a\Lambda_2 + \Lambda_1)V_1, F_{1,5} = \Lambda_1 U_2 + b\Lambda_2 V_2, F_{1,6} = \Lambda_2 U_2 + (a\Lambda_2 + \Lambda_1)V_2, F_{1,7} = T_0, F_{1,8} = T_1, F_{1,9} = S_0, F_{1,10} = S_1$. The corresponding strings are set to arbitrary distinct strings in $\{0,1\}^*$.

We may assume that $\mathcal{A}$ only makes oracle queries on strings previously obtained form $\mathcal{B}$, since $\mathcal{B}$ can make them arbitrarily hard to guess. We note that $\mathcal{B}$ can determine the index i of any given string $\xi_{1,i}$ in L_1 (or $\xi_{2,i}$ in L_2), where ties between multiple matches are broken arbitrarily.

$\mathcal{B}$ starts the game by providing $\mathcal{A}$ with the encodings $\xi_{1,i}$ $(i = 0, 1, ..., 10)$. The simulator $\mathcal{B}$ responds to algorithm $\mathcal{A}$'s queries as follows.

Group action. Given two operands $\xi_{1,i}$ and $\xi_{1,j}$ with $0 \leq i, j < \tau_1$, compute $F_{1,\tau_1} \leftarrow F_{1,i} + F_{1,j}$. If $F_{1,\tau_1} = F_{1,l}$ for some $l < \tau_1$, set $\xi_{1,\tau_1} \leftarrow \xi_{1,l}$; otherwise, set ξ_{1,τ_1} to a string in $\{0,1\}^* \backslash \{\xi_{1,0}, ..., \xi_{1,\tau_1-1}\}$. Add $(F_{1,\tau_1}, \xi_{1,\tau_1})$ to the list L_1 and give ξ_{1,τ_1} to $\mathcal{A}$, then increment τ_1 by one. Group action queries in G_T is treated similarly.

Inversion. Given a string $\xi_{1,i}$ with $0 \leq i < \tau_1$, compute $F_{1,\tau_1} \leftarrow -F_{1,i}$. If $F_{1,\tau_1} = F_{1,l}$ for some $l < \tau_1$, set $\xi_{1,\tau_1} \leftarrow \xi_{1,l}$; otherwise, set ξ_{1,τ_1} to a string in $\{0,1\}^* \backslash \{\xi_{1,0}, ..., \xi_{1,\tau_1-1}\}$. Add $(F_{1,\tau_1}, \xi_{1,\tau_1})$ to the list L_1 and give ξ_{1,τ_1} to $\mathcal{A}$, then increment τ_1 by one. Inversion in G_T is handled analogously.

Bilinear map. Given two operands $\xi_{1,i}$ and $\xi_{1,j}$ with $0 \leq i, j < \tau_1$, compute the product $F_{2,\tau_2} \leftarrow F_{1,i} F_{1,j}$. If $F_{2,\tau_2} = F_{2,l}$ for some $l < \tau_2$, set $\xi_{2,\tau_2} \leftarrow \xi_{2,l}$; otherwise, set ξ_{2,τ_2} to a string in $\{0,1\}^* \backslash \{\xi_{2,0}, ..., \xi_{2,\tau_2-1}\}$. Add $(F_{2,\tau_2}, \xi_{2,\tau_2})$ to the list L_2 and give ξ_{2,τ_2} to $\mathcal{A}$, then increment τ_2 by one.

Observe that at any time in the game, the total degree of any polynomial in each of the two lists is bounded as follows: $\deg(F_{1,i}) \leq 2$, $\deg(F_{2,i}) \leq 4$.

After at most r queries, $\mathcal{A}$ terminates and returns a guess $\hat{d}_1, \hat{d}_2 \in \{0,1\}$. At this point $\mathcal{B}$ chooses random $\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, c_2 \leftarrow F_q$. For $d_1, d_2 \in \{0,1\}$, consider

$$t_{d_1} = u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1), \quad t_{1-d_1} = c_1$$

and

$$s_{d_2} = v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2), \quad s_{1-d_2} = c_2$$

The simulation provided by $\mathcal{B}$ is perfect and reveals nothing to $\mathcal{A}$ about d_1 unless the chosen random values for the indeterminates give rise to a non-trivial equality relation between the simulated group elements that was not revealed to $\mathcal{A}$, i.e., when we assign

$$\Lambda_1 \leftarrow \lambda_1, \quad \Lambda_2 \leftarrow \lambda_2, \quad U_1 \leftarrow u_1, \quad V_1 \leftarrow v_1, \quad U_2 \leftarrow u_2, \quad V_2 \leftarrow v_2$$

and either

$$T_0 \leftarrow u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1), \quad T_1 \leftarrow c_1$$

or the converse

$$T_1 \leftarrow u_2(\lambda_1 u_1 + b\lambda_2 v_1) + bv_2(\lambda_1 v_1 + \lambda_2 u_1 + a\lambda_2 v_1), \quad T_0 \leftarrow c_1$$

This happens only if for some i, j one of the following holds:

1. $F_{1,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, t, c_1, 0, 0) - F_{1,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, t, c_1, 0, 0) = 0$, yet $F_{1,i} \neq F_{1,j}$,
2. $F_{2,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, t, c_1, 0, 0) - F_{2,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, t, c_1, 0, 0) = 0$, yet $F_{2,i} \neq F_{2,j}$,
3. $F_{1,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, t, 0, 0) - F_{1,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, t, 0, 0) = 0$, yet $F_{1,i} \neq F_{1,j}$,
4. $F_{2,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, t, 0, 0) - F_{2,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, t, 0, 0) = 0$, yet $F_{2,i} \neq F_{2,j}$.

Similarly, the simulation provided by $\mathcal{B}$ is perfect and reveals nothing to $\mathcal{A}$ about d_2 unless the chosen random values for the indeterminates give rise to a non-trivial equality relation between the simulated group elements that was not revealed to $\mathcal{A}$, i.e., when we assign

$$\Lambda_1 \leftarrow \lambda_1, \quad \Lambda_2 \leftarrow \lambda_2, \quad U_1 \leftarrow u_1, \quad V_1 \leftarrow v_1, \quad U_2 \leftarrow u_2, \quad V_2 \leftarrow v_2$$

and either

$$S_0 \leftarrow v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2), \quad S_1 \leftarrow c_2$$

or the converse

$$S_1 \leftarrow v_1(\lambda_1 u_2 + b\lambda_2 v_2) + (u_1 + av_1)(\lambda_1 v_2 + \lambda_2 u_2 + a\lambda_2 v_2), \quad S_0 \leftarrow c_2$$

This happens only if for some i, j one of the following holds:

1. $F_{1,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, s, c_2) - F_{1,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, s, c_2) = 0$, yet $F_{1,i} \neq F_{1,j}$,
2. $F_{2,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, s, c_2) - F_{2,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, s, c_2) = 0$, yet $F_{2,i} \neq F_{2,j}$,
3. $F_{1,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, c_2, s) - F_{1,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, c_2, s) = 0$, yet $F_{1,i} \neq F_{1,j}$,
4. $F_{2,i}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, c_2, s) - F_{2,j}(\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, 0, 0, c_2, s) = 0$, yet $F_{2,i} \neq F_{2,j}$.

We first need to argue that the adversary is unable to engineer any of the above equalities, so that they can only occur due to an unfortunate random choice of $\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, c_2$. First, observe that the adversary can only manipulate the polynomials on the two lists through additions and subtractions (disguised as multiplications and divisions in the groups G, G_T as well as multiplications between polynomials which are not the result of a previous multiplication (disguised as pairings between elements of G). Now, notice that in the initial population of the lists, the only occurrence of the variable U_1, V_1 is within the polynomial $F_{1,3} = \Lambda_1 U_1 + b\Lambda_2 V_1$, $F_{1,4} = \Lambda_2 U_1 + (a\Lambda_2 + \Lambda_1)V_1$ and the only occurrence of the variable U_2, V_2 is within the polynomial $F_{1,5} = \Lambda_1 U_2 + b\Lambda_2 V_2$, $F_{1,6} = \Lambda_2 U_2 + (a\Lambda_2 + \Lambda_1)V_2$. Notice that

$$U_2(\Lambda_1 U_1 + b\Lambda_2 V_1) + bV_2(\Lambda_1 V_1 + \Lambda_2 U_1 + a\Lambda_2 V_1) = U_2 F_{1,3} + bV_2 F_{1,4}.$$

Since $t(x) = x^2 - ax - b$ is an irreducible polynomial over F_q, we have $b \neq 0$. It is clear that the adversary is unable to generate the polynomial $i(U_2 F_{1,3} + bV_2 F_{1,4})$ for any $i \in F_q^*$. Next, we prove that the adversary is unable to generate the polynomial $F(U_2 F_{1,3} + bV_2 F_{1,4})$, where F is a polynomial of degree at least 1 in L_1, L_2. Since the polynomials in L_1, L_2 have degree at most 4, F is a polynomial of degree at most 1. So we assume that $F = m\Lambda_1 + n\Lambda_2$ $(m, n \neq 0)$. Compute

$$(m\Lambda_1 + n\Lambda_2)(U_2 F_{1,3} + bV_2 F_{1,4}) = m\Lambda_1 U_2 F_{1,3} + n\Lambda_2 U_2 F_{1,3} + mb\Lambda_1 V_2 F_{1,4} + nb\Lambda_2 V_2 F_{1,4}.$$

Notice that $\Lambda_1 U_2 F_{1,3}$, $\Lambda_2 U_2 F_{1,3}$, $\Lambda_1 V_2 F_{1,4}$ and $\Lambda_2 V_2 F_{1,4}$ can be obtained by $F_{1,5}F_{1,3}$, $F_{1,6}F_{1,3}$, $F_{1,5}F_{1,4}$ and $F_{1,6}F_{1,4}$ respectively. It follows that the adversary can obtained the polynomial $(m\Lambda_1 + n\Lambda_2)(U_2 F_{1,3} + bV_2 F_{1,4})$ by computing

$$mF_{1,5}F_{1,3} + nF_{1,6}F_{1,3} + mbF_{1,5}F_{1,4} + nbF_{1,6}F_{1,4}.$$

Notice that

$$mF_{1,5}F_{1,3} + nF_{1,6}F_{1,3} + mbF_{1,5}F_{1,4} + nbF_{1,6}F_{1,4} = (mF_{1,5} + nF_{1,6})(F_{1,3} + bF_{1,4}).$$

Assume that $(mF_{1,5} + nF_{1,6})(F_{1,3} + bF_{1,4}) = (m\Lambda_1 + n\Lambda_2)(U_2 F_{1,3} + bV_2 F_{1,4})$. Then we can deduce $m = 0$ and $n = 0$ by comparing the coefficients of both sides. This contradicts $m, n \neq 0$. It follows that given the available operations, in the two group representations the adversary is unable to generate the polynomial $F(U_2 F_{1,3} + bV_2 F_{1,4})$ for any non-zero polynomial F of degree at most 1. Similarly,

$$V_1(\Lambda_1 U_2 + b\Lambda_2 V_2) + (U_1 + aV_1)(\Lambda_1 V_2 + \Lambda_2 U_2 + a\Lambda_2 V_2) = V_1 F_{1,5} + (U_1 + aV_1)F_{1,6}.$$

The adversary is unable to generate the polynomial $F(V_1F_{1,5} + (U_1 + aV_1)F_{1,6})$ for any non zero polynomial F of degree at most 1.

It remains only to bound the probability that a random choice of $\lambda_1, \lambda_2, u_1, v_1, u_2, v_2, c_1, c_2$ will cause some two distinct polynomials to have the same value. All polynomials in L_1 have degree at most 2, so any two such polynomials $F_{1,i}$ and $F_{1,j}$ are such that $F_{1,i}(...) = F_{1,j}(...)$ with probability at most $2/p$ over the choice of values. Similarly, all polynomials in L_2 have degree at most 4, so any two such polynomials $F_{2,i}$ and $F_{2,j}$ are such that $F_{2,i}(...) = F_{2,j}(...)$ with probability at most $4/p$. The lists are populated initially with 11 values. If the adversary makes r queries to its oracles then the lists contain at most $r + 11$ entries, so a sum over all pairs of entries gives a bound on the success probability of the adversary:

$$\epsilon \leq \left[2\binom{r+11}{2}\frac{4}{q}\right]^2 < \frac{16(r+11)^2}{q^2}$$

Remark. It is clear that for generic groups in Shoup model, the lower bound on the computational complexity of EDDH problem is less than that on the computational complexity of Linear DDH problem (see [1]), since $8(r+9)^2/q > 16(r+11)^4/q^2$ for $r \ll q$.

Tweakable Pseudorandom Permutation from Generalized Feistel Structure

Atsushi Mitsuda and Tetsu Iwata

Dept. of Computational Science and Engineering,
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan
a_mituda@nuee.nagoya-u.ac.jp, iwata@cse.nagoya-u.ac.jp

Abstract. Tweakable pseudorandom permutations have wide applications such as the disk sector encryption, and the underlying primitive for efficient MACs and authenticated encryption schemes. Goldenberg et al. showed constructions of a tweakable pseudorandom permutation based on the Feistel structure. In this paper, we explore the possibility of designing tweakable pseudorandom permutations based on the Generalized Feistel Structure. We show that tweakable pseudorandom permutations can be obtained without increasing the number of rounds compared to the non-tweakable versions. We also present designs that take multiple tweaks as input.

Keywords: Luby-Rackoff theory, tweakable blockcipher, generalized Feistel structure, security proofs.

1 Introduction

A Tweakable BlockCipher, or a TBC, formalized by Liskov, Rivest, and Wagner [15], is a blockcipher that takes an additional input called a tweak. A conventional blockcipher takes a key and a plaintext as inputs, while a TBC takes a key, a plaintext and a tweak as inputs. TBCs have wide applications. Liskov et al. described how they can be used to implement secure symmetric encryption and authenticated encryption schemes [15]. Rogaway showed that an efficient message authentication code and an authenticated encryption scheme can be constructed from them [22]. Halevi and Rogaway [11,12] suggested an application to disk sector encryption. Bellare and Kohno pointed out the relation between TBCs and blockciphers that are secure against related-key attacks [1].

For the conventional blockcipher, it is a PseudoRandom Permutation, or a PRP, if the adversary with chosen plaintext attacks cannot distinguish it from a random permutation, and it is a Strong PRP, or an SPRP, if the adversary with chosen plaintext/ciphertext attacks cannot distinguish it from a random permutation. For TBCs, the tweak is a public data known to the adversary, and a TBC is considered secure if it is indistinguishable from a *family* of random permutations indexed by the tweak. Specifically, a TBC is a Tweakable PRP, or a TPRP, if it is indistinguishable against adversary with chosen plaintext

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 22–37, 2008.

attacks, and it is a Tweakable SPRP, or a TSPRP, if it is indistinguishable against adversary with chosen plaintext/ciphertext attacks.

There are mainly three approaches to design TBCs. One is to start from the conventional blockcipher, and use it as a black-box to obtain a TBC. This approach was taken in [15], giving two constructions for TBCs. XE and XEX are also TBCs constructed from the conventional blockcipher [22]. Another approach is the direct construction, where one designs a TBC from the scratch. HPC [24] was the first to introduce an auxiliary blockcipher input called a "spice." Mercy [5] is another example, however both ciphers known to have weaknesses [26,13,8]. The last approach is in between the former two approaches. Goldenberg et al. [6] suggested the constructions based on the Feistel structure [7], showing that the constructions are *provably secure* in the Luby-Rackoff sense [16], in contrast to the scratch designs, while the blockcipher is not used as a black-box.

Our Contributions. In this paper, we explore the possibility of designing TBCs that are provably secure, in the Luby-Rackoff sense, based on *the Generalized Feistel Structure,* or the GFS. The structure was adopted in the design of RC6 [21], and more recently in HIGHT [14] and CLEFIA [25]. The structure is suitable for hardware efficient blockciphers, as it allows smaller F-functions compared to the Feistel structure. We follow the design strategy from [6], i.e., we directly XOR the tweak to some of the data lines, which allows efficient tweak updates. The provable security of non-tweakable GFS was analyzed by Zheng, Matsumoto, and Imai [28][1]. It was proved that, if the GFS has d data lines, then $(d+1)$ rounds are enough for a PRP. Moriai and Vaudenay proved that $2d$ rounds are enough for an SPRP [19]. One of our main theoretical contributions is to present the designs of TPRPs and TSPRPs *without* increasing the number of rounds needed for the non-tweakable PRP/SPRP. That is, our TPRP has $(d+1)$ rounds and TSPRP has $2d$ rounds. The Feistel structure, in contrast, was shown to be a PRP with three rounds, and SPRP with four rounds [16], and the tweakable version cannot be constructed with these number of rounds [15]. Indeed, the constructions in [15] are four rounds for a TPRP, and six rounds for a TSPRP, which are both shown to be optimal. See Table 1 for a comparison of required number of rounds.

As with [6], our designs use the concept of universal hash functions [2], and the core of the proof is to show that the $(d-1)$-round GFS yields a universal hash function. It requires careful treatments of case analysis, which we show by constructing a series of pseudocodes. The technique might be of independent interest, as it may be useful in proving the security of similar constructions.

We also consider the problem of handling longer tweaks, the problem addressed in [6]. Certain applications require different, specific tweak sizes. For example, in TAE mode [15], each tweak holds a variety of information such that each tweak is unique. Thus, one may want to allow longer tweaks to include more information. Now [6] shows that for the Feistel structure, it is *impossible*

[1] The construction we consider in this paper is called "Type-2 Generalized Feistel Structure" in [28].

Table 1. Summary of the previous results and our results on the required number of rounds. GFS denotes the Generalized Feistel Structure with d data lines. Our results show that tweakable versions do not require increasing the number of rounds.

	Feistel structure	GFS
(non-tweakable) PRP	three [16]	$d+1$ [28]
Tweakable PRP	four [15]	$d+1$ (This paper)
(non-tweakable) SPRP	four [16]	$2d$ [19]
Tweakable SPRP	six [15]	$2d$ (This paper)

to input multiple tweaks without increasing the number of rounds. In contrast, for GFS, we present constructions that take multiple tweaks *without* increasing the number of rounds and still maintain the provable security. We show that a linear code plays a crucial role in the design, which may be seen as another technical contribution that might be applied to the designs based on similar structures. We also present constructions for multiple tweaks by increasing the number of rounds. Finally, in the Appendix, we show some negative results on the construction of TPRPs.

Other Related Works. A tweakable enciphering scheme [11,12] is closely related to TBCs. The tweakable enciphering scheme uses a conventional blockcipher (or possibly a TBC) as a blackbox, and thus they can be viewed as a blockcipher mode of operation. They take a tweak and a message as inputs, where the message length can vary. There are a number of proposals [3,4,9,10,17,18,23,27]. The primary interest of this paper (and [15]) is the blockcipher itself, a lower level primitive than the tweakable enciphering schemes.

2 Preliminaries

For any positive integer n, let $\{0,1\}^n$ be the set of all n-bit strings. For any integers i and j such that $i \leq j$, let $[i,j]$ be the set $\{s \mid i \leq s \leq j\}$, and $[i,j)$ be $\{s \mid i \leq s < j\}$. Similarly, let $e[i,j]$ be the set $\{s \mid i \leq s \leq j \text{ and } s \text{ is even}\}$, and let $o[i,j]$ be $\{s \mid i \leq s \leq j \text{ and } s \text{ is odd}\}$. We also define $e[i,j)$ and $o[i,j)$ analogously. For a set S, $s \xleftarrow{R} S$ is the selection of s from S uniformly at random.

A *tweakable blockcipher*, or a *TBC* in short, is a function $E : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{K}$ is the key space and $\mathcal{T}$ is the tweak space, and for any key $K \in \mathcal{K}$ and any tweak $T \in \mathcal{T}$, $E(K,T,\cdot) = E_K(T,\cdot)$ is a permutation over $\{0,1\}^n$.

We follow the security notion from [15]. We consider two attack scenarios; chosen-plaintext attacks (CPA) and chosen-plaintext/ciphertext attacks (CCA). Let $\mathrm{Perm}(n)$ be the set of all permutations over $\{0,1\}^n$, and let $\mathrm{Perm}(\mathcal{T},n)$ be the set of all functions $\Pi : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ such that for any tweak $T \in \mathcal{T}$, $\Pi(T,\cdot) \in \mathrm{Perm}(n)$. That is, when we write $\Pi \xleftarrow{R} \mathrm{Perm}(\mathcal{T},n)$, then for each $T \in \mathcal{T}$, $\Pi(T,\cdot)$ is a random permutation over $\{0,1\}^n$. We also let $\mathrm{Func}(n)$ be the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$.

An *adversary* A is a (possibly probabilistic) algorithm with access to one or more oracles. Oracles are written as superscripts.

Definition 1. *Let* $E : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *be a TBC. For any CPA-adversary* A*, define* $\mathbf{Adv}_E^{\mathrm{tprp}}(A) \stackrel{\mathrm{def}}{=} |P_E - P_{\Pi}|$*, where*

- $P_E \stackrel{\mathrm{def}}{=} \Pr(K \xleftarrow{R} \mathcal{K}; A^{E_K(\cdot,\cdot)} = 1)$*, and*
- $P_{\Pi} \stackrel{\mathrm{def}}{=} \Pr(\Pi \xleftarrow{R} \mathrm{Perm}(\mathcal{T}, n); A^{\Pi(\cdot,\cdot)} = 1)$.

We also define $\mathbf{Adv}_E^{\mathrm{tprp}}(q) \stackrel{\mathrm{def}}{=} \max_A\{\mathbf{Adv}_E^{\mathrm{tprp}}(A)\}$*, where the maximum is taken over all adversaries that make at most* q *queries.*

Definition 2. *Let* $E : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *be a TBC. For any CCA-adversary* A*, define* $\mathbf{Adv}_E^{\mathrm{tsprp}}(A) \stackrel{\mathrm{def}}{=} |P_E - P_{\Pi}|$*, where*

- $P_E \stackrel{\mathrm{def}}{=} \Pr(K \xleftarrow{R} \mathcal{K}; A^{E_K(\cdot,\cdot),E_K^{-1}(\cdot,\cdot)} = 1)$*, and*
- $P_{\Pi} \stackrel{\mathrm{def}}{=} \Pr(\Pi \xleftarrow{R} \mathrm{Perm}(\mathcal{T}, n); A^{\Pi(\cdot,\cdot),\Pi^{-1}(\cdot,\cdot)} = 1)$.

We also define $\mathbf{Adv}_E^{\mathrm{tsprp}}(q) \stackrel{\mathrm{def}}{=} \max_A\{\mathbf{Adv}_E^{\mathrm{tsprp}}(A)\}$*, where the maximum is taken over all adversaries that make at most* q *queries.*

We say E is a Tweakable PseudoRandom Permutation (TPRP) if $\mathbf{Adv}_E^{\mathrm{tprp}}(q)$ is sufficiently small, and E is a Tweakable Strong PseudoRandom Permutation (TSPRP) if $\mathbf{Adv}_E^{\mathrm{tsprp}}(q)$ is sufficiently small.

3 Generalized Feistel Structure

We consider *the Generalized Feistel Structure*, or the *GFS* in short, which is parameterized by R and d. $GFS_{R,d}$ has R rounds in total, and has d data lines, where $d \geq 4$ is even, and each data line is n bits. It takes $f_{r,i} \in \mathrm{Func}(n)$ as the key, where $r \in [0, R)$ and $i \in [0, d/2)$. A plaintext and a ciphertext are dn bits, i.e., it takes a plaintext $x_0 = (x_{0,0}, x_{0,1}, \ldots, x_{0,d-1}) \in (\{0,1\}^n)^d$, and outputs a ciphertext $x_R = (x_{R,0}, x_{R,1}, \ldots, x_{R,d-1}) \in (\{0,1\}^n)^d$. The r-th round of $GFS_{R,d}$, denoted RF, takes the intermediate value $x_r = (x_{r,0}, x_{r,1}, \ldots, x_{r,d-1}) \in (\{0,1\}^n)^d$ as the input, and produces the output $x_{r+1} = (x_{r+1,0}, x_{r+1,1}, \ldots, x_{r+1,d-1}) \in (\{0,1\}^n)^d$ defined by

$$x_{r+1,i} = \begin{cases} x_{r,i+1} \oplus f_{r,i/2}(x_{r,i}) & \text{if } i \in e[0,d), \\ x_{r,i+1} & \text{otherwise,} \end{cases}$$

where $r \in [0, R)$ and we assume that mod d is taken to the second index in $x_{r,i}$, i.e., $x_{r,d}$ corresponds to $x_{r,0}$ (We also use the same convention for other variables). Therefore, the r-th round RF uses $d/2$ F-functions $f_{r,0}, f_{r,1}, \ldots, f_{r,d/2-1}$. Throughout this paper, we assume that F-functions, $f_{r,i}$'s, are random functions, i.e., $f_{r,i}$ is uniformly chosen at random from $\mathrm{Func}(n)$, and thus $f_{r,i} \xleftarrow{R} \mathrm{Func}(n)$.

3.1 Tweakable Blockcipher from Generalized Feistel Structure

We consider tweakable blockciphers based on $GFS_{R,d}$, where the tweaks are directly XOR-ed to the data lines. For $GFS_{R,d}$, define the locations $L_{r,i}$ and $\tilde{L}_{r,i}$ as in Fig. 1, where $r \in [0,R]$ and $i \in [0,d)$ for $L_{r,i}$, and $r \in [0,R)$ and $i \in e[0,d)$ for $\tilde{L}_{r,i}$. Let this set be Λ_R, i.e.,

$$\Lambda_R \stackrel{\text{def}}{=} \{L_{r,i} \mid r \in [0,R] \text{ and } i \in [0,d)\} \cup \{\tilde{L}_{i,r} \mid r \in [0,R) \text{ and } i \in e[0,d)\}.$$

Let T^λ be the XOR of all the tweaks used at location $\lambda \in \Lambda_R$. The construction we consider is:

$$x_{r+1,i} = \begin{cases} x_{r,i+1} \oplus T^{L_{r,i+1}} \oplus f_{r,i/2}(x_{r,i} \oplus T^{L_{r,i}} \oplus T^{\tilde{L}_{r,i}}) & \text{if } i \in e[0,d), \\ x_{r,i+1} \oplus T^{L_{r,i+1}} & \text{otherwise.} \end{cases}$$

See Fig. 2 for an illustration.

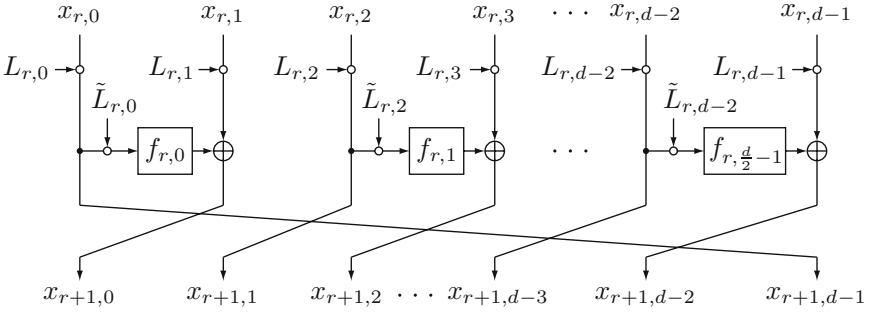

Fig. 1. Definition of the locations $L_{r,i}$ and $\tilde{L}_{r,i}$

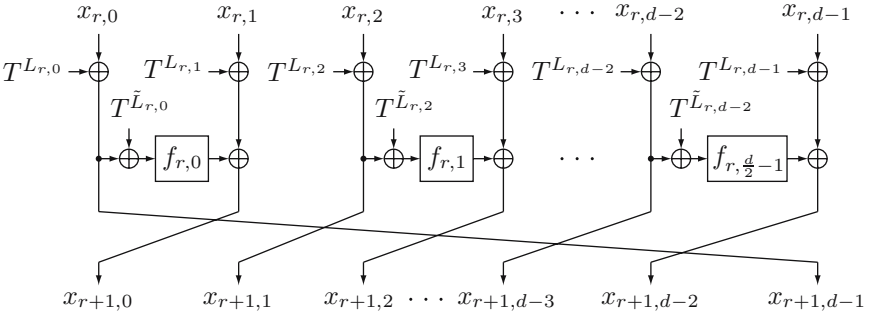

Fig. 2. The construction we consider in this paper

We use $GFS_{R,d}(\lambda)$ to refer to $GFS_{R,d}$, where a tweak T^λ is XOR-ed at the location $\lambda \in \Lambda_R$. To denote XOR-ing multiple tweaks, we write $GFS_{R,d}(\lambda_1, \ldots, \lambda_\tau)$ or $GFS_{R,d}(\Lambda)$, where Λ is a set such that $\Lambda \subseteq \Lambda_R$. Thus, in such a structure, the

tweak size is τn bits. When we XOR the same tweak at two or more locations, we write $GFS_{R,d}(\lambda_1 + \lambda_2)$, where the implication of using the compound location $\lambda_1 + \lambda_2$ is that $T^{\lambda_1} = T^{\lambda_2}$. In Λ_R, we have listed all tweaks at $\tilde{L}_{r,i}$ locations, however, as in the Feistel structure, we do not have to consider these locations.

Lemma 1. *For any $i \in e[0,d)$ and $r \in [0,R)$, $\tilde{L}_{r,i} = L_{r,i} + L_{r+1,i-1}$. Similarly, for any $i \in o[0,d)$ and $r \in [0,R)$, $L_{r,i} = L_{r+1,i-1}$.*

Since these locations are equivalent, we will use them interchangeably.

Lemma 2. *For all R, without loss of generality, we can consider only structures that never use the tweak locations $\{L_{0,i} \mid i \in e[0,d)\}$, $\{L_{1,i} \mid i \in e[0,d)\}$, or $\{L_{R,i} \mid i \in [0,d)\}$, even in compound locations, and even when considering CCA security.*

Proof. We can simulate oracle queries with or without the tweaks in these locations. To simulate a query $(T, x_{0,0}, x_{0,1}, \ldots, x_{0,d-1}) \in \mathcal{T} \times (\{0,1\}^n)^d$ to a structure with these locations, we make a query

$$(x_{0,0} \oplus T^{L_{0,0}}, x_{0,1} \oplus T^{L_{1,0}}, \ldots, x_{0,d-2} \oplus T^{L_{0,d-2}}, x_{0,d-1} \oplus T^{L_{1,d-2}})$$

to the structure without these tweaks to obtain $(x_{R,0}, x_{R,1}, \ldots, x_{R,d-1})$, and we return $(x_{R,0} \oplus T^{L_{R,0}}, x_{R,1} \oplus T^{L_{R,1}}, \ldots, x_{R,d-1} \oplus T^{L_{R,d-1}})$. Decryption queries can be simulated similarly. □

The set of tweak locations we need to consider is thus reduced, and we re-define Λ_R as $\Lambda_R \stackrel{\text{def}}{=} \{L_{r,i} \mid r \in [2,R) \text{ and } i \in e[0,d)\}$.

For any $\Lambda \in \Lambda_R$, we define the inverse of $GFS_{R,d}(\Lambda)$, which we denote $GFS^{-1}_{R,d}(\Lambda)$. For a given key and a tweak, $GFS_{R,d}(\Lambda)$ defines a permutation over $(\{0,1\}^n)^d$, and $GFS^{-1}_{R,d}(\Lambda)$ is the inverse of this permutation, i.e., $GFS^{-1}_{R,d}(\Lambda)$ takes a ciphertext $x_R \in (\{0,1\}^n)^d$ to return a plaintext $x_0 \in (\{0,1\}^n)^d$.

3.2 Almost Universal Hash Function [2]

We consider a function $H : \mathcal{K} \times \mathcal{T} \times \{0,1\}^{dn} \to \{0,1\}^{dn}$, where for any key $K \in \mathcal{K}$, $H(K,\cdot,\cdot) = H_K(\cdot,\cdot)$ takes $(T,x) \in \mathcal{T} \times \{0,1\}^{dn}$ as input and outputs $y = (y_0, \ldots, y_{d-1}) \in \{0,1\}^{dn}$, and for any key $K \in \mathcal{K}$ and any tweak $T \in \mathcal{T}$, $H_K(T,\cdot)$ is a permutation over $\{0,1\}^{dn}$.

Definition 3. *We say that H is ϵ-AU_e (Almost Universal for Even indices) if, for any (T,x) and (T',x') such that $(T,x) \neq (T',x')$, we have $\Pr(K \stackrel{R}{\leftarrow} \mathcal{K}; y_i = y'_i \text{ for some } i \in e[0,d)) \leq \epsilon$. Similarly, we say H is ϵ-AU_o (Almost Universal for Odd indices) if $\Pr(K \stackrel{R}{\leftarrow} \mathcal{K}; y_i = y'_i \text{ for some } i \in o[0,d)) \leq \epsilon$.*

In other words, $\Pr(K \stackrel{R}{\leftarrow} \mathcal{K}; y_i \neq y'_i \text{ for any } i \in e[0,d)) > 1 - \epsilon$ holds for ϵ-AU_e function H, and $\Pr(K \stackrel{R}{\leftarrow} \mathcal{K}; y_i \neq y'_i \text{ for any } i \in o[0,d)) > 1 - \epsilon$ holds for ϵ-AU_o function H.

We also define the following universal hash function which is a weaker version of the above definition.

Definition 4. *H is ϵ-AWU (Almost Weakly Universal) if, for any (T, x) and (T', x') such that $(T, x) \neq (T', x')$, $\Pr(K \xleftarrow{R} \mathcal{K}; y_i = y'_i$ for all $i \in [0, d)) \leq \epsilon$.*

In other words, $\Pr(K \xleftarrow{R} \mathcal{K}; y_i \neq y'_i$ for some $i \in [0, d)) > 1 - \epsilon$ holds for ϵ-AWU function H. These hash functions will be useful in simplifying the statements of our results and the security proofs.

4 Tweakable Blockciphers with CPA Security

In this section, we describe constructions of TBCs based on the GFS. We start with the following lemma, which shows that constructing ϵ-AU_e hash function H is enough to construct a TPRP based on the GFS.

Lemma 3. *Let $H : \mathcal{K} \times \mathcal{T} \times \{0,1\}^{dn} \to \{0,1\}^{dn}$ be any ϵ-AU_e function, and let $E = RF \circ RF \circ H$. Then $\mathbf{Adv}^{\mathrm{tprp}}_E(q) \leq 2q^2\epsilon$.*

Intuitively, H ensures that the inputs of F-functions in the second last RF are all distinct. This implies that, if we let $(x_{R,0}, x_{R,1}, \ldots, x_{R,d-1})$ be the output of $RF \circ RF \circ H$, then $x_{R,i}$ for $i \in o[0, d)$ are truly random n-bit strings. Similarly, since $x_{R,i}$ for $i \in o[0, d)$ are random strings, we rarely have collisions at the inputs of F-functions in the last RF, and thus $x_{R,i}$ for $i \in e[0, d)$ are random. A proof is similar to the proof of [20, Theorem 3.1] and hence omitted.

The next lemma shows that an ϵ-AU_e hash function can be constructed from the $(d-1)$-round GFS.

Lemma 4. *For any $a, b \in e[0, d)$ such that $a \neq b$, $H = GFS_{d-1,d}(L_{2,a} + L_{2,b})$ is ϵ-AU_e for $\epsilon = d^2/2^{n+1}$.*

A proof is based on the counting argument. We derive the lower bound on the number of $f_{r,i}$'s such that $x_{d-1,j} \neq x'_{d-1,j}$ holds for all $j \in e[0, d)$, by constructing a series of pseudocodes. A complete proof is given in the next section.

We now present our construction of a TPRP.

Theorem 1. *Let $E = GFS_{d+1,d}(L_{2,a} + L_{2,b})$, where $a, b \in e[0, d)$ and $a \neq b$. Then $\mathbf{Adv}^{\mathrm{tprp}}_E(q) \leq q^2 d^2/2^n$.*

Proof. We see that E can be seen as $E = RF \circ RF \circ GFS_{d-1,d}(L_{2,a} + L_{2,b})$. Since we know that $GFS_{d-1,d}(L_{2,a} + L_{2,b})$ is ϵ-AU_e for $\epsilon = d^2/2^{n+1}$ from Lemma 4, by substituting $\epsilon = d^2/2^{n+1}$ in Lemma 3, we obtain the result. □

For example, $GFS_{5,4}(L_{2,a} + L_{2,b})$ and $GFS_{9,8}(L_{2,a} + L_{2,b})$ are TPRPs.

5 Proof of Lemma 4

$H = GFS_{d-1,d}(L_{2,a} + L_{2,b})$ takes $\{f_{r,j} \mid r \in [0, d-1)$ and $j \in [0, d/2)\}$ as a key. We consider two inputs $(T, x_0), (T', x'_0) \in \{0,1\}^n \times (\{0,1\}^n)^d$ such that $(T, x_0) \neq (T', x'_0)$, $x_0 = (x_{0,0}, \ldots, x_{0,d-1})$, and $x'_0 = (x'_{0,0}, \ldots, x'_{0,d-1})$. Now let

```
000  for j = d/2 − 2 downto 0
010      for r = 0 to d − 2j − 2
020          fix f_{r,j} arbitrarily
030  fix f_{0,d/2−1} s.t. x_{d−j−1,j} ≠ x'_{d−j−1,j} for all j ∈ e[0, d)
040  if d − 2 ∈ {a, b}
050      then fix f_{1,d/2−1} s.t. x_{2,d−2} ⊕ T ≠ x'_{2,d−2} ⊕ T'
060      else fix f_{1,d/2−1} s.t. x_{2,d−2} ≠ x'_{2,d−2}
070  for r = 2 to d − 2
080      fix f_{r,d/2−1} s.t. x_{r+1,d−2} ≠ x'_{r+1,d−2}
090  for j = d/2 − 2 downto 0
100      for r = d − 2j − 1 to d − 2
110          fix f_{r,j} s.t. x_{r+1,2j} ≠ x'_{r+1,2j}
```

Fig. 3. Pseudocode for the Case $x_{0,d-2} \neq x'_{0,d-2}$

$x_{d-1} = (x_{d-1,0}, \ldots, x_{d-1,d-1}), x'_{d-1} = (x'_{d-1,0}, \ldots, x'_{d-1,d-1}) \in (\{0,1\}^n)^d$ be the corresponding outputs. The proof is based on the counting argument, where we derive the lower bound on the number of $f_{r,i}$'s such that $x_{d-1,j} \neq x'_{d-1,j}$ holds for all $j \in e[0,d)$. We consider three cases depending on the values of (T, x_0) and (T', x'_0); Case $x_{0,i} \neq x'_{0,i}$ for some $i \in e[0,d)$, Case $x_{0,i} = x'_{0,i}$ for all $i \in e[0,d)$ and $x_{0,j} \neq x'_{0,j}$ for some $j \in o[0,d)$, and Case $x_0 = x'_0$ and $T \neq T'$.

We first consider the Case $x_{0,i} \neq x'_{0,i}$ for some $i \in e[0,d)$. Without loss of generality, we assume $i = d-2$, i.e., $x_{0,d-2} \neq x'_{0,d-2}$, since other cases are the rotation of this case. We fix $f_{r,j}$'s according to the pseudocode given in Fig. 3.

The pseudocode was designed to meet the following requirements;

- We fix $f_{r,j}$'s to meet some condition, and its inputs and other values related to the condition are already fixed so that the condition can be satisfied. For example, if we want to satisfy $f_{r,j}(x) \oplus x' \neq f_{r,j}(y) \oplus y'$, then x, x', y, y' are all fixed constants such that $x \neq y$.
- When we fix $f_{r,j}$, its condition (e.g., $x_{d-j-1,j} \neq x'_{d-j-1,j}$ for $j \in e[0,d)$ in line 030) can be satisfied solely on $f_{r,j}$, and no other F-functions.
- $x_{d-1,j} \neq x'_{d-1,j}$ holds for all $j \in e[0,d)$.

We now count the number of $f_{r,j}$'s that can be fixed in the pseudocode. Let N_f be the number of all functions over $\{0,1\}^n$, i.e., $N_f = \#\mathrm{Func}(n) = (2^n)^{2^n}$.

In lines 000–020, we have $N_f^{3+5+\cdots+(d-1)} = N_f^{(d-2)(d+2)/4}$ choices of $f_{r,j}$'s since they can be any functions.

In line 030, since we have at most $d/2$ conditions to satisfy, we have at least $N_f(1 - d/2^{n+1})$ choices of $f_{0,d/2-1}$. To see this, we know that the inputs of $f_{0,d/2-1}$ satisfy $x_{0,d-2} \neq x'_{0,d-2}$. Now for any constant $z \in \{0,1\}^n$, we have $N_f/2^n$ choices of $f_{0,d/2-1}$ that satisfy $f_{0,d/2-1}(x_{0,d-2}) \oplus f_{0,d/2-1}(x'_{0,d-2}) = z$. The $d/2$ conditions in line 030 can be seen as if we have at most $d/2$ *bad* values of z, and thus we have at most $dN_f/2^{n+1}$ bad choices of $f_{0,d/2-1}$.

In lines 040–060, we know that the corresponding inputs of $f_{1,d/2-1}$ satisfy $x_{1,d-2} \neq x'_{1,d-2}$ from line 030. Thus in both cases, we have $N_f(1-1/2^n)$ choices of $f_{1,d/2-1}$ that satisfy the desired condition.

```
000 fix f_{0,d/2-1} arbitrarily
010 for j = d/2 − 2 downto 0
020     for r = 0 to d − 2j − 3
030         fix f_{r,j} arbitrarily
040     fix f_{d−2j−2,j} s.t. x_{d−2j−1,2j} ≠ x'_{d−2j−1,2j}
050 if d − 2 ∈ {a, b}
060     then fix f_{1,d/2−1} s.t. x_{2,d−2} ⊕ T ≠ x'_{2,d−2} ⊕ T'
070     else fix f_{1,d/2−1} s.t. x_{2,d−2} ≠ x'_{2,d−2}
080 for r = 2 to d − 2
090     fix f_{r,d/2−1} s.t. x_{r+1,d−2} ≠ x'_{r+1,d−2}
100 for j = d/2 − 2 downto 0
110     for r = d − 2j − 1 to d − 2
120         fix f_{r,j} s.t. x_{r+1,2j} ≠ x'_{r+1,2j}
```

Fig. 4. Pseudocode for the Case $x_{0,i} = x'_{0,i}$ for all $i \in e[0,d)$ and $x_{0,d-1} \neq x'_{0,d-1}$

In lines 070–080, since the inputs are fixed to be distinct when we choose $f_{r,d/2-1}$, we have $N_f(1-1/2^n)$ choices of $f_{r,d/2-1}$ for each $r \in [2,d-2]$. This implies we have $N_f^{d-3}(1-1/2^n)^{d-3} \geq N_f^{d-3}(1-(d-3)/2^n)$ choices of $f_{2,d/2-1},\ldots,f_{d-2,d/2-1}$.

With the similar reasoning, since we fix $2+4+\cdots+(d-4) = (d-4)(d-2)/4$ $f_{r,j}$'s in lines 090–110, we have $N_f^{(d-4)(d-2)/4}(1-(d-4)(d-2)/2^{n+2})$ choices.

Now $\Pr(f_{r,j} \xleftarrow{R} \mathrm{Func}(n); x_{d-1,j} \neq x'_{d-1,j}$ for all $j \in e[0,d))$ is at least

$$\left(1-\frac{d}{2^{n+1}}\right)\left(1-\frac{1}{2^n}\right)\left(1-\frac{d-3}{2^n}\right)\left(1-\frac{(d-4)(d-2)}{2^{n+2}}\right) \geq 1-\frac{d^2}{2^{n+2}} \quad (1)$$

in this case.

We next consider the Case $x_{0,i} = x'_{0,i}$ for all $i \in e[0,d)$ and $x_{0,j} \neq x'_{0,j}$ for some $j \in o[0,d)$. Without loss of generality, we assume $j = d-1$, i.e, $x_{0,d-1} \neq x'_{0,d-1}$. As in the previous case, we fix $f_{r,j}$'s according to the pseudocode given in Fig. 4. By counting the number of $f_{r,j}$'s that can be fixed in the pseudocode, we see that $\Pr(f_{r,j} \xleftarrow{R} \mathrm{Func}(n); x_{d-1,j} \neq x'_{d-1,j}$ for all $j \in e[0,d))$ is at least

$$\left(1-\frac{d-2}{2^{n+1}}\right)\left(1-\frac{1}{2^n}\right)\left(1-\frac{d-3}{2^n}\right)\left(1-\frac{(d-4)(d-2)}{2^{n+2}}\right) \geq 1-\frac{d^2-4}{2^{n+2}}. \quad (2)$$

We next consider the last case $x_0 = x'_0$ and $T \neq T'$. Without loss of generality, we assume $a \in e[0,d-2)$ and $b = d-2$, and use the pseudocode in Fig. 5.

Note that we have $(x_{2,0} \oplus x'_{2,0},\ldots,x_{2,d-1} \oplus x'_{2,d-1}) = (0^n,\ldots,0^n)$ after the line 010. The pseudocode can be seen as using the pseudocode in Fig 3 twice; one for $x_{2,j}$ and $x'_{2,j}$ with $j \in [0,a-2]$, and the other for $j \in [a+2,d-4]$. Now all the $f_{r,j}$'s in lines 000–070 are fixed arbitrarily, and we only have conditions after the line 080. For lines 080–100, we have $(d-3)+(d-5)+\cdots+(d-a-3) = (2d-a-6)(a+2)/4$ conditions, and we have $(d-3)+(d-5)+\cdots+(a+1) = (d+a-2)(d-a-2)/4$ conditions in lines 110–130. Therefore, we have at most

```
000 for j = 0 to d/2 − 1
010     fix f_{0,j} and f_{1,j} arbitrarily
020 for j = 0 to a/2 − 1
030     for r = 2 to a − 2j + 1
040         fix f_{r,j} arbitrarily
050 for j = a/2 + 1 to d/2 − 2
060     for r = 2 to d − 2j − 1
070         fix f_{r,j} arbitrarily
080 for j = a/2 downto 0
090     for r = a − 2j + 2 to d − 2
100         fix f_{r,j} s.t. x_{r+1,2j} ≠ x'_{r+1,2j}
110 for j = d/2 − 1 downto a/2 + 1
120     for r = d − 2j to d − 2
130         fix f_{r,j} s.t. x_{r+1,2j} ≠ x'_{r+1,2j}
```

Fig. 5. Pseudocode for the Case $x_0 = x'_0$ and $T \neq T'$

$((2d-a-6)(a+2)+(d+a-2)(d-a-2))/4$ conditions, which is at most $3d^2/8 \leq d^2/2$ by taking the maximum over $a \in e[0, d-2)$. We conclude that the number of $f_{r,j}$'s that can be fixed in the pseudocode is at least $N_f^{d(d-1)/2}(1-1/2^n)^{d^2/2}$, and thus $\Pr(f_{r,j} \xleftarrow{R} \mathrm{Func}(n); x_{d-1,j} \neq x'_{d-1,j}$ for all $j \in e[0,d))$ is at least

$$\left(1 - \frac{1}{2^n}\right)^{d^2/2} \geq 1 - \frac{d^2}{2^{n+1}}. \tag{3}$$

Finally, we conclude the proof by taking the minimum of (1), (2), and (3). □

6 Tweakable Blockciphers with CCA Security

In this section, we consider the construction of TSPRPs. We first present the following lemma, which shows that constructing ϵ-AU_e hash function H_1 and ϵ-AU_o hash function H_2 is enough to construct a TSPRP based on the GFS, where for a fixed key and a tweak, H_2 is a permutation over $(\{0,1\}^d)^n$ and its inverse is denoted H_2^{-1}.

Lemma 5. *Let $H_1 : \mathcal{K} \times \mathcal{T} \times \{0,1\}^{dn} \to \{0,1\}^{dn}$ be any ϵ-AU_e function, $H_2 : \mathcal{K} \times \mathcal{T} \times \{0,1\}^{dn} \to \{0,1\}^{dn}$ be any ϵ-AU_o function, and let $E = H_2^{-1} \circ RF \circ RF \circ H_1$, where the tweak space of E is $\mathcal{T}$, and we use the same tweak for both H_1 and H_2. Then $\mathbf{Adv}_E^{\mathrm{tsprp}}(q) \leq 2q^2\epsilon$.*

A proof is similar to the proof of [20, Theorem 3.2] by adopting the tweak, and hence omitted. The next lemma shows the construction of an ϵ-AU_o hash from the decryption of the $(d-1)$-round GFS.

Lemma 6. *For any $a, b \in e[0,d)$ such that $a \neq b$, $H = GFS_{d-1,d}^{-1}(L_{d-2,a} + L_{d-2,b})$ is ϵ-AU_o for $\epsilon = d^2/2^{n+1}$.*

A proof is very similar to the proof of Lemma 4 and omitted. We now present our construction of a TSPRP.

Theorem 2. *Let $a, b, a', b' \in e[0, d)$, where $a \neq b$ and $a' \neq b'$, and let $E = GFS_{2d,d}(L_{2,a} + L_{2,b} + L_{2d-1,a'} + L_{2d-1,b'})$. Then $\mathbf{Adv}_E^{\mathrm{tsprp}}(q) \leq q^2 d^2 / 2^n$.*

Proof. Since E can be seen as $E = GFS_{d-1,d}(L_{d-2,a'} + L_{d-2,b'}) \circ RF \circ RF \circ GFS_{d-1,d}(L_{2,a} + L_{2,b})$, from Lemma 5 and Lemma 6, we obtain the result. □

For example, $GFS_{8,4}(L_{2,a}+L_{2,b}+L_{7,a'}+L_{7,b'})$ and $GFS_{16,8}(L_{2,a}+L_{2,b}+L_{15,a'}+L_{15,b'})$ are TSPRPs.

7 How to Input Multiple Tweaks

So far, all tweaks were assumed to be one data line in length. It may be desirable however, to have longer tweaks. In this section, we present such constructions.

7.1 Multiple Tweaks without Increasing the Number of Rounds

We first consider constructions without increasing the number of rounds.

For $r \geq 2$, let $\mathcal{L}_r = (L_{r,0}, L_{r,2}, \ldots, L_{r,d-2})$ be the locations. In this section, we are particularly interested in the case $r = 2$, i.e., $\mathcal{L}_2 = (L_{2,0}, L_{2,2}, \ldots, L_{2,d-2})$.

Let $T = (T_1, T_2, \ldots, T_\tau)$ be a τn-bit tweak. Let $\mathcal{C}$ be a $\tau \times d/2$ matrix with $0/1$ elements. We consider to XOR the i-th element of $T \cdot \mathcal{C}$ to the i-th element of $\mathcal{L}_2$. That is, we consider $\mathcal{T}_2 = (T^{L_{2,0}}, T^{L_{2,2}}, \ldots, T^{L_{2,d-2}}) = T \cdot \mathcal{C}$, or, equivalently, we consider $\mathcal{L}_2 \cdot \mathcal{C}^t \subseteq \Lambda_R$.

For example, if $d = 8$, $\tau = 2$, and

$$\mathcal{C} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix},$$

then $T = (T_1, T_2)$, and $T^{L_{2,0}} = T_1, T^{L_{2,2}} = T_1 \oplus T_2, T^{L_{2,4}} = T_2, T^{L_{2,6}} = 0^n$, or equivalently, $\mathcal{L}_2 \cdot \mathcal{C}^t = (L_{2,0} + L_{2,2}, L_{2,2} + L_{2,4})$.

We have the following lemma.

Lemma 7. *Let $\mathcal{C}$ be a $\tau \times d/2$ matrix, and consider $\Lambda = \mathcal{L}_2 \cdot \mathcal{C}^t$. Suppose that the following condition holds; For any distinct tweaks $T, T' \in (\{0,1\}^n)^\tau$, there are at least two distinct indices $i, j \in e[0, d)$ such that $T^{L_{2,i}} \neq T'^{L_{2,i}}$ and $T^{L_{2,j}} \neq T'^{L_{2,j}}$, where $(T^{L_{2,0}}, T^{L_{2,2}}, \ldots, T^{L_{2,d-2}}) = T \cdot \mathcal{C}$ and $(T'^{L_{2,0}}, T'^{L_{2,2}}, \ldots, T'^{L_{2,d-2}}) = T' \cdot \mathcal{C}$. Then, $GFS_{d-1,d}(\Lambda)$ is ϵ-AU_e for $\epsilon = d^2/2^{n+1}$.*

This lemma can be proved similarly to Lemma 4. We next show that a linear code can be used as $\mathcal{C}$ in Lemma 7.

Lemma 8. *Let $\mathcal{G}$ be a generator matrix of a $(d/2, d/2 - 1, 2)$ linear code, and let $\Lambda = \mathcal{L}_2 \cdot \mathcal{G}^t$. Then $GFS_{d-1,d}(\Lambda)$ is ϵ-AU_e for $\epsilon = d^2/2^{n+1}$, where it takes a $(d/2 - 1)n$-bit tweak.*

Proof. Consider two distinct tweaks $T, T' \in (\{0,1\}^n)^{d/2-1}$, and let $\mathcal{T}_2 = T \cdot \mathcal{G}$ and $\mathcal{T}_2' = T' \cdot \mathcal{G}$. Since $\mathcal{G}$ is a $(d/2-1) \times d/2$ matrix with minimum Hamming distance 2, we have at least two nonzero elements in $\mathcal{T}_2 \oplus \mathcal{T}_2' = (T \oplus T') \cdot \mathcal{G}$. Therefore, the condition in Lemma 7 is satisfied, and we obtain the result. □

We next show our construction of a TPRP based on Lemma 8.

Theorem 3. *Let $\mathcal{G}$ be a generator matrix of a $(d/2, d/2-1, 2)$ linear code, and let $\Lambda = \mathcal{L}_2 \cdot \mathcal{G}^t$. Then for $E = GFS_{d+1,d}(\Lambda)$, we have $\mathbf{Adv}_E^{\mathrm{tprp}}(q) \le q^2 d^2 / 2^n$.*

Proof. We see that $E = RF \circ RF \circ GFS_{d-1,d}(\Lambda)$. Since $GFS_{d-1,d}(\Lambda)$ is ϵ-AU_e for $\epsilon = d^2/2^{n+1}$ from Lemma 8, by using Lemma 3, we obtain the result. □

By following the similar argument to Sec. 6, we also obtain a TSPRP.

Theorem 4. *Let $\mathcal{G}$ be a generator matrix of a $(d/2, d/2-1, 2)$ linear code, and let $\Lambda = (\mathcal{L}_2 + \mathcal{L}_{2d-1}) \cdot \mathcal{G}^t$. Then for $E = GFS_{2d,d}(\Lambda)$, $\mathbf{Adv}_E^{\mathrm{tsprp}}(q) \le q^2 d^2 / 2^n$.*

For example, $GFS_{9,8}(L_{2,0} + L_{2,2}, L_{2,2} + L_{2,4}, L_{2,4} + L_{2,6})$ is a TPRP, and $GFS_{16,8}(L_{2,0} + L_{2,2} + L_{15,0} + L_{15,2}, L_{2,2} + L_{2,4} + L_{15,2} + L_{15,4}, L_{2,4} + L_{2,6} + L_{15,4} + L_{15,6})$ is a TSPRP, where they take a $3n$-bit tweak, and $\mathcal{G}$ is given by

$$\mathcal{G} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

for both cases.

7.2 Multiple Tweaks with Increasing the Number of Rounds

We finally consider constructions by increasing the number of rounds. We show that our construction can take additional $dn/2$-bit tweak as its input by adding one round.

The following lemma shows a construction of an ϵ-*AWU* function.

Lemma 9. *Let $l \ge 2$ be an integer. Then $GFS_{l,d}(\mathcal{L}_2, \mathcal{L}_3, \ldots, \mathcal{L}_l)$ is ϵ-AWU for $\epsilon = l/2^n$, and it takes a $(l-1)dn/2$-bit tweak.*

This can be proved similarly to the proof of Lemma 4 and Lemma 7, and hence the proof is omitted.

We next show that the composition of an ϵ_1-*AWU* function and an ϵ_2-AU_e function yields an $(\epsilon_1 + \epsilon_2)$-$AU_e$ function.

Lemma 10. *Let H_1 be any ϵ_1-AWU function with a tweak space $\mathcal{T}_1$, and H_2 be any ϵ_2-AU_e function with a tweak space $\mathcal{T}_2$. Then, $H_2 \circ H_1$ is an ϵ-AU_e for $\epsilon = \epsilon_1 + \epsilon_2$ with a tweak space $\mathcal{T}_1 \times \mathcal{T}_2$.*

Proof. We first fix some terminology. Let $(T_1, X) \in \mathcal{T}_1 \times \{0,1\}^{dn}$ be the input of H_1, and $Z \in \{0,1\}^{dn}$ be its output. Similarly, let $(T_2, Z) \in \mathcal{T}_2 \times \{0,1\}^{dn}$ be the input of H_2, and $Y = (y_0, \ldots, y_{d-1}) \in \{0,1\}^{dn}$ be its output. That is, $H_2 \circ H_1$ takes $((T_1, T_2), X)$ as input, and outputs Y. Let $((T_1, T_2), X)$ and $((T_1', T_2'), X')$ be distinct inputs. We consider two cases; Case $X \neq X'$ or $T_1 \neq T_1'$, and Case $X = X'$, $T_1 = T_1'$, and $T_2 \neq T_2'$.

For the first case, from Lemma 9, we have $Z \neq Z'$ with probability $1 - \epsilon_1$. Then $\Pr(y_i \neq y_i' \text{ for any } i \in e[0,d))$ is at least $(1-\epsilon_1)(1-\epsilon_2) \geq 1 - (\epsilon_1 + \epsilon_2)$.

For the second case, $\Pr(y_i \neq y_i' \text{ for any } i \in e[0,d))$ is at least $1 - \epsilon_2$, and we have the claimed bound. □

We next show our TPRP based on Lemma 8, Lemma 9, and Lemma 10.

Theorem 5. *Let $l \geq 2$ be an integer, and let $\mathcal{G}$ be a generator matrix of a $(d/2, d/2-1, 2)$ linear code. Then for $E = GFS_{l+d+1,d}(\mathcal{L}_2, \mathcal{L}_3, \ldots, \mathcal{L}_l, \mathcal{L}_{l+2} \cdot \mathcal{G}^t)$,*

$$\mathbf{Adv}_E^{\mathrm{tprp}}(q) \leq \frac{q^2 d^2}{2^n} + \frac{2lq^2}{2^n},$$

where it takes a $(ld/2-1)n$-bit tweak.

Proof. We see that $E = RF \circ RF \circ GFS_{l+d-1,d}(\mathcal{L}_2, \mathcal{L}_3, \ldots, \mathcal{L}_l, \mathcal{L}_{l+2} \cdot \mathcal{G}^t)$. From Lemma 8, Lemma 9, and Lemma 10, $GFS_{l+d-1,d}(\mathcal{L}_2, \mathcal{L}_3, \ldots, \mathcal{L}_l, \mathcal{L}_{l+2} \cdot \mathcal{G}^t)$ is ϵ-AU_e for $\epsilon = (d^2 + 2l)/2^{n+1}$. From Lemma 3, we obtain the result. □

We also obtain the following construction of a TSPRP, which can be proved by following the similar argument to Sec. 6.

Theorem 6. *Let $l \geq 2$ be an integer, and let $\mathcal{G}$ be a generator matrix of a $(d/2, d/2-1, 2)$ linear code. Then for $E = GFS_{2d+2l,d}(\mathcal{L}_2 + \mathcal{L}_{2d+2l-1}, \mathcal{L}_3 + \mathcal{L}_{2d+2l-2}, \ldots, \mathcal{L}_l + \mathcal{L}_{2d+l+1}, (\mathcal{L}_{l+2} + \mathcal{L}_{2d+l-1}) \cdot \mathcal{G}^t)$, we have*

$$\mathbf{Adv}_E^{\mathrm{tprp}}(q) \leq \frac{q^2 d^2}{2^n} + \frac{2lq^2}{2^n},$$

where it takes a $(ld/2-1)n$-bit tweak.

8 Conclusions

In this paper, we considered the problem of constructing TPRPs and TSPRPs from the GFS. Our basic construction of a TPRP in Theorem 1 shows that $(d+1)$ rounds are enough, and a TSPRP in Theorem 2 shows that $2d$ rounds are enough. We also studied how we can handle multiple tweaks with/without increasing the number of rounds and still maintain the provable security. It would be interesting to see designs based on other structures, including the SPN (Substitution-Permutation Network) of the AES.

References

1. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
2. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. JCSS 18, 265–278 (1979)
3. Chakraborty, D., Sarkar, P.: HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 287–302. Springer, Heidelberg (2006)
4. Chakraborty, D., Sarkar, P.: A new mode of encryption providing a tweakable strong pseudo-random permutation. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 293–309. Springer, Heidelberg (2006)
5. Crowley, P.: Mercy: A fast large block cipher for disk sector encryption. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 49–63. Springer, Heidelberg (2001)
6. Goldenberg, D., Hohenberger, S., Liskov, M., Schwartz, E.C., Seyalioglu, H.: On tweaking Luby-Rackoff blockciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 342–356. Springer, Heidelberg (2007)
7. Feistel, H.: Cryptography and computer privacy. Scientific American, 15–23 (1973)
8. Fluhrer, S.R.: Cryptanalysis of the Mercy block cipher. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 21–40. Springer, Heidelberg (2002)
9. Halevi, S.: EME*: Extending EME to handle arbitrary-length messages with associated data. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 315–327. Springer, Heidelberg (2004)
10. Halevi, S.: Invertible universal hashing and the TET encryption mode. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 412–429. Springer, Heidelberg (2007)
11. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
12. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 292–304. Springer, Heidelberg (2004)
13. D'Halluin, C., Bijnens, G., Preneel, B., Rijmen, V.: Equivalent keys of HPC. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 29–42. Springer, Heidelberg (1999)
14. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
15. Liskov, M., Rivest, R., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
16. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. on Computing 17(2), 373–386 (1988)
17. Minematsu, K.: Improved security analysis of XEX and LRW modes. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 96–113. Springer, Heidelberg (2007)
18. Minematsu, K., Matsushima, T.: Tweakable enciphering schemes from hash-sum-expansion. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 252–267. Springer, Heidelberg (2007)
19. Moriai, S., Vaudenay, S.: On the pseudorandomness of top-level schemes of block ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 289–302. Springer, Heidelberg (2000)

20. Naor, M., Reingold, O.: On the construction of pseudorandom permutations: Luby-Rackoff revisited. J. Cryptology 12(1), 29–66 (1999)
21. Rivest, R.L., Robshaw, M.J.B., Sidney, R., Yin, Y.L.: The RC6 block cipher. Submission to AES (1998), http://www.rsa.com/
22. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to mode OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
23. Sarkar, P.: Improving upon the TET mode of operation. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 180–192. Springer, Heidelberg (2007)
24. Schroeppel, R.: The Hasty Pudding Cipher. NIST AES proposal (1998), http://www.cs.arizona.edu/~rcs/hpc
25. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
26. Wagner, D.: Equivalent keys in HPC. Presentation at the rump session of AES2, Rome (1999)
27. Wang, P., Feng, D., Wu, W.: HCTR: A variable-input-length enciphering mode. In: Feng, D., Lin, D., Yung, M. (eds.) CISC 2005. LNCS, vol. 3822, pp. 175–188. Springer, Heidelberg (2005)
28. Zheng, Y., Matsumoto, T., Imai, H.: On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 461–480. Springer, Heidelberg (1990)

A Negative Results on Tweakable Blockciphers with CPA Security

In this appendix, we present negative results on the constructions of TBCs based on the GFS.

In the following lemma, we show that $GFS_{R,d}$ cannot be a TPRP if the tweaks are located in the last $(d-1)$ rounds in a triangular way. Let $\Sigma_{r,i}$ be any subset of $\{L_{r,j} \mid j \in e[i,d)\}$.

Lemma 11. *Let $R \geq d-1$. Then $GFS_{R,d}(\Sigma_{R-1,0}, \Sigma_{R-2,1}, \ldots, \Sigma_{R-d+1,d-2})$ is not a TPRP.*

Proof. Suppose that $\Sigma_{R-d+1,d-2} \neq \emptyset$ and T is XOR-ed at location $L_{R-d+1,d-2}$. Consider two inputs $(T, x) \in \mathcal{T} \times (\{0,1\}^n)^d$ and $(T', x) \in \mathcal{T} \times (\{0,1\}^n)^d$, where $T \neq T'$ and other tweaks (if any) are all fixed to 0^n. Let $(x_{R,0}, x_{R,1}, \ldots, x_{R,d-1})$ and $(x'_{R,0}, x'_{R,1}, \ldots, x'_{R,d-1})$ be the corresponding ciphertexts. Then, it is easy to verify that we always have $x_{R,d-1} \oplus x'_{R,d-1} = T \oplus T'$, which allows trivial distinguishing attack. It is also easy to verify that a similar attack can be applied for other locations. □

We next show that, for any tweak location, d rounds are not enough to construct a TPRP.

Lemma 12. *For any $\Lambda \subseteq \Lambda_d$, $GFS_{d,d}(\Lambda)$ is not a TPRP.*

Proof. If we set all tweaks to 0^n, $GFS_{d,d}(\Lambda)$ is equivalent to $GFS_{d,d}$. Now it is easy to see that d-round GFS with d data lines is not a PRP. Consider two plaintexts $x = (x_{0,0}, x_{0,1}, \ldots, x_{0,d-1})$ and $x' = (x'_{0,0}, x'_{0,1}, \ldots, x'_{0,d-1})$, where $x_{0,i} = x'_{0,i}$ for all $i \in [0, d-1)$ but $x_{0,d-1} \neq x'_{0,d-1}$. We see that the corresponding ciphertexts $(x_{d,0}, x_{d,1}, \ldots, x_{d,d-1})$ and $(x'_{d,0}, x'_{d,1}, \ldots, x'_{d,d-1})$ always satisfy $x_{d,d-1} \oplus x'_{d,d-1} = x_{0,d-1} \oplus x'_{0,d-1}$. □

Timed-Release Encryption Revisited

Sherman S.M. Chow[1] and S.M. Yiu[2]

[1] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu
[2] Department of Computer Science
University of Hong Kong
Pokfulam, Hong Kong
smyiu@cs.hku.hk

Abstract. Timed-release encryption (TRE) is a two-factor encryption scheme combining public key encryption and time-dependent encryption – decryption requires a trapdoor which is kept confidential by a time-server until at an appointed time. This paper revisits two recent results.

In ESORICS 2007, Chalkias *et al.* proposed an efficient anonymous TRE scheme. Unfortunately, we show the security threats of their scheme in the presence of a curious time-server and an impatient recipient.

Recently, Chow *et al.* proposed an encryption scheme in the standard model which can be used as TRE. Nevertheless, only confidentiality is guaranteed. We demonstrate how to support pre-open capability, which is often desirable in applications of TRE. Our extension also enables only the recipient to know the release-time from the ciphertext. This feature is not considered in previous notion of release-time confidentiality.

Keywords: Timed-release encryption, release-time confidentiality.

1 Introduction

This paper considers the problem of sending information into the future, i.e. encrypting a message so that it cannot be decrypted before a future time chosen by the sender, and after that time only the designated recipient can decrypt it. This can be generally resolved by two means. The first one is to require the intended recipient to invest a significant amount of time to do the computation necessary for the decryption; however, this approach is very computationally expensive, and the release-time varies with the computing power of the recipient, thus is not precisely controllable. Instead of relying on the computational effort of the recipients, the second one depends on their knowledge, such that a special piece of information is withheld from the recipient until the designated time. It can be trivially done if the senders are permanently online, instead we assign this job to a trusted third party, which is also known as time server. We want a solution that does not require too much help from the time server. Ideally, the time server only publishes some system parameters but does not interact

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 38–51, 2008.

with each sender; and for each time a certain (set of) ciphertext should become decryptable, it only publishes a single time-dependent trapdoor but not anything specific to each ciphertext. Cryptographic techniques make such a timed-release encryption (TRE) system possible. The applications of TRE can be broadly classified into two categories.

Rapid Dissemination of Information. The size of the time-dependent trapdoor is small compared with the ciphertext (even of text message, due to the inherent ciphertext expansion in probabilistic encryption). With TRE, one can send the bulky ciphertext beforehand, without worrying the leakage of the confidential information. When it should be made public, a small trapdoor can be made available to a potentially large set of recipients. This avoids the problem of any network impedance at the release time. Examples of applicable scenarios are abundant, such as strategic business plans, news agencies timed publications, licensed software updates, scheduled payments, or "casual" applications like internet contests, where participants should not see the challenge before the designated time.

Commitment of Confidential Information. Commitment of confidential information is needed in many scenarios, such as sealed-bid auction, electronic lotteries, legal will, certified e-mail [18], *etc.* One can view the ciphertext as a kind of commitment made by the sender. In TRE, the decryption algorithm deterministically recovers the message from the ciphertext by a time-dependent trapdoor and the user's private key. Once the ciphertext is sent, there is no chance for the message sender to change the message that will be obtained from the decryption by the recipient later.

A special class of TRE scheme supports pre-open capability [14,18], which means that the sender can help the recipient to decrypt the ciphertext by publishing a pre-open key. Since the pre-open key is given by the sender, it may give an opportunity to the sender to somehow control what message will be output by the decryption algorithm by manipulating the pre-open key. Using a TRE with pre-open capability as a way to commit some confidential information requires the TRE scheme to be binding.

1.1 Related Work

The concept of TRE was suggested by May [19] in 1993, and other timed-release cryptographic protocols were later proposed [2,4,15,16]. Early TRE schemes [12,20] either require multiple-round interaction with the time-server, or are computationally expensive. The first attempt to construct a non-interactive TRE was made in [3], but it has neither formal security model nor proof. The formal model of confidentiality is later considered independently by Cheon *et al.* [7] and Cathalo *et al.* [5]. The former focuses on authenticated TRE. Ignoring authentication, the latter one is stronger than the implicit non-authenticated version of the former. Cathalo *et al.* [5] also formalizes the release-time confidentiality.

In many applications of TRE, it is desirable to have a pre-open mechanism that the sender can enable the recipient to decrypt the ciphertext before the

pre-specified release-time, without re-sending the plaintext. For TRE with such pre-open capability [18], the sender gets hold of a pre-open key that can functionally substitute the role of the system's time-dependent trapdoor, for the ciphertext prepared by him/her. The concept of pre-open capability is introduced to the TRE paradigm by [18]. However, the scheme of [18] does not consider the security threat that the sender can give a pre-open key which opens the ciphertext to another message that is different from the one originally being encrypted. This deficiency is pointed out by [14], where the property of binding is formally defined and a scheme with binding pre-open key is proposed.

Recently, Chalkias *et al.* proposed an efficient TRE scheme [6] in the random oracle model (ROM). Based on the observation that many TRE schemes are based on certificateless encryption (CLE) mechanism, Chow *et al.* [9] proposed a new CLE model which is general enough to support TRE. They also proposed a concrete hierarchical CLE scheme which can be used as TRE with a hierarchy of time-identifiers, resulting the first provably secure (hierarchical) TRE scheme in the standard model. Moreover, their scheme supports the option of parial decryption by the time-server – instead of releasing a system-wide time-dependent trapdoor, the time-server can partially decrypt a particular ciphertext without leaking any information which helps the decryption of any other ciphertext. This essentially gives the time-server another mode of operation, e.g. it can charge the decryptor for each decryption. However, their generic CLE model omits pre-open capability because it does not make a good sense in the context of CLE. For a survey of major CLE constructions in the literature, one may refer to [11].

Apart from CLE and TRE, another example of two-factor encryption is token-controlled encryption (TCE) [1,8,17], which can be used to substitute TRE in some cases, e.g. when the sender is willing to interact with a trusted agent. Chow [8] proposed a TCE scheme in the standard model. Interested readers are referred to [8] for a survey and the applicabilities of TCE.

1.2 Contributions

In this paper, we investigate a number of aspects in the TRE paradigm.

Attack: Power of Strong Decryption. Chalkias *et al.*'s scheme is claimed to achieve indistinguishability against adaptive chosen ciphertext even if the adversary is entitled to access a strong decryption oracle. Nevertheless, we show attacks against their scheme compromising these properties. While the first attack can only be launched by an impatient recipient with an "imaginary" strong decryption oracle, we show how this vulnerability can be exploited by a curious time-server. We also discuss a possible fix.

Definition: Practical Consideration of Release-time Confidentiality. In the aspect of formalization, we give a new formal definition of release-time confidentiality. Existing schemes either assume the recipient learns the release-time from out-of-band channel, or the time is sent in clear with the ciphertext. We refine the syntax of TRE and give the corresponding security definition so that

only the intended recipient can learn the release-time from the ciphertext, but not any other parties including a curious time-server.

Scheme: Pre-Open Capability. Chow *et al.*'s study focuses on the relationship between the confidentiality of CLE and TRE; other features of TRE are not covered. We demonstrate how to incorporate Chow *et al.*'s (CRR) scheme with pre-open capability and our practice-oriented release-time confidentiality.

2 Security Models of Timed-Release Encryption

2.1 Syntax of Timed-Release Encryption

Definition 1. *A TRE scheme (with pre-open capability) is defined by the following quintuple (sextuple) of probabilistic polynomial time (PPT) algorithms:*

- Setup *(run by the server) is a probabilistic algorithm which takes a security parameter* 1^λ*, outputs a master secret key* Msk*, and the global parameters* Pub*. We assume that* λ *is implicit in* Pub *and all other algorithms take* Pub *implicitly as an input.*
- $\mathsf{Extract}$ *(run by the server) is a possibly probabilistic algorithm which takes the master secret key* Msk *and a string* $T \in \{0,1\}^*$*, outputs a trapdoor key* d_T *associated with the time-identifier* T*.*
- KeyGen *(run by a user) is a probabilistic algorithm which generates a private/public key pair* $(\mathsf{sk}_u, \mathsf{pk}_u)$*.*
- Enc *(run by a sender) is a probabilistic algorithm which takes a message* m *from some implicit message space, an identifier* $T \in \{0,1\}^*$*, and the receiver's public key* pk_u *as input, returns a ciphertext* C *(and its pre-open key* V_C*, if pre-open capability is supported).*
- $\mathsf{PreOpen}$ *(run by a receiver) is a possibly probabilistic algorithm supported by a TRE with pre-open capability, it takes the ciphertext* C*, the receiver's private key* sk_u *and a pre-open key* V_C *as input, returns either the plaintext, an invalid flag* $\perp_V$ *denoting* V_C *is an invalid pre-open key, or an invalid flag* $\perp_C$ *denoting the ciphertext is invalid.*
- Dec *(run by a receiver) is a deterministic algorithm which takes the ciphertext* C*, the receiver's private key* sk_u *and a trapdoor key* d_T *as input, returns either the plaintext or an invalid flag* $\perp_C$*.*

Correctness requires $\mathsf{Dec}(C, \mathsf{sk}, \mathsf{Extract}(\mathsf{Msk}, T)) = \mathsf{PreOpen}(C, \mathsf{sk}, V_C) = m$ *for all* $\lambda \in \mathbb{N}$*, all* $(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*, all* $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}$*, all message* m*, all string* T *in* $\{0,1\}^*$ *and all* $(C, V_C) \xleftarrow{\$} \mathsf{Enc}(m, T, \mathsf{pk})$*.*

2.2 Confidentiality

Following common practice, we consider the following two kinds of adversaries. A Type-I adversary models any coalition of rogue users, and who aims to break the confidentiality of a user's ciphertext. A Type-II adversary that models a curious

time server, who also aims to break the confidentiality of a user's ciphertext. Security against these adversaries are modeled by the experiment below executed between a simulator and an adversary of type $\mathsf{X} \in \{\mathrm{I}, \mathrm{II}\}$, denoting whether an PPT adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{find}}, \mathcal{A}_{\mathsf{guess}})$ is of Type-I or Type-II. The auxiliary information Aux depends on X.

Definition 2. Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-X}}(\lambda)$

$(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(m_0, m_1, \mathsf{pk}^*, T^*, \mathsf{state}) \leftarrow \mathcal{A}_{\mathsf{find}}^{\mathsf{ExtractO}(\cdot),\mathsf{DecO}(\cdot,\cdot,\cdot)}(\mathsf{Pub}, \mathsf{Aux})$
$b \xleftarrow{\$} \{0,1\}$, $C^* \xleftarrow{\$} \mathsf{Enc}(m_b, T^*, \mathsf{pk}^*)$
$b' \leftarrow \mathcal{A}_{\mathsf{guess}}^{\mathsf{ExtractO}(\cdot),\mathsf{DecO}(\cdot,\cdot,\cdot)}(C^*, \mathsf{state})$
If $(|m_0| \neq |m_1|) \vee (b \neq b')$ *then return 0 else return 1*

where $\mathsf{ExtractO}$ *oracle takes a time-identifier* $T \in \{0,1\}^*$ *as input and returns its trapdoor* d_T*; a* DecO *oracle takes a ciphertext* C*, a time-identifier* T*, and a public key* pk*, and outputs* $\mathsf{Dec}(C, \mathsf{sk}, d_T)$ *where* sk *is the secret key that matches* pk*, and* C *may or may not be encrypted under* T *and* pk*.*

Below are the formal definitions of indistinguishability against adaptive chosen-ciphertext attack (IND-CCA), which captures the confidentiality of TRE against these two types of adversary.

Definition 3. *A timed-release encryption scheme is* (t, q_E, q_D, ϵ) *IND-CCA secure against a Type-I adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-I}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_E *extraction queries and* q_D *decryption queries, subjects to the following constraints:*

1. $\mathsf{Aux} = \emptyset$*, i.e. no auxiliary information is given to the adversary.*
2. *No* $\mathsf{ExtractO}(T^*)$ *query throughout the game.*
3. *No* $\mathsf{DecO}(C^*, T^*, \mathsf{pk}^*)$ *query throughout the game.*

Definition 4. *A timed-release encryption scheme is* (t, q_D, ϵ) *IND-CCA secure against a Type-II adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-II}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_D *decryption queries, but no* $\mathsf{DecO}(C^*, T^*, \mathsf{pk}^*)$ *query throughout the game, with* $\mathsf{Aux} = (\mathsf{Msk}, \mathsf{pk}^*)$*, where* $(\mathsf{sk}^*, \mathsf{pk}^*) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ *is executed by the simulator.*

2.3 Pre-open Capability

The addition of pre-open capability affects the security models of confidentiality. In [14], it has been shown that a TRE scheme which is IND-CCA secure against a curious time server implies its IND-CCA security against an outsider attacker (i.e. with neither the master secret key nor the intended recipient's private key). Besides, a Type-I adversary can supply its own public key to be challenged with, which naturally means it knows the corresponding private key and hence it is not entitled to have the pre-open key (or it knows both pieces of secret). So it is sufficient to alter the Type-II adversary model.

Definition 5. Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-II-PO}}(\lambda)$

$(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(\mathsf{sk}^*, \mathsf{pk}^*) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1, \mathsf{pk}^*, T^*, \mathsf{state}) \leftarrow \mathcal{A}_{\mathsf{find}}^{\mathsf{PreOpenO}(\cdot,\cdot,\cdot),\mathsf{DecO}(\cdot,\cdot,\cdot)}(\mathsf{Pub}, \mathsf{Aux} = (\mathsf{Msk}, \mathsf{pk}^*))$
$b \xleftarrow{\$} \{0,1\}$, $(C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Enc}(m_b, T^*, \mathsf{pk}^*)$,
$b' \leftarrow \mathcal{A}_{\mathsf{guess}}^{\mathsf{PreOpenO}(\cdot,\cdot,\cdot),\mathsf{DecO}(\cdot,\cdot,\cdot)}((C^*, V_{C^*}), \mathsf{state})$
If $(|m_0| \neq |m_1|) \vee (b \neq b')$ *then return 0 else return 1*

where PreOpenO *oracle takes a ciphertext* C*, a pre-open key* V_C*, and a public key* pk*, and outputs* $\mathsf{PreOpen}(C, \mathsf{sk}, V_C)$ *where* sk *is the secret key that matches* pk*, and* V_C *may or may not be the pre-open key of* C*;* DecO *oracle is defined as in Definition 2.*

Definition 6. *A timed-release encryption scheme with pre-open capability is* (t, q_D, ϵ) *IND-CCA secure against a Type-II adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-II}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_K *public key queries and* q_D *decryption queries, subjects to the following conditions:*

1. *No* $\mathsf{PreOpenO}(C^*, V_{C^*}, \mathsf{pk}^*)$ *query throughout the game.*
2. *No* $\mathsf{DecO}(C^*, T^*, \mathsf{pk}^*)$ *query throughout the game.*

Definition 7. *A TRE scheme is binding if the following probability is negligible for all PPT algorithm* $\mathcal{A}$*:*

$$\Pr[\ (C^*, T^*, V_{C^*}) \leftarrow \mathcal{A}(\mathsf{Pub})(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$$
$$\wedge\ \mathsf{PreOpen}(C^*, \mathsf{sk}, V_{C^*}) \neq \{\mathsf{Dec}(C^*, \mathsf{sk}, \mathsf{Extract}(\mathsf{Msk}, T^*)), \perp_V, \perp_C\}].$$

2.4 Release-Time Confidentiality

Release-time confidentiality protects the release-time from being known to anyone but the recipient. In order to let the recipient to know the release-time from the ciphertext, we need to add an algorithm called GetTime to the TRE framework, which outputs a time T taking a ciphertext and a secret key as input. Correctness requires $\mathsf{GetTime}(\mathsf{Enc}(m, T, \mathsf{pk}), \mathsf{sk}) = T$ for all $\ell, n \in \mathbb{N}$, all Pub given by $\mathsf{Setup}(1^\ell, n)$, all $(\mathsf{sk}, \mathsf{pk})$ given by KeyGen, all message m, and all identifier T in $\{0,1\}^*$.

The formal security requirement is similar to that in [5], but on top of that we need to add an GetTimeO oracle that takes a ciphertext and a public key of the adversary's choice. Basically, instead of encrypting one of the two messages under the fixed identifier, all given by the adversary, in the challenge ciphertext; a fixed message is encrypted under one of the two identifiers. The adversary's goal is to tell which identifier is randomly chosen by the challenger. Security is only defined against a Type-II adversary. For any Type-I adversary, it can replace the challenge public key pk^*, and hence obtaining T from $\mathsf{GetTime}(C, \mathsf{sk}^*)$ is trivial.

Definition 8. Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{RTC-II}}(\lambda)$

$(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$,
$(m^*, \mathsf{pk}^*, T_0^*, T_1^*, \mathsf{state}) \leftarrow \mathcal{A}_{\mathsf{find}}^{\mathsf{GetTimeO}(\cdot,\cdot,\cdot),\mathsf{DecO}(\cdot,\cdot,\cdot)}(\mathsf{Pub}, \mathsf{Aux} = (\mathsf{Msk}, \mathsf{pk}^*))$
$b \xleftarrow{\$} \{0,1\}, C^* \xleftarrow{\$} \mathsf{Enc}(m^*, T_b^*, \mathsf{pk}^*), b' \leftarrow \mathcal{A}_{\mathsf{guess}}^{\mathsf{GetTimeO}(\cdot,\cdot,\cdot),\mathsf{DecO}(\cdot,\cdot,\cdot)}(C^*, \mathsf{state})$
If $b \neq b'$ *then return 0 else return 1*

Definition 9. *A timed-release encryption scheme is* (t, q_D, ϵ) *RTC-CCA secure against a Type-II adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{RTC-II}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_D *decryption queries, subjects to the following conditions:*

1. *No* $\mathsf{GetTime}(C^*, \mathsf{pk}^*)$ *query throughout the game.*
2. *No* $\mathsf{Dec}(C^*, T^*, \mathsf{pk}^*)$ *query throughout the game, where* $T^* \in \{T_0^*, T_1^*\}$.

3 Analysis of a Recent TRE Scheme in ESORICS '07

3.1 Review

We first review how the ciphertext is constructed in the TRE scheme proposed by Chalkias *et al.* [6].

$\mathsf{Setup}(1^\lambda, 1^{\lambda_0})$: Given security parameters λ and λ_0, where λ_0 is a polynomially-bounded function of λ, generate $\mathbb{G}$ and $\mathbb{G}_T$ which are two multiplicative groups with a bilinear map $\hat{e}$ as defined before. They are of the same order p, which is a prime and $2^\lambda < p < 2^{\lambda+1}$. The public parameters Pub and the master secret key Msk are given by

$$\mathsf{Pub} = (\lambda, \lambda_0, p, \mathbb{G}, \mathbb{G}_T, \hat{e}(\cdot,\cdot), P, S = P^s, H_1(\cdot), H_2(\cdot), H_3(\cdot), H_4(\cdot)), \mathsf{Msk} = s.$$

where $s \in_R \mathbb{Z}_p$, P is an arbitrary generator of $\mathbb{G}$, $H_1(\cdot), H_2(\cdot), H_3(\cdot), H_4(\cdot)$ are cryptographic hash functions modeled as random oracles. Their domains and ranges will be clear from the description of the other algorithms.

$\mathsf{Extract}(\mathsf{Msk}, T)$: Given a time-identifier T, the time-dependent trapdoor is $d_T = P^{\frac{1}{(s+t)}}$, where $t = H_1(T)$.

$\mathsf{KeyGen}()$: Pick $\mathsf{sk} \in_R \mathbb{Z}_p^*$, return $(\mathsf{sk}, g^{\mathsf{sk}})$ as the private/public key pair.

$\mathsf{Enc}(m, T, \mathsf{pk})$: To encrypt a message m under time T and a public key pk:

1. Compute $t = H_1(T) \in \mathbb{Z}_p^*$;
2. Choose $x \in_R \{0,1\}^{\lambda_0}$ and $h = H_2(m||x||T) \in \{0,1\}^{2\lambda}$;
3. Treat $\bar{h}$ as the 2λ-bit integer value of h, parse it as $r_1||r_2$, where $r_1, r_2 \in \mathbb{Z}_p^*$;
4. Compute $c_1 = (S \cdot P^t)^{r_1}$ and $c_2 = P^{r_2}$;
5. Compute $d = H_3(\hat{e}(P,P)^{r_1}) \in \mathbb{Z}_p^*$;

6. Compute $K = H_4(\mathsf{pk}^{(d \cdot r_2)})$ and $c_3 = (m||x||h) \oplus K$;
7. Return $C = (c_1, c_2, c_3, T)$.

$\mathsf{Dec}(C, d_T, \mathsf{sk})$: To decrypt C using the trapdoor d_T and the secret key sk:

1. Compute $d = H_3(\hat{e}(c_1, d_T))$;
2. Compute $K = H_4(c_2^{(d \cdot \mathsf{sk})})$;
3. Parse $c_3 \oplus K$ as $m||x||h$;
4. Return m if $H_2(m||x||T) = h$.

3.2 Attacks

Even though there is a checking of $H_2(m||x||T) = h$, there is no checking whether r_1 in $c_1 = (S \cdot P^t)^{r_1}$ and r_2 in $c_2 = P^{r_2}$ are really from the 2λ-bit integer value of h. Our attacks exploit this fact. Given the challenge $C^* = (c_1^*, c_2^*, c_3^*, T^*)$ that is encrypted under the public key pk^*, our first attack proceeds as follows.

Attack 1 (with a strong decryption oracle)

1. Randomly choose $z \in \mathbb{Z}_p$;
2. Compute $c_2' = c_2^{*z}$;
3. Query the decryption oracle to decrypt $(c_1^*, c_2', c_3^*, T^*)$ with respect to the *replaced* public key $\mathsf{pk}^{*1/z}$.

Suppose $\mathsf{pk}^* = g^{\mathsf{sk}^*}$, the adversary does not know the secret key sk^*/z corresponding to $(\mathsf{pk}^*)^{1/z}$. However, in the security model ([6, Definition 2]), the adversary is entitled with a decryption oracle that can decrypt any ciphertext except the challenge one, under any public key without supplying the corresponding private key. So this is a legitimate decryption query.

We claim that the decryption oracle will just return the message encrypted inside the challenge ciphertext. To see this, the decryption oracle computes $d = H_3(\hat{e}(c_1^*, d_{T^*}))$ and $K = H_4(c_2'^{(d \cdot \mathsf{sk}')}) = H_4(c_2^{*z(d \cdot \mathsf{sk}^*/z)}) = H_4(c_2^{*(d \cdot \mathsf{sk}^*)})$, which is exactly the K computed by $\mathsf{Dec}(C^*, d_{T^*}, \mathsf{sk}^*)$.

Attack 2 (by a curious time-server)
Following the reasoning of the above attack, a curious time server can launch a similar attack without the strong decryption oracle of a replaced public key.

1. Compute $t^* = H_1(T^*) \in \mathbb{Z}_p^*$;
2. Randomly choose $z \in \mathbb{Z}_p$;
3. Compute $c_1' = (S \cdot P^{t^*})^z$
4. Compute $d' = H_3(\hat{e}(P, P)^z) \in \mathbb{Z}_p^*$;
5. Recover $d^* = H_3(\hat{e}(c_1^*, d_{T^*}))$;
6. Compute $c_2' = c_2^{*(d^*/d')}$;
7. Query the decryption oracle to decrypt (c_1', c_2', c_3^*, T^*) with respect to *original* public key pk^*.

To see the correctness, the decryption oracle computes $d = H_3(\hat{e}(c'_1, d_{T^*})) = H_3(\hat{e}(P,P)^z) = d'$ and $K = H_4(c_2'^{(d' \cdot \mathsf{sk}')}) = H_4(c_2^{*(d^* \cdot \mathsf{sk}^* \cdot d'/d')}) = H_4(c_2^{*(d^* \cdot \mathsf{sk}^*)})$, which is exactly the K computed by $\mathsf{Dec}(C^*, d_{T^*}, \mathsf{sk}^*)$.

Here we pinpoint the flaw in the proof of CCA security in [6]. The adversary never query H_4 directly in the above attacks; however, the proof in [6] assumes that the adversary would have to request H_4 for a special value which lets the simulator to solve the underlying computational problem.

It is possible to fix the scheme by requiring the decryption algorithm to return m if and only if $c_1 = (S \cdot P^t)^{r_1}$ and r_2 in $c_2 = P^{r_2}$ where $r_1 || r_2 = \bar{h}$ and $\bar{h}$ is the 2λ-bit integer value of h. However, it adds two exponentiations in the decryption algorithm and lessens the purported advantage of their scheme. We remark that the encryption algorithm is unaffected and is still more efficient than other existing schemes.

4 Augmenting Chow *et al.*'s TRE Scheme with Pre-open Capability and Release-Time Confidentiality

4.1 Preliminaries

Let $\mathbb{G}$ be a multiplicative group of prime order p and $\mathbb{G}_T$ be a multiplicative group also of order p. We assume the existence of an efficiently computable bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that

1. *Bilinearity*: For all $u, v \in \mathbb{G}$ and $r, s \in \mathbb{Z}_p$, $\hat{e}(u^r, v^s) = \hat{e}(u, v)^{rs}$.
2. *Non-degeneracy*: $\hat{e}(u, v) \neq 1_{\mathbb{G}_T}$ for all $u, v \in \mathbb{G} \setminus \{1_{\mathbb{G}}\}$.

For the security of our scheme, the following problem is assumed to be intractable in such groups.

Definition 10. *The* Modified Decision **3**-Party Diffie-Hellman Problem *(3-MDDH) in* $\mathbb{G}$ *is to decide if* $Z = g^{\alpha\beta\gamma}$ *given* $(g, g^\alpha, g^\beta, g^\gamma, g^{\beta\gamma/\alpha}, Z) \in \mathbb{G}^6$.

Compared with the original 3-DDH problem, an extra element $g^{\beta\gamma/\alpha}$ is included in the problem instance.

Our scheme also requires a hash function H drawn from a family of collision resistant hash functions.

Definition 11. *A hash function* $H \xleftarrow{\$} \mathcal{H}(k)$ *is* collision resistant *if for all PPT algorithms* $\mathcal{C}$ *the advantage*

$$Adv_{\mathcal{C}}^{\mathrm{CR}}(k) = \Pr[H(x) = H(y) \wedge x \neq y | (x, y) \xleftarrow{\$} \mathcal{C}(1^k, H) \wedge H \xleftarrow{\$} \mathcal{H}(k)]$$

is negligible as a function of the security parameter k.

4.2 Construction

Below we reuse the basic version of the CRR scheme, which only supports a single-level of time-identifier and with two partial decryption algorithms combined into a single one. These simplifications give us back the CLE scheme

proposed by [13]; however, it has been pointed out that a generic transformation from CLE to TRE is unlikely to be provable secure [5] and Chow *et al.*'s notion of general CLE filled this gap [9]. Even though the scheme is defined based on n-bit time identifiers, one can use a collision-resistant hash function which maps from $\{0,1\}^*$ to $\{0,1\}^n$ for time identifiers of arbitrary length.

Capabilities for pre-opening and recovering the release-time have been added in the below description. We also changed the input of the hash function to provide release-time confidentiality, the change seems to spoil the nice feature of public ciphertext validity checking, which may harm CCA security, but it will be justified in our security proof.

- $\mathsf{Setup}(1^\lambda)$: Let $\mathbb{G}$, $\mathbb{G}_T$ be two multiplicative groups with a bilinear map $\hat{e}$ as defined before. They are of the same order p, which is a prime and $2^\lambda < p < 2^{\lambda+1}$. Pick the following components:
  - **Encryption key:** choose two generators $g, g_2 \in_R \mathbb{G}$.
  - **Master public key:** choose an exponent $\alpha \in_R \mathbb{Z}_q$ and set $g_1 = g^\alpha$.
  - **Hash key for time-identifier:** pick $(\ell+1)$ $\mathbb{G}$ elements $\vec{U} = (u', u_1, \cdots, u_\ell)$. Let $T = t_1 \cdots t_\ell$. Define $F_u(T) = u' \prod_{j=1}^{\ell} u_j^{t_j}$.
  - **Hash key for ciphertext validity:** pick $\vec{V} = (v', v_1, \cdots, v_\ell) \in_R \mathbb{G}^{\ell+1}$. This vector defines $F_v(w) = v' \prod_{j=1}^{\ell} {v_j}^{b_j}$ where w is an ℓ-bit string $b_1 b_2 \cdots b_\ell$.
  - **Key-derivation function (KDF):** In addition to the above basic parameters which also present in the CRR scheme, we need a KDF $K : \mathbb{G}_T \to \{0,1\}^{n+k+1}$, which we assume that the output of K is computationally indistinguishable from a random distribution when the input comes from a uniformly distribution. We also assume an implicit one-to-one mapping between $\mathbb{G}$ and $\{0,1\}^{k+1}$,

  The output is: $\mathsf{Msk} = g_2^\alpha$, $\mathsf{Pub} = \langle \lambda, q, \mathbb{G}, \mathbb{G}_T, \hat{e}(\cdot,\cdot), \ell, H(\cdot), K(\cdot), g, g_1, g_2, \vec{U}, \vec{V} \rangle$. Same as [9], we require the discrete logarithms (with respect to g) of all $\mathbb{G}$ elements in Pub except g, g_1 to be unknown to the time-server. In particular, the knowledge of the discrete logarithm of g_2 with respect to g enables the time-server to retrieve the release-time of a ciphertext. In practice, these elements can be generated from a pseudorandom function of a public seed.

- $\mathsf{Extract}(\mathsf{Msk}, T)$: Pick $r \in_R \mathbb{Z}_q^*$, return $D_T = \langle d_1, d_2 \rangle = \langle {g_2}^\alpha \cdot F_u(T)^r, g^r \rangle$.
- KeyGen: Pick $\mathsf{sk} \in_R \mathbb{Z}_q^*$ and $\mathsf{pk} = \langle X, Y \rangle = \langle g^{\mathsf{sk}}, {g_1}^{\mathsf{sk}} \rangle$.
- $\mathsf{Enc}(M, T, \langle X, Y \rangle)$:
  1. Check if pk is valid by $e(X, g_1) = e(g, Y)$, return $\perp$ if not.
  2. Pick $s \in_R \mathbb{Z}_q^*$, compute $k = K(\hat{e}(X, g_2)^s)$.
  3. Return $\langle C_1, C_2, \tau, \sigma \rangle = \langle M \cdot \hat{e}(Y, g_2)^s, (T || F_u(T)^s) \oplus k, g^s, F_v(w)^s \rangle$ where $w = H(C_1 || C_2 || k || \mathsf{pk})$.
  4. Pre-open key is computed as $V_C = g_1^s$.

- $\mathsf{GetTime}(\langle C_1, C_2, \tau, \sigma\rangle, \mathsf{sk})$:
  1. Compute $k' = K(\hat{e}(\tau, g_2)^{\mathsf{sk}})$.
  2. Parse $C_2 \oplus k'$ as $(T'||f_{T'})$.
  3. Check if $\hat{e}(\tau, F_u(T')F_v(w')) = \hat{e}(g, f_{T'}\sigma)$ where $w' = H(C_1||C_2||k'||\mathsf{pk})$.
  4. Return $\perp_C$ if inequality holds or any parsing is not possible.
  5. Otherwise return the time T' and auxiliary data $f_{T'}$.
- $\mathsf{PreOpen}(\langle C_1, C_2, \tau, \sigma\rangle, \mathsf{sk}, V_C)$:
  1. Check if the pre-open key is valid by $\hat{e}(V_C, g) = \hat{e}(g_1, \tau)$, returns $\perp_V$ if it does not hold.
  2. Return $\perp_C$ if $\mathsf{GetTime}(\langle C_1, C_2, \tau, \sigma\rangle, \mathsf{sk})$ returns $\perp_C$; otherwise, return $m \leftarrow C_1/\hat{e}(V_C, g_2)^{\mathsf{sk}}$.
- $\mathsf{Dec}(\langle C_1, C_2, \tau, \sigma\rangle, \langle d_1, d_2\rangle, \mathsf{sk})$:
  1. Compute $\mathsf{GetTime}(\langle C_1, C_2, \tau, \sigma\rangle, \mathsf{sk})$ to get $f_{T'}$.
  2. Return $\perp_C$ if $\mathsf{GetTime}$ returns $\perp_C$; otherwise, return $m = C_1 \cdot \{\frac{\hat{e}(\tau, d_2)}{\hat{e}(f_{T'}, d_1)}\}^{\mathsf{sk}}$.

4.3 Discussions on the Security Properties

It is easy to see that the ciphertext is binding with the pre-open key. Given τ, the random factor in a valid ciphertext is uniquely fixed. From the pre-open key validity checking $\hat{e}(V_C, g) = \hat{e}(g_1, \tau)$ and the bilinearity, V_C must be in a correct form. Hence, the probability for breaking the binding property is zero.

We have the following theorems for the security of our modified scheme.

Theorem 1. *Our scheme is secure against Type-I attack (Definition 3) if 3-MDDH problem is intractable.*

Theorem 2. *Our scheme is secure against Type-II attack (Definition 6) if 3-DDH problem is intractable.*

Theorem 3. *Our scheme is* RTC-II*-secure (Definition 9) if 3-DDH problem is intractable.*

The detailed proof of all these three theorems are all similar to the single proof in [10]. We claim that the proof for mode II in [10] is general enough to cover both cases for our Type-II attack: CCA-II-PO and RTC-II. To see, the first attack concerns about the indistinguishability of the messages, which is encrypted by $\hat{e}(Y^*, g_2)^{\gamma}$; and the second is about that of the time periods, which is encrypted by $\hat{e}(X^*, g_2)^{\gamma}$. Note that in the Type-II simulation, the challenger knows the master secret key α such that $Y^* = (X^*)^{\alpha}$. Both of them have the problem instance embedded. The ability to distinguish one of them in the respective mode of attack gives the solution of the underlying hard problem. This point can be seen in the second last game of the proof.

Here we highlight the changes of the proof in [10] we should make for our case. The new things in the security proof include:

1. all decryption oracles including $\mathsf{GetTime}$
2. the new well-formness checking of the ciphertext (to output w, the hash H now takes $\hat{e}(X, g_2)^s$ as part of the input, but not just public information)

3. the last term is not related to the time (the hash H does not take T as input)
4. the simulation of the pre-open oracle and the pre-open key
5. the simulation of the challenge ciphertext in a new format

For the first two issues, even though the term $F_U(T)^s$ is now hidden by $K(\hat{e}(X, g_2)^s)$ and it seems that the ciphertext validity cannot be checked, the simulator $\mathcal{S}$ only takes C_1 and τ to compute g_2^s (in [10, Game 6]), and hence the term $K(\hat{e}(X, g_2)^s)$ can be recovered. Specifically, $\mathcal{S}$ computes $\hat{e}(Y, g_2)^s$ as $\hat{e}(Y, (\sigma/\tau^{K_v(w)})^{\frac{1}{J_v(w)}})$, we thus have $\hat{e}(X, (\sigma/\tau^{K_v(w)})^{\frac{1}{J_v(w)}}) = \hat{e}(X, g_2)^s$. The same is true for computing w for well-formness checking.

Intuitively, the malleable XOR cipher can be used in C_2 since the decryption algorithms checks the σ term, which is computed from the hash taking C_2 as part of the input. It is possible for the adversary to change T in C_2; however, the adversary needs to change the T embedded in $F_u(T)^s$ as well.

When PreOpen oracle is queried upon a valid pre-open key corresponding to the ciphertext to be decrypted, the whole thing can be simulated by DecO oracle. From the discussion of binding property, we have a robust method to rule out invalid pre-open key, and hence the simulation is perfect too.

The pre-open key of the challenge ciphertext to be given to a Type-II adversary can be computed by $\tau^\alpha = g_1^\gamma$, since α is known in Type-II simulation.

Lastly, what remains is to show the simulation of the challenge ciphertext in a new format for a Type-II adversary. This is done in different way (in [10, Game 8]) according to the type of the adversary. For Type-I, $\mathcal{S}$ computes $\hat{e}(X, g_2)^\gamma$ by $\hat{e}(Y^*, g^{\beta\gamma/\alpha}) = \hat{e}((X^*)^\alpha, g^{\beta\gamma/\alpha}) = \hat{e}(X^*, g^\beta)^\gamma = \hat{e}(X^*, g_2)^\gamma$. That is the reason we require a modified version of 3-DDH assumption. For Type-II, since $Y^* = (X^*)^\alpha$, $\mathcal{S}$ can easily obtain $\hat{e}(X^*, g_2)^\gamma$ by $(\hat{e}(Y^*, g_2)^\gamma)^{\frac{1}{\alpha}}$.

5 Conclusion

This paper revisits two recent results in TRE. We demonstrate an "imaginary" attack which makes use of a strong decryption oracle on Chalkias *et al.*'s TRE scheme, and show how this vulnerability can be exploited by a curious time-server in a more realistic attack. We then equip Chow *et al.*'s TRE scheme in the standard model with pre-open capability and release-time confidentiality, some features that are desirable in practical applications.

Since the release-time of a ciphertext is something that the intended recipient should know, we do not assume it is sent in out-of-band channel, instead it should be encrypted in the ciphertext. We formalize this notion of release-time confidentiality, and show it can be obtained similar to how we get message confidentiality. We leverage the fact that a kind of double-encryption is done in TRE, so nothing like an anonymous identity-based encryption is used as a building block. It is also interesting to see that the public ciphertext validity checking is "removed" but the security is still preserved.

References

1. Baek, J., Safavi-Naini, R., Susilo, W.: Token-Controlled Public Key Encryption. In: Deng, R.H., Bao, F., Pang, H., Zhou, J. (eds.) ISPEC 2005. LNCS, vol. 3439, pp. 386–397. Springer, Heidelberg (2005)
2. Bellare, M., Goldwasser, S.: Verifiable Partial Key Escrow. In: ACM Conference on Computer and Communications Security, pp. 78–91 (1997)
3. Blake, I.F., Chan, A.C.-F.: Scalable, Server-Passive, User-Anonymous Timed Release Cryptography. In: ICDCS 2005, pp. 504–513. IEEE Computer Society Press, Los Alamitos (2005)
4. Boneh, D., Naor, M.: Timed Commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
5. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and Non-interactive Timed-Release Encryption. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005)
6. Chalkias, K., Hristu-Varsakelis, D., Stephanides, G.: Improved Anonymous Timed-Release Encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 311–326. Springer, Heidelberg (2007)
7. Cheon, J.H., Hopper, N., Kim, Y., Osipko, I.: Timed-Release and Key-Insulated Public Key Encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 191–205. Springer, Heidelberg (2006)
8. Chow, S.S.M.: Token-Controlled Public Key Encryption in the Standard Model. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 315–332. Springer, Heidelberg (2007)
9. Chow, S.S.M., Roth, V., Rieffel, E.: General Certificateless Encryption and Timed-Release Encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 126–143. Springer, Heidelberg (2008)
10. Chow, S.S.M., Roth, V., Rieffel, E.G.: General Certificateless Encryption and Timed-Release Encryption. Cryptology ePrint Archive, Report 2008/023 (2008)
11. Chow, S.S.M.: Certificateless Encryption. In: Identity-Based Cryptography. IOS Press, Amsterdam (2008)
12. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional Oblivious Transfer and Timed-Release Encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)
13. Dent, A.W., Libert, B., Paterson, K.G.: Certificateless Encryption Schemes Strongly Secure in the Standard Model. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 344–359. Springer, Heidelberg (2008), http://eprint.iacr.org/2007/121
14. Dent, A.W., Tang, Q.: Revisiting the Security Model for Timed-Release Public-Key Encryption with Pre-Open Capability. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 158–174. Springer, Heidelberg (2007)
15. Dodis, Y., Yum, D.H.: Time Capsule Signature. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 57–71. Springer, Heidelberg (2005)
16. Dwork, C., Naor, M.: Pricing via Processing or Combatting Junk Mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993)
17. Galindo, D., Herranz, J.: A Generic Construction for Token-Controlled Public Key Encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 177–190. Springer, Heidelberg (2006)

18. Hwang, Y.H., Yum, D.H., Lee, P.J.: Timed-Release Encryption with Pre-open Capability and Its Application to Certified E-mail System. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 344–358. Springer, Heidelberg (2005)
19. May, T.: Time-release Crypto, Manuscript (February 1993), http://www.cyphernet.org/cyphernomicon/chapter14/14.5.html
20. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock Puzzles and Timed-release Crypto. Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology (1996)

Efficient and Provably Secure Certificateless Multi-receiver Signcryption

S. Sharmila Deva Selvi[1], S. Sree Vivek[1,⋆], Deepanshu Shukla[2,⋆⋆], and Pandu Rangan Chandrasekaran[1,⋆]

Department of Computer Science and Engineering,
Indian Institute of Technology Madras
sharmila@cse.iitm.ac.in, svivek@cse.iitm.ac.in, prangan@iitm.ac.in
and Institute of Technology Banaras Hindu University
deepanshus.itbhu@gmail.com

Abstract. Certificateless cryptography aims at combining the advantages of identity based and public key cryptography, so as to avoid the key escrow problem inherent in the identity based system and cumbersome certificate management in public key infrastructure. Signcryption achieves confidentiality and authentication simultaneously in an efficient manner. Multi-receiver signcryption demands signcrypting the same message efficiently for a large number of receivers. In this paper, we propose the first efficient certificateless multi-receiver signcryption scheme and prove it secure in the random oracle model. Our scheme does not require pairing to signcrypt a message for any number of receivers. We are considering a more realistic adversarial model and proving the security against insider attacks, which guarantees non-repudiation and forward secrecy.

Keywords: Certificateless, Signcryption, Multi-receiver, Bilinear Pairing.

1 Introduction

Signcryption proposed by Zheng in [3] is a cryptographic primitive providing signature and encryption simultaneously, at a lower computational cost and communication overhead than the signature-then-encryption approach. A proper signcryption scheme should provide confidentiality as well as authentication and non-repudiation. Besides this, security model for signcryption should consider insider attacks also i.e. a corrupted receiver should not be able to forge a valid signcryption from any legal user on a message that was not already sent by that user. Sometimes forward secrecy is also a desired property, which requires that even if a sender's secret key is exposed at some point of time, the past messages sent by him should remain secret.

⋆ Work supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation sponsored by Department of Information Technology, Government of India.

⋆⋆ Work supported by INAE undergraduate mentoring programme.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 52–67, 2008.

Need for multi-receiver signcryption arises when the same message is to be sent to a large number of receivers. Consider the case of a company in which there are several managers and each of them has to send authenticated and confidential report to a large number of employees. In this case, simply signcrypting the message for each receiver will be highly inefficient. Therefore, there is a need to design efficient schemes for this task. In this paper, we propose an efficient multi-receiver signcryption scheme(CLMSC) in certificateless setting.

Related works: The concept of multi-receiver setting was formalized by Bellare *et al.* in [18]. The naive way to do this is to simply process the message for all the n receivers, which is very inefficient. In [11], Kurosawa showed that one could use "randomness re-use" technique to design multi-receiver encryption schemes. This saves bandwidth and minimizes the computation cost. Almost all the signcryption schemes are based on bilinear pairings. Therefore, even after using "randomness re-use" we end up with n pairings, which is also inefficient. In [9], Zheng proposed a signcryption scheme for multiple recipients. In [10], Duan *et al.* proposed an efficient identity based multi-receiver signcryption scheme. In [2], Yu *et al.* propose an identity based signcryption scheme for multiple receivers. In [13], the scheme in [2] is shown to be insecure and a fix is also given for the same.

Since the introduction of certificateless cryptography by Al-Riyami and Paterson[1] in 2003, many good encryption and signature schemes [19] [21] [20] have been proposed in certificateless setting. In 2008, Barbosa *et al.* proposed the first certificateless signcryption scheme [12] and gave the security model for the same. But they proved the security of their scheme in slightly weaker model. In [12], while proving confidentiality, access to signcryption oracle was not given to the adversary. It is desirable to allow the adversary to access the signcryption oracle because the adversary may get some extra information about senders secret key by querying the signcryption oracle. Besides this limitation, [12] cannot be directly adapted for multi-receiver settings.

Our Contribution: In this paper, we introduce the notion of certificateless multi-receiver signcryption(CLMSC) and give a security model for the same. We also propose a concrete, efficient scheme and prove its security against insider attacks in random oracle model. We also show that our scheme is efficient than the naive extension of [12] for multi-receiver settings. Even for single receiver our scheme is more efficient than [12]. We have also shown that our scheme is efficient than few existing identity based multi-receiver signcryption schemes [10] [13].

2 Preliminaries

2.1 Computational Assumptions

In this section, we recall the computational assumptions related to bilinear maps [16] that are relevant to the security of our scheme:

1. **Strong Diffie-Hellman Problem (SDHP).** SDH problem is a stronger version of DHI(Diffie-Hellman Inversion problem)[16]. Given $(P, aP) \in \mathbb{G}_1^2$ for

any random $a \in \mathbb{Z}_q^*$, the SDH problem in $\mathbb{G}_1$ is to compute $\left(h, (a+h)^{-1}P\right)$, $h \in \mathbb{Z}_Q^*$.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the SDH problem in $\mathbb{G}_1$ is defined as:

$$Adv_{\mathcal{A}}^{SDH} = Pr\left[\mathcal{A}(P, aP) = \left(h, (a+h)^{-1}P\right) \mid a, h \in \mathbb{Z}_q^*\right]$$

We say that SDH is (t, ϵ) hard if for any t time probabilistic algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{SDH} < \epsilon$.

2. **Collusion Attack Algorithm with k-traitors (k-CAA).** Given $(P, aP, (h_1+a)^{-1}P, \ldots, (h_k+a)^{-1}P) \in \mathbb{G}_1^{k+2}$ for any random $a \in \mathbb{Z}_q^*$ and known values $h_1, \ldots, h_k \in \mathbb{Z}_q^*$, the k-CAA problem in $\mathbb{G}_1$ is to compute $(a+h)^{-1}P$ for some $h \notin \{h_1, \ldots, h_k\}$.

   The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the k-CAA problem in $\mathbb{G}_1$ is defined as:

$$\begin{aligned} Adv_{\mathcal{A}}^{k-CAA} &= Pr[\mathcal{A}(P, aP, (h_1+a)^{-1}P, \ldots, (h_k+a)^{-1}P, h_1, \ldots, h_k) \\ &= (a+h)^{-1}P \mid a, h \in \mathbb{Z}_q^*, h \notin \{h_1, \ldots, h_k\}] \end{aligned}$$

   We say that k-CAA is (t, ϵ) hard if for any t time probabilistic algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{k-CAA} < \epsilon$.

3. **Modified BDHI for k-values (k-mBDHIP).** k-mBDHIP is the bilinear variant of the k-CAA problem [22]. Given $(P, sP, (h_1+s)^{-1}P, \ldots, (h_k+s)^{-1}P) \in \mathbb{G}_1^{k+2}$ for any random $s \in \mathbb{Z}_q^*$ and known values $h_1, \ldots, h_k \in \mathbb{Z}_q^*$, the k-mBDHIP problem is to compute $\hat{e}(P, P)^{(s+h)^{-1}}$ for some $h \notin \{h_1, \ldots, h_k\}$.

   The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the k-mBDHIP problem in is defined as:

$$\begin{aligned} Adv_{\mathcal{A}}^{k-mBDHIP} &= Pr[\mathcal{A}(P, sP, (h_1+s)^{-1}P, \ldots, (h_k+s)^{-1}P, h_1, \ldots, h_k) \\ &= \hat{e}(P, P)^{(s+h)^{-1}} \mid s, h \in \mathbb{Z}_q^*, h \notin \{h_1, \ldots, h_k\}] \end{aligned}$$

   We say that k-mBDHIP is (t, ϵ) hard if for any t time probabilistic algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{k-mBDHIP} < \epsilon$.

4. **Gap Bilinear Diffie-Hellman Problem (GBDHP).** [12] Given $(P, aP, bP, cP) \in \mathbb{G}_1^4$ for any random $a, b, c \in \mathbb{Z}_q^*$, the GBDH problem in $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ is to compute $\hat{e}(P, P)^{abc}$ given access to DBDH oracle $\mathcal{O}_\Gamma$ which on input $(P, aP, bP, cP, T) \in \mathbb{G}_1^4 \times \mathbb{G}_2$ outputs 1 if $T = \hat{e}(P, P)^{abc}$ and 0 otherwise.

   The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the GBDH problem in $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ is defined as:

$$Adv_{\mathcal{A}}^{GBDH}(\mathcal{O}_\Gamma, q_{DBDH}) = Pr\left[\mathcal{A}^{\mathcal{O}_\Gamma}(P, aP, bP, cP) = \hat{e}(P, P)^{abc} \mid a, b, c \in \mathbb{Z}_q^*\right]$$

   where q_{DBDH} is the number of queries to the decisional oracle. We say that GBDHP is (t, ϵ, q_{DBDH}) hard if for any t time probabilistic algorithm $\mathcal{A}$ asking q_{DBDH} oracle queries, advantage $Adv_{\mathcal{A}}^{k-CAA} < \epsilon$.

3 Certificateless Multi-receiver Signcryption

3.1 Framework of Certificateless Multi-receiver Signcryption

Any generic certificateless multi-receiver signcryption scheme is a five tuple of probabilistic polynomial time algorithms defined as follows-

1. ***Setup***(1^κ): This algorithm is run by the KGC. It takes as input the security parameter 1^κ and returns the KGC's master secret key Msk, master public key Mpk, public parameters $Params$ and a description of message space($\mathcal{M}_{CLMSC}$) and cipher-text space ($\mathcal{C}_{CLMSC}$).
2. ***Partial Private Key Extract***$(ID_i, Msk, Params)$: This algorithm is run by the KGC. It takes as input $Msk, Params$, a string $ID_i \in \{0,1\}^*$ and returns a partial private key D_i.
3. ***Key Extract***$(ID_i, D_i, Params)$: This algorithm is run by the user. It takes as input the partial private key of the user and returns a public key PK_i and a secret value x_i. The full secret key of the user is set to $SK_i = \langle x_i, D_i \rangle$.
4. ***Signcrypt***$(m, ID_S, SK_S, PK_S, L = \{ID_1, ID_2, \ldots, ID_n\}, PK_1, \ldots, PK_n, Params)$: The signcryption algorithm takes as input a message $m \in \mathcal{M}_{CLMSC}$, identity ID_S and the full secret key SK_S of the sender, a list L of the receiver identities and their public keys and returns a ciphertext $\sigma \in \mathcal{C}_{CLMSC}$.
5. ***Designcrypt***$(\sigma, SK_R, ID_R, PK_R, ID_S, PK_S, L)$: This is a deterministic algorithm which takes as input the ciphertext σ, receivers full secret key SK_R, identity ID_R, the public key PK_R of the receiver, list of receivers L, the identity ID_S and the public key PK_S of the sender and returns either a plaintext $m \in \mathcal{M}_{CLMSC}$ or an error symbol $\perp$.

For consistency, we require that
if $\sigma = Signcrypt(m, ID_S, SK_S, PK_S, L = \{ID_{R_1}, \ldots, ID_{R_n}\}, PK_1, \ldots, PK_n, Params)$, then $m = Designcrypt\,(\sigma, SK_{R_i}, ID_{R_i}, PK_{R_i}, ID_S, PK_S, L)$ for $1 \leq i \leq n$.

3.2 Security Model for Certificateless Multi-receiver Signcryption

Now, we describe the security model for certificateless multi-receiver signcryption. In our model we consider the security against selective identity attack. Although it is a slightly weaker model, it is the common approach prevalent in the literature [9] [10] [2] [15] for multiple-receiver settings. It means that adversary reveals in advance the challenge identity. This was first proposed by Canetti *et al.* in [14] and extended to selective multi-identity attack in [10]. In selective multi-identity attack it is assumed that adversary outputs the set of receiver identities to be attacked in advance. However our model is different from [14] because in our model adversary is required to submit the list of target identities after seeing the master public key. However, in Canetti *et al.*'s paper, the adversary is required to submit an identity before seeing the master public key. We consider a stronger adversary as it can choose the list of target identities based

on its knowledge of public parameters. Following the trend in literature [2] [4] [6] [7] [8], we do not consider the attacks targeting the signcryptions where the sender is same as one of the receivers. We also disallow the signcryption queries, where both sender and one or more of the receivers belong to the challenge set. In confidentiality and unforgeability game we provide access to the following six oracles :

1. ***Extract Partial Private Key:*** On input of an identity ID_i, this oracle returns the partial private key D_i generated using the *Partial Private Key Extract* algorithm.
2. ***Extract Secret Key:*** On input of an identity ID_i, this oracle returns the full secret key $SK_i = \langle x_i, D_i \rangle$ of the identity using the appropriate algorithms.
3. ***Request Public Key:*** On input of an identity ID_i, this oracle returns the corresponding public key PK_i associated with ID_i. If such a key does not exist then it is constructed using *Key Extract* algorithm.
4. ***Replace Public Key:*** On input of an identity ID_i and a valid public key PK_i', this oracle replaces the public key associated with ID_i with PK_i'. If such a key does not exist then it is generated using the *Key Extract* algorithm and then the public key corresponding to ID_i is replaced with PK_i'.
5. ***Signcrypt:*** On input of a message, a sender's identity ID_S and a set of receiver identities $L = \{ID_{R_1}, ID_{R_2}, \ldots, ID_{R_n}\}$, this oracle returns the result of running the signcryption algorithm on the message, sender's full secret key and the receiver's public parameters.
6. ***Designcrypt:*** On input of a ciphertext, a sender's identity ID_S and a receiver's identity ID_R, this oracle returns the result of running the *Designcrypt* algorithm on the ciphertext, the sender's public parameters and the receiver's full secret key.

Next, we give the security definitions. Following the trend in literature we also consider Type-I and Type-II adversary. Roughly speaking Type-I adversary models a common user who is not in possession of the master secret key Msk and a type-II adversary models the honest but curious KGC.

Confidentiality: Security game that captures the confidentiality is based on the ciphertext indistinguishability. We define it separately for Type-I and Type-II adversary:

Type-I: A certificateless multi-receiver signcryption scheme is Type-I-iCCA2 secure if every probabilistic polynomial-time attacker $\mathcal{A}$ has negligible advantage in winning the IND-CLMSC-iCCA2-I game. A type-I adversary is given access to all the 6 oracles defined above under the following constraints-

1. Adversary does not have access to master secret key Msk.
2. No *Extract Secret Key* query is allowed on any of the challenge identities.
3. Adversary is not allowed to ask *Extract Partial Private Key* query for any of the challenge identities.

IND-CLMSC-iCCA2-I game played between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ is defined below:

Setup: Challenger $\mathcal{C}$ runs the setup algorithm to generate master secret key Msk and public parameters $Params$. $\mathcal{C}$ gives $Params$ to $\mathcal{A}$ while keeping Msk secret. After receiving $Params$ $\mathcal{A}$ outputs list of target identities denoted by $L^* = \{ID_1^*, ID_2^*, \ldots, ID_n^*\}$ respectively. $\mathcal{C}$ interacts with $\mathcal{A}$ in two phases:
Phase1: $\mathcal{A}$ is given access to all the six oracles. $\mathcal{A}$ adaptively queries the oracles consistent with the constraints described above.
Challenge: $\mathcal{A}$ outputs two equal length messages m_0, m_1 and an arbitrary sender's identity ID_S. $\mathcal{C}$ randomly chooses a bit $b \in_R \{0,1\}$ and computes a signcryption

$$\sigma^* = Signcrypt\left(m_b, ID_S, SK_S, PK_S, L = \{ID_1^*, \ldots, ID_n^*\}, PK_1^*, \ldots, PK_n^*\right)$$

σ^* is sent to $\mathcal{A}$ as challenge.
Phase2:$\mathcal{A}$ adaptively queries the oracles consistent with the constraints described above. Besides this it cannot query *Designcrypt* on σ^* for any $ID \in \{ID_1^*, ID_2^*, \ldots, ID_n^*\}$.
Guess: $\mathcal{A}$ outputs a bit b' at the end of the game. $\mathcal{A}$ wins if $b = b'$. The advantage of $\mathcal{A}$ is defined as-

$$Adv_{\mathcal{A}}^{IND-CLMSC-iCCA2-I} = |2Pr\left[b = b'\right] - 1|$$

Type-II: A certificateless multi-receiver signcryption scheme is Type-II-iCCA2 secure if every probabilistic polynomial-time attacker $\mathcal{A}$ has negligible advantage in winning the IND-CLMSC-iCCA2-II game. A type-II adversary is given access to all the 6 oracles defined above and master secret key Msk under the following constraints-

1. No *Extract Secret Key* query is allowed on any of the challenge identities.
2. No *Replace Public Key* query is allowed on any of the challenge identities before the challenge phase.

IND-CLMSC-iCCA2-II game played between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ is same as the IND-CLMSC-iCCA2-I game with the restrictions mentioned above

The advantage of $\mathcal{A}$ is defined as-

$$Adv_{\mathcal{A}}^{IND-CLMSC-iCCA2-II} = |2Pr\left[b = b'\right] - 1|$$

Authenticity: Strong existential unforgeability(sEUF-CLMSC-iCMA) game captures the authenticity as a security requirement for any certificateless multi-receiver signcryption. By strong unforgeability we mean that adversary should not be able to signcrypt a message on behalf of a sender even if it knows the secret keys of all the receivers. The game is defined as below:

Type-I: A certificateless multi-receiver signcryption scheme is Type-I-sEUF-iCMA-I secure if every probabilistic polynomial-time attacker $\mathcal{F}$ has negligible advantage in winning the sEUF-CLMSC-iCMA-I game. A type-I adversary is given access to all the 6 oracles defined above under the following constraints:

1. Adversary does not have access to master secret key Msk.
2. No *Extract Secret Key* query is allowed on any of the challenge identities.
3. Adversary is not allowed to ask *Extract Partial Private Key* query for any of the challenge identities.

sEUF-CLMSC-iCMA-I game played between the challenger $\mathcal{C}$ and the adversary $\mathcal{F}$ is defined below:

Setup: Challenger $\mathcal{C}$ runs the setup algorithm to generate master secret key Msk and public parameters $Params$. $\mathcal{C}$ gives $Params$ to $\mathcal{F}$ while keeping Msk secret. After receiving $Params$ $\mathcal{F}$ outputs list of target identities denoted by $L^* = \{ID_1^*, ID_2^*, \ldots, ID_n^*\}$ respectively. $\mathcal{C}$ interacts with $\mathcal{F}$ in two phases:
Attack: $\mathcal{F}$ is given access to all the six oracles. $\mathcal{F}$ adaptively queries the oracles consistent with the constraints described above.
Forgery: $\mathcal{F}$ outputs a signature σ^* and n arbitrary receiver's identities $L = \{ID_{R_1}, \ldots, ID_{R_n}\}$(there exists atleast one receiver ID_{R_i} such that, $ID_{R_i} \notin L^*$). $\mathcal{F}$ wins if $Designcrypt(\sigma^*, SK_{R_i}, ID_{R_i}, PK_{R_i}, ID_j^*, PK_j^*, L)$ returns m for $i, j \in \{1, \ldots, n\}$ and σ^* was not the output of any signcrypt query $Signcrypt(m, ID_i^*, L = \{ID_{R_1}, \ldots, ID_{R_n}\})$.That is, $\mathcal{F}$ wins if it outputs a valid signcryption from a target identity to the set of receiver identities L by itself.

$Adv_{\mathcal{F}}^{sEUF-CLMSC-iCMA-II}$ is defined as the probability that $\mathcal{F}$ wins the above game.

Type-II: A certificateless multi-receiver signcryption scheme is Type-II-sEUF-iCMA-I secure if every probabilistic polynomial-time attacker $\mathcal{F}$ has negligible advantage in winning the sEUF-CLMSC-iCMA-II game. A type-II adversary is given access to all the 6 oracles defined above and the master secret key Msk under the following constraints-

1. No *Extract Secret Key* query is allowed on any of the challenge identities.
2. Adversary is not allowed to ask *Replace Public Key* for any of the challenge identities.

sEUF-CLMSC-iCMA-II game played between the challenger $\mathcal{C}$ and the adversary $\mathcal{F}$ is same as sEUF-CLMSC-iCMA-I with the restrictions given above. $Adv_{\mathcal{F}}^{sEUF-CLMSC-iCMA-II}$ is defined as the probability that $\mathcal{F}$ wins the sEUF-CLMSC-iCMA-II game.

4 Certificateless Multi-receiver Signcryption Scheme (CLMSC)

In this section, we present a new certificateless multi-receiver signcryption scheme (CLMSC). In this scheme the ciphertext σ consists of two parts, first part c is common to all the users and second part is a n tuple. The i^{th} component d_i of the n tuple is specific to ID_i. For designcrypting, the receiver does the following: 1) Extracts the common part. 2) Identifies it's rank i in the list L. 3) Extracts the i^{th} component and then runs the designcryption algorithm. Note that the form of the ciphertext in our scheme is similar to the one proposed in [10], but our

scheme is entirely different from [10] in all other details. The CLMSC consists of the five algorithms that are given below.

Setup(1^κ)***:*** On providing security parameter 1^κ as input, the KGC chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, two random generators P and Q of $\mathbb{G}_1$ such that $P \neq Q$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. It then computes $g = \hat{e}(P, Q) \in \mathbb{G}_2$ and defines five hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \mathbb{Z}_q^* \times \mathbb{G}_2 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_3 : \{0,1\}^m \times \mathbb{G}_2 \times \{0,1\}^* \times \mathbb{G}_2 \times \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_4 : \mathbb{Z}_q^* \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_5 : \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{G}_2 \times \{0,1\}^* \rightarrow \{0,1\}^{k_1+k_2}$, where k_1 and k_2 are the number of bits required to represent $\mathbb{G}_1$ and $\mathbb{Z}_q^*$ elements respectively. Then KGC chooses $s \in_R \mathbb{Z}_q^*$ as the master secret key and sets $P_{pub} = sP$. The KGC now publishes the public parameters $Params$ of the system as $\langle \mathbb{G}_1, \mathbb{G}_2, P, Q, P_{pub}, \hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2, g, H_1, H_2, H_3, H_4, H_5 \rangle$.

Partial Private Key Extract*$(ID_i, msk, Params)$:* On input ID_i, the partial private key of user with identity ID_i is computed as $D_i = (q_i + s)^{-1} Q$, where $q_i = H_1(ID_i)$.

Key Extract*$(ID_i, D_i, Params)$:* This algorithm is run by each user to compute his private and public keys. The user ID_i chooses $x_i \in_R \mathbb{Z}_q^*$ and sets his private key $SK_i = \langle x_i, D_i \rangle$ and sets his public key as $Pk_i = \langle PK_{i1}, PK_{i2} \rangle = \langle g^{x_i}, x_i T_i \rangle$, where $T_i = (q_i + s)P$.

Signcrypt*$(m, ID_S, SK_S, PK_S, L = \{ID_{R_1}, ID_{R_2}, \ldots, ID_{R_n}\}, PK_{R_1}, PK_{R_2}, \ldots, PK_{R_n}, Params)$:*

1. Choose $r_1 \in_R \mathbb{Z}_q^*$ and compute $\omega = g^{r_1}$
2. Set $r_2 = H_2(r_1, \omega, ID_S)$ and $h_3 = H_3(m, \omega, r_2, ID_S, PK_{S1}, PK_{S2}, L)$
3. Compute $Z_S = \dfrac{r_1}{(x_S + h_3)} D_S$
4. Compute $c = H_4(r_2, ID_S) \oplus m$
5. Repeat the following steps for all $ID_{R_i} \in L$, $i = 1, 2, \ldots, n$.
   (a) Parse PK_{R_i} as $\langle PK_{i1}, PK_{i2} \rangle$
   (b) Set $h_{5i} = H_5(g^{r_1}, (PK_{i1})^{r_1}, PK_{i1}, ID_{R_i})$
   (c) Compute $d_{i1} = r_1(q_i + s)P$ and $d_{i2} = h_{5i} \oplus r_2 \| Z_S$
   (d) Set $d_i = \langle d_{i1}, d_{i2} \rangle$
6. Return ciphertext $\sigma = \langle c, d_1, d_2, \ldots, d_n, L \rangle$.

Designcrypt*$(\sigma = \langle c, d_1, d_2, \ldots, d_n, L \rangle, ID_S, ID_i, SK_i, Params)$:*

1. Parse d_i as $\langle d_{i1}, d_{i2} \rangle$ and SK_i as $\langle x_i, D_i \rangle$
2. Compute $\omega' = \hat{e}(d_{i1}, D_i)$ and $(\omega')^{x_i} = (PK_{i1})^{r_1}$
3. Set $h'_{5i} = H_5(\omega', (\omega')^{x_i}, PK_{i1}, ID_i)$
4. Compute $r'_2 \| Z'_S = h'_{5i} \oplus d_{i2}$
5. Compute $m' = c \oplus H_4(r'_2, ID_S)$
6. Set $h'_3 = H_3(m', \omega', r'_2, ID_S, PK_{S1}, PK_{S2}, L)$
7. If $\hat{e}(PK_{S2} + h'_3(q_S + s)P, Z'_S) = \omega'$ then return m', else return $\perp$.

5 Security Results

Theorem 1. *If an IND-CLMSC-iCCA2-I adversary $\mathcal{A}$ has advantage ϵ against our scheme running in time τ and asking q_{H_i} $(i = 1, 2, 3, 4, 5)$ queries to random*

oracles H_i $(i = 1, 2, 3, 4, 5)$, q_{sc} signcryption queries, q_{dsc} designcryption queries, q_{ske} extract secret key queries, q_{ppe} partial private key extract queries, q_{pk} public key request queries and q_{pkr} public key replacement queries, then there exists an algorithm $\mathcal{C}$ that solves the $k - mBDHIP$ for $k = q_{H_1} - n$, with advantage

$$\epsilon' > \frac{\epsilon}{(2nq_{sc} + q_{H_3} + q_{H_5})} \left(1 - \frac{q_{sc}\,(nq_{sc} + q_{H_3})}{2^{k_2}}\right) \left(1 - q_{dsc} \left(\frac{1}{2^{k_2}} + \frac{1}{2^{k_1+k_2}}\right)\right)$$

and with time

$$\tau' < \tau + O\,(q_{pk} + nq_{sc} + q_{dsc})\,.t_p + O\,(q_{pk} + nq_{sc} + q_{dsc} + n)\,.t_{sm} +$$

$$O(q_{pk} + nq_{sc} + q_{dsc}).t_e$$

where t_p, t_{sm}, t_e are the cost of pairing computation, scalar multiplication in $\mathbb{G}_1$ and exponentiation in $\mathbb{G}_2$, n is the number of receivers in the challenge set and k_1, k_2 are the number of bits needed to represent elements of $\mathbb{G}_1$ and $\mathbb{Z}_q^*$ respectively.

Proof

Algorithm $\mathcal{C}$ takes as input an instance $(P, sP, (q_1 + s)^{-1}P, \ldots, (q_k + s)^{-1}P, q_1, \ldots, q_k)$ of k-mBDHIP and interacts with $\mathcal{A}$.

Setup: $\mathcal{C}$ sets $P_{pub} = sP, Q = tP$ and sends the system parameters $\langle P, Q, P_{pub}, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ to $\mathcal{A}$. $\mathcal{A}$ then outputs a set of target identities, $L^* = \{ID_1^*, \ldots, ID_n^*\}$. $\mathcal{C}$ chooses $q_1^*, \ldots, q_n^* \in_R \mathbb{Z}_q^*$.

Phase1: $\mathcal{C}$ simulates $\mathcal{A}$'s queries as follows:

H_1 **queries:** On input ID_i, if an entry corresponding to ID_i is present in L_1, then $\mathcal{C}$ retrieves q_i from L_1 and returns q_i. Else, if $ID_i = ID_j^* (j = 1, \ldots, n)$, then $\mathcal{C}$ returns q_j^* and stores $\langle j, ID_j^*, q_j^*, \perp, \perp, \perp, b_i = 0\rangle$ in L_1. Otherwise, returns q_i, stores $\langle\, i, ID_i, q_i, \perp, \perp, \perp, b_i = 0\rangle$ in L_1. The three $\perp$ entries are for storing public keys PK_{i1}, PK_{i2} and user secret key x_i. The b_i is a marker bit used to denote whether the public keys have been replaced or not.

H_2 **queries:** On input (r, ω, ID), $\mathcal{C}$ returns the previously defined value if a tuple corresponding to (r, ω, ID) exists in L_2 . Otherwise, it returns a random $h_2 \in_R \mathbb{Z}_q^*$ and stores the entry $\langle h_2, r, \omega, ID\rangle$ in L_2.

H_3 **queries:** On input $(m, \omega, r_2, ID_S, PK_{S1}, PK_{S2}, L = (ID_{R_1}, \ldots, ID_{R_n}))$, $\mathcal{C}$ checks whether a tuple of the form $\langle h_3, m, \omega, r_2, ID_S, PK_{S1}, PK_{S2}, L = (ID_{R_1}, \ldots, ID_{R_n}), c, h_4\rangle$ exists in L_3. If it exists in L_3, then return h_3. Otherwise, select a random $h_3 \in_R \mathbb{Z}_q^*$ and return h_3. Also, for convenience in responding to the designcryption queries, compute h_4 by calling the H_4 oracle on (r_2, ID_S) and $c = m \oplus h_4$. Add $\langle h_3, m, \omega, r_2, ID_S, PK_{S1}, PK_{S2}, L = (ID_{R_1}, \ldots, ID_{R_n}), c, h_4\rangle$ to the list L_3.

H_4 **queries:** On input (r_2, ID), $\mathcal{C}$ returns the previously defined h_4 if a tuple of the form $\langle h_4, r_2, ID\rangle$ exists in the list L_4. Otherwise, it returns a random $h_4 \in_R \mathbb{Z}_q^*$ and stores the entry $\langle h_4, r_2, ID\rangle$ in L_4.

H_5 **queries:** On input $(g^r, g^{r.x_{R_i}}, PK_{R_i1}, ID_{R_i})$, $\mathcal{C}$ returns the previously defined h_5, if a tuple of the form $\langle h_5, g^r, g^{r.x_{R_i}}, PK_{R_{i1}}, ID_{R_i}\rangle$ exists in L_5. Otherwise,

it returns a random $h_5 \in_R \mathbb{Z}_q^*$ and stores the entry $\langle h_5, g^r, g^{r.x_{R_i}}, PK_{R_{i1}}, ID_{R_i}\rangle$ in L_5.

Request Public Key queries: On input ID_i, challenger returns the tuple $\langle PK_{i1}, PK_{i2}\rangle$ if an entry of the form $\langle i, ID_i, q_i, PK_{i1}, PK_{i2}, x_i, b_i\rangle$ exists in L_1. Otherwise, it retrieves the entry $\langle i, ID_i, q_i, \ldots\rangle$ from L_1 and selects a random $x_i \in_R \mathbb{Z}_q^*$ and computes $PK_{i1} = \hat{e}\,(P,Q)^{x_i}, PK_{i2} = x_i\,(q_i P + P_{pub})$. $\mathcal{C}$ returns the public key $\langle PK_{i1}, PK_{i2}\rangle$ and updates the tuple corresponding to ID_i in L_1 with PK_{i1}, PK_{i2}, x_i.

Replace Public Key queries: On input ID_i and a valid public key tuple $\langle PK'_{i1}, PK'_{i2}\rangle$, $\mathcal{C}$ checks whether public key fields of tuple coresponding to ID_i in L_1 is empty or not. If it is not empty, then $\mathcal{C}$ replaces the public keys in the tuple corresponding to ID_i in list L_1 with PK'_{i1}, PK'_{i2} and the marker bit b_i is set to 1. Else, if public key fields are empty then generate the public keys using *Request Public Key* oracle and then replace the public key of ID_i with PK'_{i1} and PK'_{i2} using *Replace Public Key* oracle.

Extract Partial Private Key queries: On input ID_i, if $(ID_i = ID_j^*, j = 1, \ldots, n)$, then abort. Otherwise, retrieve the tuple $\langle i, ID_i, q_i, \ldots\rangle$ from L_1 and return $\dfrac{t}{(q_i + s)}P$.

Extract Secret Key queries: On input ID_i, if $ID_i = ID_j^*, j = 1, \ldots, n$ then aborts. Otherwise, $\mathcal{C}$ retrieves the entry $\langle i, ID_i, q_i, PK_{i1}, PK_{i2}, x_i, b_i\rangle$ from L_1. If $b = 0$, then $\mathcal{C}$ returns $\left\langle \dfrac{t}{(q_i + s)}P, x_i \right\rangle$. Else, if b=1, public key of the identity ID_i has been replaced and $\mathcal{C}$ asks $\mathcal{A}$ for the secret key x'_i and returns the tuple accordingly.

Signcrypt queries: On input $(m, ID_S, L = ID_{R_1}, ID_{R_2}, \ldots, ID_{R_n})$. If $ID_S = ID_{R_i}$ $(i = 1, \ldots, n)$ or ($ID_S \in L^*$ and one or more of $ID_{R_i} \in L^*, i = 1, \ldots, n$), then abort. Else, if $ID_S \neq ID_j^*$ $(j = 1, \ldots, n)$, then $\mathcal{C}$ knows the secret key of the sender and does the computations as per the signcryption algorithm to return the ciphertext $\sigma = \langle c, d_1, d_2, \ldots, d_n, L\rangle$. Otherwise, if $ID_S = ID_j^*$ for $j \in \{1, \ldots, n\}$, $\mathcal{C}$ does not know the secret key of the sender and hence it generates the ciphertext as follows:

$\mathcal{C}$ retrieves the entry $\left\langle j, ID_j^*, q_j^*, PK_{j1}^*, PK_{j2}^*, x_j^*, b_j^*\right\rangle$ from L_1 and selects $r_2, \hat{r} \in_R \mathbb{Z}_q^*$. $\mathcal{C}$ computes h_4 by calling oracle H_4 with the input (r_2, ID_j^*) and $c = h_4 \oplus m$. $\mathcal{C}$ executes the following steps for each receiver ID_{R_i}:

1. Retrieves the tuple $\langle i, ID_{R_i}, q_i, PK_{i1}, PK_{i2}, x_i, b_i\rangle$ from L_1 and selects a random $h_{3i} \in_R \mathbb{Z}_q^*$.
2. Computes $Z_{R_i} = \dfrac{\hat{r}}{(q_i + s)}Q$, $d_{i1} = \hat{r}\left(x_j^* + h_{3i}\right).\left(q_j^* P + P_{pub}\right)$ and $\omega_i = \hat{e}\left(d_{i1}, \dfrac{t}{(q_i + s)}P\right)$.
3. Stores (r_2, ω_i, ID_S) in L_2 . Computes h_{5i} by calling H_5 oracle with $(\omega_i, \omega_i^{x_i}, PK_{i1}, ID_{R_i})$ as input and $d_{i2} = (r_2 \| Z_{R_i}) \oplus h_{5i}$. Note that if $b_i = 1$, then the public key of the receiver has been replaced and challenger asks $\mathcal{A}$ for the x'_i and uses it in place of the x value stored in the tuple. $\mathcal{C}$ sets $d_i = \langle d_{i1}, d_{i2}\rangle$.

4. Add $\langle h_{3i}, m, \omega_i, r_2, ID_S, PK_{S1}, PK_{S2}, L = (ID_{R_1}, \ldots, ID_{R_n}), c, h_{4i}\rangle$ in L_2. ($\mathcal{C}$ fails if H_3 is already defined on any of such entries, but this happens only with probability $\frac{(n.q_{sc}+q_{H_3})}{2^{k_2}}$). $\mathcal{C}$ sends $\sigma = \langle c, d_1, d_2, \ldots, d_n, L\rangle$ to $\mathcal{A}$.

Designcrypt queries: On input ciphertext σ, a sender's identity ID_S and a receiver's identity ID_R. $\mathcal{C}$ extracts c, $d_R = \langle d_{R_1}, d_{R_2}\rangle$ and L from σ. If $ID_R \neq ID_j^*$ $(j = 1, \ldots, n)$ then $\mathcal{C}$ knows the secret key of the receiver and hence it follows the *Designcrypt* algorithm for designcrypting σ. Otherwise, $\mathcal{C}$ searches in L_3 for all tuples
$\langle h_3', m', \omega', r_2', ID_S, PK_{S1}, PK_{S2}, L = (ID_{R_1}, \ldots, ID_{R_n}), c, h_4'\rangle$, having c and L. If no such tuple is found, then σ is rejected. Otherwise, it looks for the entry $\langle h_5, \omega', -, PK_{R1}, ID_R\rangle$ in L_5, reject if no such entry is found. Compute $r_2 \| Z_S = d_{R_2} \oplus h_5$. If $r_2 = r_2'$ and Z_S passes the verification test, then return m, else reject σ. Note that a valid ciphertext is rejected with probability at-most $q_{dsc}\left(\frac{1}{2^{k_2}} + \frac{1}{2^{k_1+k_2}}\right)$ across the game.

Challenge. $\mathcal{A}$ outputs two messages $\{m_0, m_1\}$ with an arbitrary sender's identity $ID_S (ID_S \neq ID_j^*, j = 1, \ldots, n)$ for which $\mathcal{A}$ wants $\mathcal{C}$ to generate the challenge ciphertext. $\mathcal{C}$ selects $c \in_R \{0,1\}^n$, $Z_S \in_R \mathbb{G}_1$, to return to $\mathcal{A}$ the signcryption $\sigma^* = \langle c, d_1, d_2, \ldots, d_n, L = \{ID_1^*, \ldots, ID_n^*\}\rangle$. $\mathcal{C}$ computes $d_j = \langle d_{j_1}, d_{j_2}\rangle$for $j = 1, \ldots, n$ as follows:

1. $\mathcal{C}$ chooses $\gamma_j \in \mathbb{Z}_q^*$. Sets $d_{j_1} = \gamma_j P$. If we define $\gamma_j = r_{1_j}.(q_j^* + s)$, then we can check that $d_{j_1} = r_{1_j}.(q_j^* + s)P$ has the proper form.
2. $\mathcal{C}$ chooses $d_{j_2} \in_R \{0,1\}^{k_1+k_2}$. Stores the pair $\left(\gamma_j, ID_j^*\right)$.

$\mathcal{A}$ cannot recognize that σ^* is not the proper signcryption until unless it queries H_3 or H_5 on $\omega_j = \hat{e}\,(P,Q)^{r_{1j}}$, $ID_j^* (j = 1, \ldots, n)$.

Phase2:$\mathcal{C}$ simulates $\mathcal{A}$'s queries as in Phase1, except that $\mathcal{A}$ is not allowed to ask *Designcrypt* queries on σ^* for $ID_j^* (j = 1, \ldots, n)$.

Guess: Finally, $\mathcal{A}$ outputs a bit, which is ignored by $\mathcal{C}$. Standard arguments can show that a successful $\mathcal{A}$ is very likely to query H_3 or H_5 on $\omega_j = \hat{e}\,(P,Q)^{r_{1j}}$, $ID_j^* (j = 1, \ldots, n)$ if the simulation is indistinguishable from a real attack environment.

$\mathcal{C}$ fetches a random entry $\langle h_3, m, \omega, r_2, ID_S, PK_{S1}, PK_{S2}, L = (ID_{R_1}, \ldots, ID_{R_n}), c, h_4\rangle$ or$\langle h_5, \omega, \omega^{x_i}, ., ID_i\rangle$ from lists L_3 or L_5.

With probability $\frac{1}{(2.n.q_{sc} + q_{H_3} + q_{H_5})}$(as L_3, L_5 contains no more than $n.q_{sc} + q_{H_3}, n.q_{sc} + q_{H_5}$ elements respectively by construction), the chosen entry contains the right elements $\omega_j = \hat{e}(P,Q)^{r_{1j}}, ID_j^*$. $\mathcal{C}$ retrieves the pair $\left(\gamma_j, ID_j^*\right)$. Now, $\mathcal{C}$ calculates

$$(\hat{e}(P,Q)^{r_{1j}})^{(\gamma_j.t)^{-1}} = \left(\hat{e}(P,Q)^{\gamma_j(q_j^*+s)^{-1}}\right)^{(\gamma_j.t)^{-1}} = \left(\hat{e}(P,P)^{(\gamma_j.t)(q_j^*+s)^{-1}}\right)^{(\gamma_j.t)^{-1}}$$

$= \hat{e}(P,P)^{(q_j^*+s)^{-1}}$. $\mathcal{C}$ returns $\left(q_j^*, \hat{e}(P,P)^{(q_j^*+s)^{-1}}\right)$ as its output.

Now, we analyze the success probability of $\mathcal{C}$. Let S be the event that $\mathcal{A}$ outputs the correct bit $b' = b$.

Simulation fails if any of the following event occurs:
1. E_1*:Extract Partial Private Key* query for some challenge Identity has been queried.
2. E_2*:Extract Secret Key* query for some challenge Identity queried.
3. E_3*:* Simulation is aborted because both sender and one or more of the receivers belong to the challenge set in some signcryption queries.
4. E_4*:* Simulation is aborted due to the H_3 collisions in *Signcryption* queries.
5. E_5*:* $\mathcal{C}$ rejects a valid ciphertext at some point of the game.
6. E_6*:* $\mathcal{C}$ chooses the correct tuple from L_3 or L_5.

We clearly have $Pr[S] = \epsilon$ and S implies $\neg E_1 \wedge \neg E_2 \wedge \neg E_3$.Also, we have $Pr[E_4] \leq \frac{q_{sc}\left(nq_{sc} + q_{H_3}\right)}{2^k}$ as there are a total of q_{sc} signcryption queries and atmost $nq_{sc} + q_{H_3}$ entries are there in L_3. It is already observed that $Pr[E_5] \leq q_{dsc}\left(\frac{1}{2^{k_2}} + \frac{1}{2^{k_1+k_2}}\right)$ as an invalid rejection occurs only when $\mathcal{A}$ is able to guess hash value of one of the H_2 or H_5 correctly. It has already been argued that $Pr[E_6] \leq \frac{1}{(2nq_{sc} + q_{H_3} + q_{H_5})}$. Now, the advantage ϵ of $\mathcal{C}$ is defined as.

$$\epsilon' = Pr[S \wedge \neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4 \wedge \neg E_5 \wedge E_6]$$

Therefore, we obtain the bound

$$\epsilon' > \frac{\epsilon}{(2nq_{sc} + q_{H_3} + q_{H_5})}\left(1 - \frac{q_{sc}\left(nq_{sc} + q_{H_3}\right)}{2^{k_2}}\right)\left(1 - q_{dsc}\left(\frac{1}{2^{k_2}} + \frac{1}{2^{k_1+k_2}}\right)\right).$$

Also, as we have to compute atmost n parings, n scalar multiplications, n exponentiations for simulating signcryption queries, atmost 1 parings,1 scalar multiplications,1 exponentiations for simulating each public key request query. Similarly, atmost 1 paring,1 scalar multiplication and 1 exponentiation is required for simulating designcryption queries. n scalar multiplications are needed during Challenge phase. So we obtain the desired bound on the time τ'

$$\tau' < \tau + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_p + O\left(q_{pk} + nq_{sc} + q_{dsc} + n\right).t_{sm} + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_e$$

□

Theorem 2. *If an IND-CLMSC-iCCA2-II adversary* $\mathcal{A}$ *has advantage* ϵ *against our scheme running in time* τ *and asking* q_{H_i} $(i = 1, 2, 3, 4, 5)$ *hash queries to random oracles* H_i $(i = 1, 2, 3, 4, 5)$*,* q_{sc} *signcryption queries,* q_{dsc} *designcryption queries,* q_{ske} *extract secret key queries,* q_{ppe} *partial private key extract queries,* q_{pk} *public key request queries and* q_{pkr} *public key replacement queries, then there exist an algorithm* $\mathcal{C}$ *that solves the GBDHP with advantage* ϵ' *and time* τ'

$$\epsilon' > \epsilon\left(1 - \frac{q_{sc}\left(nq_{sc} + q_{H_3}\right)}{2^{k_2}}\right)\left(1 - q_{dsc}\left(\frac{1}{2^{k_2}} + \frac{1}{2^{k_1+k_2}}\right)\right)$$

$$\tau' < \tau + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_p + O\left(q_{pk} + nq_{sc} + q_{dsc} + n\right).t_{sm} + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_e$$

with atmost $(nq_{sc} + q_{H_5})$ *calls to DBDH oracle* $\mathcal{O}_\Gamma$. *Here* t_p, t_{sm}, t_e *are the costs of pairing computation, scalar multiplication in* $\mathbb{G}_1$ *and exponentiation in* $\mathbb{G}_2$, n *is the number of receivers in the challenge set and* k_1, k_2 *are the number of bits needed to represent elements of* $\mathbb{G}_1$ *and* $\mathbb{Z}_q^*$ *respectively.*

Proof. Proof is omitted due to page limitation and is given in the full version of this paper.

Theorem 3. *If an sEUF-CLMSC-iCMA-I adversary* $\mathcal{F}$ *has advantage* ϵ *against our scheme running in time* τ *and asking* q_{H_i} $(i = 1, 2, 3, 4, 5)$ *hash queries to random oracles* H_i $(i = 1, 2, 3, 4, 5)$, q_{sc} *signcryption queries,* q_{dsc} *designcryption queries,* q_{ske} *extract secret key queries,* q_{ppe} *partial private key extract queries,* q_{pk} *public key request queries and* q_{pkr} *public key replacement queries, then there exist an algorithm* $\mathcal{C}$ *that solves the* $k - CAA$ *problem for* $k = q_{H_1}$ *with an advantage*

$$\epsilon' > \frac{1}{n}\left(\epsilon - \frac{n}{2^{k_2}} - \frac{n}{2^{k_1+k_2}}\right).\left(1 - \frac{q_{sc}\left(nq_{sc} + q_{H_3}\right)}{2^{k_2}}\right)\left(1 - q_{dsc}\left(\frac{1}{2^{k_2}} + \frac{1}{2^{k_1+k_2}}\right)\right)$$

and with time

$$\tau' < \tau + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_p + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_{sm} + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_e$$

where t_p, t_{sm}, t_e *are the cost of pairing computation, scalar multiplication in* $\mathbb{G}_1$ *and exponentiation in* $\mathbb{G}_2$, n *is the number of receivers in the challenge set and* k_1, k_2 *are the number of bits needed to represent elements of* $\mathbb{G}_1$ *and* $\mathbb{Z}_q^*$ *respectively.*

Proof. Proof is omitted due to page limitation and is given in the full version of this paper.

Theorem 4. *If an sEUF-CLMSC-iCMA-II adversary* $\mathcal{F}$ *has advantage* ϵ *against our scheme running in time* τ *and asking* q_{H_i} $(i = 1, 2, 3, 4, 5)$ *hash queries to random oracles* H_i $(i = 1, 2, 3, 4, 5)$, q_{sc} *signcryption queries,* q_{dsc} *designcryption queries,* q_{ske} *extract secret key queries,* q_{ppe} *partial private key extract queries,* q_{pk} *public key request queries and* q_{pkr} *public key replacement queries, then there exist an algorithm* $\mathcal{C}$ *that solves the SDH problem with advantage*

$$\epsilon' > \frac{1}{n}\left(\epsilon - \frac{n}{2^{k_2}} - \frac{n}{2^{k_1+k_2}}\right).\left(1 - \frac{q_{sc}\left(nq_{sc} + q_{H_3}\right)}{2^{k_2}}\right)\left(1 - q_{dsc}\left(\frac{1}{2^{k_2}} + \frac{1}{2^{k_1+k_2}}\right)\right)$$

and time

$$\tau' < \tau + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_p + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_{sm} + O\left(q_{pk} + nq_{sc} + q_{dsc}\right).t_e$$

where t_p, t_{sm}, t_e are the cost of pairing computation, scalar multiplication in $\mathbb{G}_1$ and exponentiation in $\mathbb{G}_2$, n is the number of receivers in the challenge set and k_1, k_2 are the number of bits needed to represent elements of $\mathbb{G}_1$ and $\mathbb{Z}_q^*$ respectively.

Proof. Proof is omitted due to page limitation and is given in the full version of this paper.

6 Efficiency Analysis and Comparison

In this section, we compare the efficiency of our scheme(CLMSC) with the related schemes. [12] is the only certificateless signcryption scheme available till date and no certificateless multi-receiver signcryption scheme is available. Hence we compare our scheme with the naive extension of [12] for multi-receiver setting in *Table* 1. Also, we have [10] and [13] to be the two recent efficient identity based multi-receiver signcryption schemes. We compare the efficiency of our scheme with [10] and [13] in *Table* 2. However, it should be noted that the ciphertext size in our scheme is almost twice the ciphertext size of [13].

Table 1. Efficiency Comparison with [12]

Scheme	Signcrypt				Designcrypt			
	PA	SM	GE	MG	PA	SM	GE	MG
BF_CLSC(SR)	1	4	1	2	5	1	-	2
BF_CLSC(MR)	n	3n+1	n	3n+1	5	1	-	2
CLMSC(SR)	-	2	2	-	2	1	1	-
CLMSC(MR)	-	n	n+1	-	2	1	1	-

BF_CLSC(SR) - Single Receiver [12]
BF_CLSC(MR) - Naive Extension of [12] to Multi-receiver
CLMSC(SR) - CLMSC Single Receiver
CLMSC(MR) - CLMSC Multi-receiver

Table 2. Efficiency Comparison with [10] and [13]

Scheme	Signcrypt				Designcrypt			
	PA	SM	GE	MG	PA	SM	GE	MG
DC_IBMSC	1	4+n	-	n+2	4	1	-	1
SS_IBMSC	1	3+n	1	n+1	4	1	-	-
CLMSC	-	n	n+1	-	2	1	1	-

DC_IBMSC - Identity based multi-receiver signcryption in [10]
SS_IBMSC - Identity based multi-receiver signcryption in [13]
CLMSC - CLMSC Multi-receiver

7 Conclusion

We have presented an efficient and provably secure certificateless multi-receiver signcryption scheme and a security model for the same. We have proved the security of our scheme in random oracle model under insider attacks. In our security model, we have considered the selective multi-identity attack for confidentiality and unforgeability. In the proposed model, we have considered the type-I adversary which is not allowed to query the *partial private key* oracle for the challenge identities. In the literature, a different type of type-I adversary (say type-I') has also been considered. Type-I' adversary can query either the *Replace Public Key* oracle or *Partial Private Key Extract* oracle for the challenge identities. It will be interesting to see whether such type of an adversary can be considered in the case of multi-receiver signcryption also. We have proved the security of our scheme in random oracle model against the selective identity attack. Designing certificateless multi-receiver signcryption schemes which can be proved secure in standard model against adaptive identity attacks is another interesting open problem.

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public-Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Yu, Y., Yang, B., Huang, X., Zhang, M.: Efficient identity-based signcryption scheme for multiple receivers. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 13–21. Springer, Heidelberg (2007)
3. Zheng, Y.: Digital signcryption or How to achieve cost (signature & Encrytpion) $\ll$ cost(signature) + cost(encryption). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)
4. Malone-Lee, J., Mao, M.: Two birds one stone: signcryption using RSA. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 211–226. Springer, Heidelberg (2003)
5. Malone-Lee, J.: Identity based signcryption. Cryptology ePrint Archive. Report 2002/098 (2002)
6. Libert, B., Quisquator, J.J.: A new identity based signcryption scheme from pairings. In: 2003 IEEE information theory workshop, Paris, France, pp. 155–158 (2003)
7. Boyen, X.: Multipurpose identity based signcryption: a swiss army knife for identity based cryptography. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 383–399. Springer, Heidelberg (2003)
8. Barreto, P.S.L.M., Libert, B., McCullagh, N., Quisquater, J.J.: Efficient and provably-secure identity based signatures and signcryption from bilinear maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 515–532. Springer, Heidelberg (2005)
9. Zheng, Y.: Signcryption and its applications in efficient public key solutions. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 291–312. Springer, Heidelberg (1998)
10. Duan, S., Cao, Z.: Efficient and provably secure multi-receiver identity-based signcryption. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 195–206. Springer, Heidelberg (2006)

11. Kurosawa, K.: Multi-recipient public-key encryption with shortened ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002)
12. Barbosa, M., Farshim, P.: Certificateless Signcryption. In: Conference on Computer and Communications Security archive Proceedings of the, ACM symposium on Information, Computer and Communications Security (2008)
13. Selvi, S.S.D., Vivek, S.S., Gopalakrishnan, R., Karuturi, N.N., Rangan, C.P.: Cryptanalysis of ID-Based Signcryption Scheme for Multiple Receivers, http://eprint.iacr.org/2008/238.pdf
14. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
15. Baek, J., Naini, R., Susilo, W.: Efficient Multi-Receiver Identity-Based Encryption and Its Application to Broadcast Encryption. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 1611–3349. Springer, Heidelberg (2006)
16. Dutta, R., Barua, R., Sarkar, P.: Pairing-Based Cryptographic Protocols: A Survey, http://eprint.iacr.org/2004/064.pdf
17. Baek, J., Steinfeld, R., Zheng, Y.: Formal proofs for the security of signcryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 80–98. Springer, Heidelberg (2002)
18. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
19. Dent, A.W.: A Survey of Certificateless Encryption Schemes and Security Model, eprint.iacr.org/2006/211.pdf
20. Dent, A.W., Libert, B., Paterson, K.G.: Certificateless Encryption Schemes Strongly Secure in the Standard Model. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 344–359. Springer, Heidelberg (2008)
21. Libert, B., Quisquate, J.-J.: On Constructing Certificateless Cryptosystems from Identity Based Encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006)
22. Chow, S.S.M., Hui, L.C.K., Yiu, S.M., Chow, K.P.: Two Improved Partially Blind Signature Schemes from Bilinear Pairings. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 316–328. Springer, Heidelberg (2005)

A CCA Secure Hybrid Damgård's ElGamal Encryption

Yvo Desmedt[1,⋆] and Duong Hieu Phan[2,⋆⋆]

[1] University College London
Gower Street, London WC1E 6BT, United Kingdom
[2] University of Paris 8
2, rue de la Libertè 93526 - Saint-Denis cedex 02, France
hieu.phan@univ-paris8.fr

Abstract. ElGamal encryption, by its efficiency, is one of the most used schemes in cryptographic applications. However, the original ElGamal scheme is only provably secure against passive attacks. Damgård proposed a slight modification of ElGamal encryption scheme (named Damgård's ElGamal scheme) that provides security against non-adaptive chosen ciphertext attacks under a knowledge-of-exponent assumption. Recently, the CCA1-security of Damgård's ElGamal scheme has been proven under more standard assumptions.

In this paper, we study the open problem of CCA2-security of Damgård's ElGamal. By employing a data encapsulation mechanism, we prove that the resulted hybrid Damgård's ElGamal Encryption is secure against adaptive chosen ciphertext attacks. The down side is that the proof of security is based on a knowledge-of-exponent assumption. In terms of efficiency, this scheme is more efficient (e.g. one exponentiation less in encryption) than Kurosawa-Desmedt scheme, the most efficient scheme in the standard model so far.

1 Introduction

ElGamal encryption was introduced by ElGamal in 1985 [7] as a variant of Diffie-Hellman key exchange. Since then, it has become one of the two most extensively used for cryptographic applications, besides RSA [17].

The notion of *semantic security*, introduced by Goldwasser and Micali [11], captures the intuition that an adversary should not be able to obtain any partial information about the underlying plaintext of a *challenge* ciphertext. At the same time, various kinds of attack have been modeled and the strongest formalized attack is a chosen ciphertext attack where the adversary is given access to a *decryption oracle* that allows him to obtain the decryptions of his chosen

⋆ The author is BT Professor of Information Security and is also funded by EPSRC EP/C538285/1.

⋆⋆ Part of this research was done while being at University College London, Adastral Campus.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 68–82, 2008.

ciphertexts (with a natural restriction that these chosen ciphertexts are different from the challenge ciphertext). There are two kinds of chosen ciphertext attack: *non-adaptive attacks* or *lunchtime attacks* [15], denoted CCA1, where the access to the decryption oracle is limited until the challenge is known; and *adaptive chosen-ciphertext attack* [16], denoted CCA2 or CCA where adversaries have access to decryption oracle before and after receiving the challenge.

The original ElGamal scheme was proven to be semantically secure against passive adversaries and there was the open question whether it is secure against CCA1 attacks.

In [5] Damgård proposed a variant of the ElGamal encryption scheme (later called Damgård's ElGamal Encryption), by only adding an exponentiation to ciphertexts, and provided a proof of security against non-adaptive chosen ciphertext attacks. This proof requires however an informal assumption of *knowledge-of-exponent* assumption. Later, Bellare and Palacio [2] formalized this notion, calling it the *Diffie-Hellman knowledge* (DHK for short) assumption, and provided a formal proof of security against non-adaptive chosen ciphertext attacks for Damgård's ElGamal Encryption. Dent [6] has shown that the DHK assumption holds in generic groups. Recently, Gjøsteen [10], Wu and Stinson [21], Lipmaa [14] improved the security results of Damgård's ElGamal Encryption but all of these works only concern CCA1 security.

The Damgård's ElGamal encryption is obviously not CCA secure because it is homomorphic. An important problem is thus to consider a simple variant of Damgård's ElGamal encryption that could achieve CCA security, as this security level is well-admitted to be the standard notion for confidentiality in cryptography. We investigate in this paper the natural issue of using Damgård's ElGamal encryption in a hybrid framework.

A hybrid encryption scheme [20] employs public-key encryption techniques to derive a shared key that is then used to encrypt the actual messages using symmetric-key techniques. It can thus take advantage of symmetric encryption to encrypt arbitrary long messages. In [4], a formal treatment of hybrid schemes was proposed, composed of two parts : first, a KEM (Key Encapsulation Mechanism) is invoked to encrypt a random key; second, a DEM (Data Encapsulation Mechanism) is performed to encrypt messages using a symmetric encryption scheme. Kurosawa-Desmedt [13] proposed later a hybrid variant of the Cramer-Shoup scheme [3] in the KEM/DEM framework which results in the most efficient scheme in the standard model so far.

Contribution. We propose a hybrid Damgård's ElGamal Encryption, *i.e.*, embedding a DEM in the Damgård's ElGamal Encryption, and show that this scheme achieves CCA security. The proposed scheme is more efficient than the Kurosawa-Desmedt scheme in many aspects: it requires one less exponentiation, it needs no target collision resistance, it produces shorter secret keys and public keys.

The security proof for our proposed hybrid Damgård's ElGamal scheme has to be however based on an extension of DHK assumption (which is an another

formalization of the KEA3 assumption in [1]). We give a proof of the hardness of this assumption in generic groups. Although this kind of proof in generic groups can be seen as a minimum requirement for the use of an assumption in a security proof, we strongly believe that the hybrid Damgård's ElGamal scheme would be very useful in practice and deserves to be investigated further.

2 Notation and Standard Definitions

In this section, we recall the formalization of the most important security notion for asymmetric encryption, namely the CCA security.

Firstly, let us briefly remind that a public-key encryption scheme π is defined by three algorithms: the key generation algorithm $\mathcal{K}(1^\lambda)$, which on the security parameter λ, produces a pair of matching public and private keys $(\mathsf{pk}, \mathsf{sk})$; the encryption algorithm $\mathcal{E}_{\mathsf{pk}}(m; r)$ which outputs a ciphertext c corresponding to the plaintext $m \in \mathcal{M}$, using random coins $r \in \mathcal{R}$; and the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(c)$ which outputs the plaintext m associated to the ciphertext c.

Security Notions. *Semantic security* (*a.k.a. polynomial security* or *indistinguishability of encryptions* [11], denoted IND) is considered to be the standard notion: if the attacker has some *a priori* information about the plaintext, it should not learn more from the view of the ciphertext. More formally, this security notion requires the computational indistinguishability between two strings, chosen by the adversary, one of which has been encrypted. The adversary needs to decide which one has been actually encrypted with a probability significantly higher than one half: the advantage $\mathsf{Adv}^{\mathsf{ind}}_{\pi}(\mathcal{A})$, where the adversary $\mathcal{A}$ is seen as a 2-stage Turing machine $(\mathcal{A}_1, \mathcal{A}_2)$ whose running time is bounded by a polynomial function in λ, should be a negligible function in λ. Formally, with λ the security parameter:

$$\mathsf{Adv}^{\mathsf{ind}}_{\pi}(\mathcal{A}) = 2 \times \Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^\lambda), (m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ b \overset{R}{\leftarrow} \{0,1\}, c = \mathcal{E}_{\mathsf{pk}}(m_b) : \mathcal{A}_2(m_0, m_1, s, c) = b\end{array}\right] - 1.$$

On the other hand, an attacker can employ various kinds of attacks, depending on thee available information. Since we are considering asymmetric encryptions, the adversary can encrypt any plaintext of its choice with the public key, hence the basic *chosen-plaintext attack*. The strongest formalized attack can have a polynomial access to the decryption oracle (except the challenge ciphertext $c = \mathcal{E}_{\mathsf{pk}}(m_b)$), *adaptive chosen-ciphertext attacks* [16], denoted CCA or CCA2 (by opposition to the earlier *non-adaptive chosen-ciphertext attacks* or *lunchtime attacks* or [15], denoted CCA1, where the access to the decryption oracle is limited until the challenge is known.)

The strongest security notion for asymmetric encryptions is thus the *semantic security against adaptive chosen-ciphertext attacks* denoted IND-CCA or CCA security. In this paper, as we deal with this security notion, the adversary $\mathcal{A}$ is allowed to access the decryption oracle.

Denote $max_{\mathcal{A}}(\mathsf{Adv}^{\mathsf{ind}}_{\pi}(\mathcal{A}))$ by $\mathsf{Adv}^{\mathsf{ind-cca}}_{\pi}(\lambda)$. The CCA security of π requires that $\mathsf{Adv}^{\mathsf{ind-cca}}_{\pi}(\lambda)$ is a negligible function in λ.

3 Construction

In the following sections, one assumes having a group generator $\mathcal{G}$ which, on input 1^λ, returns (g, q) satisfying that g is a generator of a cyclic group $\mathcal{G}_q = \langle g \rangle$ of prime order q, and $2^{\lambda-1} < q < 2^\lambda$.

3.1 Damgård's ElGamal Encryption [5]

First of all, we recall Damgård's ElGamal Encryption.

Key Generation $\mathcal{K}(1^\lambda)$**:** $(g, q) \leftarrow \mathcal{G}(1^\lambda)$, random elements $x, y \in \mathbb{Z}_q$ are also chosen.
Secret Key: $\mathsf{sk} = (x, y)$
Public Key: $\mathsf{pk} = (g, c = g^x, d = g^y)$

Encryption: Given a message m, the encryption runs as follows. First, it chooses $r \overset{R}{\leftarrow} \mathbb{Z}_q$ and then computes:

$$u_1 = g^r, u_2 = c^r, e = d^r \cdot m$$

The outputted ciphertext is (u_1, u_2, e)

Decryption: Given a ciphertext (u_1, u_2, e), the decryption runs as follows. First, it tests if $u_2 = u_1^x$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise it computes:

$$K = H(u_1^y), \quad m = K^{-1} \cdot e$$

and outputs m.

3.2 Hybrid Damgård's ElGamal Encryption

We now propose a hybrid variant of the above Damgård's ElGamal Encryption. We just replace, in the encryption of Damgård's ElGamal scheme, the operation $e = d^r \cdot m$ by a symmetric encryption of m under the key $K = H(d^r)$, where H is randomly chosen from a universal familly of hash functions. The detailed description follows.
In our scheme, we make use of:

- a Data Encapsulation Mechanism (DEM). A DEM is a symmetric key encryption scheme, with encryption algorithm E and decryption algorithm D, such that for the key $K \in \mathcal{K}_D$ ($\mathcal{K}_D$ is key space of DEM whose size depends on λ) and the plaintext $m \in \{0,1\}^\star$, $e := E_K(m)$ is the encryption of m under K, and for key $K \in \mathcal{K}_D$ and the ciphertext $e \in \{0,1\}^\star$, $m := D_K(m)$ is the decryption of e under K.
  For our purpose, we require that DEM is CCA secure, *i.e.* the advantage to distinguish $E_K(m_0)$ from $E_K(m_1)$ should be a negligible function in λ, (*i.e.* given the challenge ciphertext $E_K(m_b)$ for a random bit b, hard to guess b) for randomly chosen K and adversarially chosen m_0 and m_1 (where

m_0 and m_1 are of equal length but different), even though the adversary has access to the decryption oracle for its chosen ciphertext (except the challenge ciphertext). We denote this advantage by $\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{dem}}(\lambda)$.
Unlike the case of public-key, secure symmetric encryption schemes against chosen-ciphertext attacks can be easily built out of weaker primitives: all one needs is a secure symmetric encryption scheme against passive adversaries, and a secure message authentication code (MAC).

– a key derivation function H which maps an element $V \in \mathcal{G}_q$ to $H(V) \in \mathcal{K}_D$. We require that it's hard to distinguish $H(V)$ from K', where V and K' are both randomly chosen. We can thus use H as a function from a universal family of hash functions $\mathcal{H}_\lambda$and let $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$. In other words, we denote by $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{H}}(\lambda)$ the maximum advantage of all adversaries, whose running times are bounded by a polynomial function in λ, to distinguish $H(V)$ from K', where V and K' and $H \in \mathcal{H}_\lambda$ are randomly chosen, and assume that $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{H}}(\lambda)$ is a negligible function in λ. However, we will need some more requirements about H which will be described in Section 4.

We now describe our proposed scheme:

Key Generation $\mathcal{K}(1^\lambda)$**:** $(g, q) \leftarrow \mathcal{G}(1^\lambda)$, random elements $x, y \in \mathbb{Z}_q$ are also chosen. Next a hash function H is randomly chosen from a universal family of hash functions $\mathcal{H}_\lambda$.
Secret Key: $\mathsf{sk} = (x, y)$
Public Key: $\mathsf{pk} = (H, g, c = g^x, d = g^y)$

Encryption: Given a message m, the encryption runs as follows. First, it chooses $r \overset{R}{\leftarrow} \mathbb{Z}_q$ and then it computes:

$$u_1 = g^r, u_2 = c^r, K = H(d^r), e = E_K(m)$$

The ciphertext is (u_1, u_2, e)

Decryption: Given a ciphertext (u_1, u_2, e), the decryption runs as follows. First, it tests if $u_2 = u_1^x$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise it computes:

$$K = u_1^y, \quad m = D_K(e)$$

and outputs m.

3.3 Comparision with Kurosawa-Desmedt Scheme

We briefly recall the Kurosawa-Desmedt scheme which makes use of:

– a CCA secure DEM.
– a key derivation function H which is a universal one-way hash function [9].
– a target collision resistance function $\mathsf{TCR} : \mathcal{G}_q \times \mathcal{G}_q \to \mathbb{Z}_q$: given $u_1 := g^{r_1}$ and $u_2 := g^{r_2}$, for random $r_1, r_2 \in \mathbb{Z}_q$, it is hard to find $(u_1', u_2') \in \mathcal{G}_q \times \mathcal{G}_q \setminus \{(u_1, u_2)\}$ such that $H(u_1', u_2') = H(u_1, u_2)$

We now describe their scheme: one uses a group generator $\mathcal{G}'$, which on input 1^λ, returns (g_1, g_2, q) where g_1, g_2 are two generators of the cyclic group $\langle g \rangle$ of order q, and $2^{\lambda-1} < q < 2^\lambda$.

Key Generation $\mathcal{K}(1^\lambda)$**:** $(g_1, g_2, q) \leftarrow \mathcal{G}(1^\lambda)$, random elements $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ are also chosen.
Secret Key: $\mathsf{sk} = (x_1, x_2, y_1, y_2)$
Public Key: $\mathsf{pk} = (H, \mathsf{TCR}, g_1, g_2, c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2})$

Encryption: Given a message m, the encryption runs as follows. First, it chooses $r \overset{R}{\leftarrow} \mathbb{Z}_q$ and then it computes:

$$u_1 = g_1^r, u_2 = g_2^r, \alpha = \mathsf{TCR}(u_1, u_2)$$

$$v = c^r d^{r\alpha}, K = H(v), e = E_K(m)$$

The ciphertext is (u_1, u_2, e)

Decryption: Given a ciphertext (u_1, u_2, e), the decryption runs as follows. First, it computes

$$\alpha = \mathsf{TCR}(u_1, u_2), v = u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha}, K = H(v)$$

and then $m = D_K(e)$ (m may be "reject").

Comparison. In terms of efficiency, our scheme has the following advantages over the Kurosawa-Desmedt one:

Key Size: The secret key contains two group elements, compared to four group elements in the Kurosawa-Desmedt scheme. Our scheme needs one group generator, compared to two group generators in Kurosawa-Desmedt scheme and does not need to use the TCR function, the public key is thus shorter.

Encryption Computation: We get rid of the TCR function. Moreover, in our scheme $v = d^r$ while in the Kurosawa-Desmedt scheme $v = c^r d^{r\alpha}$. We can thus save one exponentiation. A secondary advantage is that, in our scheme, the values u_1, u_2, d could be computed in parallel, while in the Kurosawa-Desmedt scheme, one should first compute u_1, u_2, then compute $\alpha = \mathsf{TCR}(u_1, u_2)$, and finally compute v.

Decryption Computation: We also get rid of the computation of the TCR function. Otherwise, both schemes need two exponentiations. However, in our scheme, as all exponentiations are with respect to the same base, the algorithm can be executed faster.

4 Assumptions Used in the Security Analyses

4.1 Hashed Decisional Diffie-Hellman Assumption

We first recall the definition of the Hashed Diffie-Hellman (HDDH) problem needed for the sake of this work.

Assumption 1 (Hashed Decisional Diffie-Hellman Assumption) *Assume that there is no adversary that can effectively distinguish the two following distributions:*

- $(g, q) \leftarrow \mathcal{G}(1^\lambda)$ *and a hash function* H *;*
- *the distribution* R_H *of random elements in* $\mathcal{G}_q^4$*:* $(g, g^a, g^b, H(Z))$*, for randomly distributed* $a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q, Z \stackrel{R}{\leftarrow} \mathcal{G}_q$*;*
- *the distribution* D_H *of tuples elements in* $\mathcal{G}_q^4$*:* $(g, g^a, g^b, H(g^{ab}))$*, for randomly distributed* $a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q$*.*

In other words, for all probabilistic algorithms $\mathcal{A}$ *whose running times are bounded by a polynomial function in* λ*, we define* $\mathsf{Adv}_{\mathcal{G}}^{\mathsf{hddh}}(\mathcal{A})$ *the advantage that* $\mathcal{A}$ *can distinguish the two above distributions. The* HDDH *assumption states that* $\mathsf{Adv}_{\mathcal{G}}^{\mathsf{hddh}}(\lambda) = max_{\mathcal{A}}(\mathsf{Adv}_{\mathcal{G}}^{\mathsf{hddh}}(\mathcal{A}))$ *is a negligible function in* λ*.*

For our scheme, we need to use the following variant of the HDDH assumption:

Assumption 2 (Modified Hashed Decisional Diffie-Hellman: MHDDH**)** *Assume that there is no adversary that can effectively distinguish the two following distributions:*

- $(g, q) \leftarrow \mathcal{G}(1^\lambda)$ *and a hash function* H*;*
- *the distribution* R_{MH} *of random elements in* $\mathcal{G}_q^6$*:* $(g, g^a, g^b, g^c, g^{ac}, H(Z))$*, for randomly distributed* $a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_q, Z \stackrel{R}{\leftarrow} \mathcal{G}_q$*;*
- *the distribution* D_{MH} *of tuples elements in* $\mathcal{G}_q^6$*:* $(g, g^a, g^b, g^c, g^{ac}, H(g^{bc}))$*, for randomly distributed* $a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_q$*.*

In other words, for all probabilistic algorithms $\mathcal{A}$ *whose running times are bounded by a polynomial function in* λ*, we define* $\mathsf{Adv}_{\mathcal{G}}^{\mathsf{mhddh}}(\mathcal{A})$ *the advantage that* $\mathcal{A}$ *can distinguish the two above distributions. The* MHDDH *assumption states that* $\mathsf{Adv}_{\mathcal{G}}^{\mathsf{mhddh}}(\lambda) = max_{\mathcal{A}}(\mathsf{Adv}_{\mathcal{G}}^{\mathsf{mhddh}}(\mathcal{A}))$ *is a negligible function in* λ*.*

Proposition 3. *The* MHDDH *and the* HDDH *assumptions are equivalent:*

$$\mathsf{Adv}_{\mathcal{G}}^{\mathsf{hddh}}(\lambda) = \mathsf{Adv}_{\mathcal{G}}^{\mathsf{mhddh}}(\lambda)$$

Proof. The proof is quite trivial:

- Suppose that there is a MHDDH distinguisher for R_{MH} and D_{MH}, we construct an HDDH distinguisher for R_H and D_H as follows. Given an instance (g, g^b, g^c, Z), one randomly chooses $a \stackrel{R}{\leftarrow} \mathbb{Z}_q$, computes g^a, g^{ac}, then gives $(g, g^a, g^b, g^c, g^{ac}, Z)$ to MHDDH distinguisher and finally outputs what the MHDDH distinguisher returns. Thus:

$$\mathsf{Adv}_{\mathcal{G}}^{\mathsf{hddh}}(\lambda) \leq \mathsf{Adv}_{\mathcal{G}}^{\mathsf{mhddh}}(\lambda)$$

- Inversely, suppose that there is a HDDH distinguisher for R_H and D_H, we construct an MHDDH distinguisher for R_{MH} and D_{MH} as follows. Given an

instance $(g, g^a, g^b, g^c, g^{ac}, U)$, one just gives (g, g^a, g^c, U) to HDDH distinguisher and outputs what the HDDH distinguisher returns. Thus:

$$\mathsf{Adv}^{\mathsf{mhddh}}_{\mathcal{G}}(\lambda) \leq \mathsf{Adv}^{\mathsf{hddh}}_{\mathcal{G}}(\lambda) \qquad \square$$

We also need to introduce another assumption about the function H.

Assumption 4 (Extended Hashed Decisional Diffie-Hellman: EHDDH) *Assume that there is no adversary that can effectively distinguish the two following distributions R_{EH} and D_{EH} :*

- $(g, q) \leftarrow \mathcal{G}(1^\lambda)$ *and a hash function* H*;*
- *an element* $U \in \mathcal{G}_q, U \neq 1$ *and an element* $v \in \mathbb{Z}_q^\star$ *adversarially chosen;*
- *the distribution* R_{EH} *of random elements in* $\mathcal{G}_q^4$*:* $(g, g^a, g^b, H(g^{ab}), H(Z))$*, for randomly distributed* $a, b \overset{R}{\leftarrow} \mathbb{Z}_q, Z \overset{R}{\leftarrow} \mathcal{G}_q$*;*
- *the distribution* D_{EH} *of tuples elements in* $\mathcal{G}_q^6$*:* $(g, g^a, g^b, H(g^{ab}), H(Ug^{abv}))$*, for randomly distributed* $a, b \overset{R}{\leftarrow} \mathbb{Z}_q$*.*

In other words, for all probabilistic algorithms $\mathcal{A}$ whose running times are bounded by a polynomial function in λ, we define $\mathsf{Adv}^{\mathsf{ehddh}}_{\mathcal{G}}(\mathcal{A})$ the advantage that $\mathcal{A}$ can distinguish the two above distributions. The EHDDH *assumption states that $\mathsf{Adv}^{\mathsf{ehddh}}_{\mathcal{G}}(\lambda) = max_{\mathcal{A}}(\mathsf{Adv}^{\mathsf{ehddh}}_{\mathcal{G}}(\mathcal{A}))$ is a negligible function in λ.*

4.2 Diffie-Hellman Knowledge Assumption (DHK)

We now recall the definition of Diffie-Hellman Knowledge Assumptions.

For $A, B, C \in \mathcal{G}_q$, we say that (A, B, C) is a DH-triple if there exists $a, b \in \mathbb{Z}_q$ such that $A = g^a, B = g^b$ and $C = g^{ab}$. We say that (B, C) is a DH-pair relative to A if (A, B, C) is a DH-triple (throughout the text, when we say (A, B, C) is a DH-triple, we assume the base g to be fixed and known). We also write $C = DH(A, B)$. One way for an adversary $\mathcal{A}$ taking input g, A to output a DH-pair (B, C) relative to A is to pick (and thus "know") some $b \in \mathbb{Z}_q$, set $B = g^b$ and $C = A^b$, and output (B, C). Damgård [5] makes an assumption which, informally, implies that this is the "only" way that a polynomial-time adversary $\mathcal{A}$, given (g, A), can output a DH-pair (B, C) relative to A.

Bellare and Palacio [2] provide a formalization of this assumption that we refer to as the DHK (DHK stands for Diffie-Hellman Knowledge) assumption: the adversary $\mathcal{A}$, given (g, A), interacts with an extractor $\mathcal{A}^\star$, querying it adaptively, where $\mathcal{A}^\star$ is an algorithm that takes a pair of group elements and some state information (the state of $\mathcal{A}^\star$ is denoted by $St[\mathcal{A}^\star]$), and returns an exponent and a new state.

Experiment $\mathsf{Exp}^{\mathsf{dhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$
Below, $\mathcal{A}, \mathcal{A}^\star$ are polynomial-time probabilistic algorithms (whose running times are bounded by a polynomial function in λ).

– $(g, q) \leftarrow \mathcal{G}(1^\lambda)$; $a \stackrel{R}{\leftarrow} \mathbb{Z}_q, A = g^a$.
– Choose coins $R[\mathcal{A}], R[\mathcal{A}^\star]$ for $\mathcal{A}; \mathcal{A}^\star$, respectively ; $St[\mathcal{A}^\star] \leftarrow ((g, A), R[\mathcal{A}])$.
– Run $\mathcal{A}$ on input g, A and coins $R[\mathcal{A}]$ until it halts, replying to its oracle queries as follows:
  If $\mathcal{A}$ makes query (B, C) then:
  • $(b, St[\mathcal{A}^\star]) \leftarrow \mathcal{A}^\star((B, C), St[\mathcal{A}^\star]; R[\mathcal{A}^\star])$
  • If $C = B^a$ and $B \neq g^b$ then return 1,
    else return b to $\mathcal{A}$ as the reply.
– Return 0

Assumption 5 (DHK). *We define the* DHK*-advantage of $\mathcal{A}$ relative to $\mathcal{A}^\star$ as:*

$$\mathsf{Adv}^{\mathsf{dhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{dhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda) = 1]$$

We say that $\mathcal{G}_q$ satisfies the DHK *assumption if for every polynomial-time* dhk*-adversary $\mathcal{A}$, there exists a polynomial-time* dhk*-extractor $\mathcal{A}^\star$ such that $\mathsf{Adv}^{\mathsf{dhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$ is a negligible function in λ.*

We define $\mathsf{Adv}^{\mathsf{dhk}}_{\mathcal{G}}(\lambda) = max_{\mathcal{A}}(min_{\mathcal{A}^\star}(\mathsf{Adv}^{\mathsf{dhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)))$. The DHK assumption can be expressed as: $\mathsf{Adv}^{\mathsf{dhk}}_{\mathcal{G}}(\lambda)$ is a negligible function in λ.

4.3 Extended DHK Assumptions

We now consider an extension of the DHK assumption that we refer to as EDHK (EDHK stands for Extended Diffie-Hellman Knowledge). The adversary is now given not only (g, A) but also a DH-pair (B, C) relative to A. One way for an adversary $\mathcal{A}$ taking input (g, A), a pair (B, C) relative to A, to output a DH-pair (B', C') relative to A is to pick (and thus "know") some $x, y \in \mathbb{Z}_q$, and set $B' = B^x g^y$ and $C' = C^x A^y$.

Bellare and Palacio [1] introduced an assumption saying that this is the "only" way that a polynomial-time adversary $\mathcal{A}$, given g, A and a pair (B, C) relative to A, can output a DH-pair B', C' relative to A. Their formalization in [1] (called KEA3 assumption) is for non-randomized adversaries. For our purpose of dealing with probabilistic adversaries, we reuse the terms in [2] to formalize this assumption.

Considering an adversary $\mathcal{A}$, given (g, A) and a pair (B, C) relative to A, interacts with an extractor $\mathcal{A}^\star$, queries it adaptively, where $\mathcal{A}^\star$ is an algorithm that takes a tuple of group elements (g, A, B, C) and some state information (the state of $\mathcal{A}^\star$ is denoted by $St[\mathcal{A}^\star]$), and finally returns two exponents and a new state.

Experiment $\mathsf{Exp}^{\mathsf{edhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$
Below, $\mathcal{A}, \mathcal{A}^\star$ are probabilistic algorithms whose running times are bounded by a polynomial function in λ.

– $(g, q) \leftarrow \mathcal{G}(1^\lambda)$; $a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q, A = g^a, B = g^b, C = g^{ab}$.

– Choose coins $R[\mathcal{A}], R[\mathcal{A}^\star]$ for $\mathcal{A}; \mathcal{A}^\star$, respectively ; $St[\mathcal{A}^\star] \leftarrow ((g, A, B, C), R[\mathcal{A}])$.
– Run $\mathcal{A}$ on input g, A and coins $R[\mathcal{A}]$ until it halts, replying to its oracle queries as follows:
  If $\mathcal{A}$ makes query (B', C') then:
  • $(x||y, St[\mathcal{A}^\star]) \leftarrow \mathcal{A}^\star((B', C'), St[\mathcal{A}^\star]; R[\mathcal{A}^\star])$
  • If $(C = B^a$ and $C' = B'^a)$: if $(B' = B^x g^y$ and $C' = C^x A^y))$ then return $x||y$,
    else return 1, as the reply, to $\mathcal{A}$.
– Return 0

Assumption 6 (EDHK). *We define the* EDHK*-advantage of $\mathcal{A}$ relative to $\mathcal{A}^\star$ as:*

$$\mathsf{Adv}^{\mathsf{edhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{edhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda) = 1]$$

We say that $\mathcal{G}_q$ satisfies the EDHK *assumption if for every polynomial-time* edhk*-adversary $\mathcal{A}$, there exists a polynomial-time* edhk*-extractor $\mathcal{A}^\star$ such that $\mathsf{Adv}^{\mathsf{edhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$ is a negligible function in λ.*

We define $\mathsf{Adv}^{\mathsf{edhk}}_{\mathcal{G}}(\lambda) = max_{\mathcal{A}}(min_{\mathcal{A}^\star}(\mathsf{Adv}^{\mathsf{edhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)))$.The EDHK assumption can be expressed as: $\mathsf{Adv}^{\mathsf{edhk}}_{\mathcal{G}}(\lambda)$ is a negligible function in λ.

5 Security of the Hybrid Damgård's ElGamal Encryption

Theorem 7. *The Hybrid Damgård's ElGamal encryption is* CCA *secure assuming that:*

1. *the* HDDH *and* EHDDH *assumptions hold, and*
2. *the* DHK *and* EDHK *assumptions hold, and*
3. *the DEM is* CCA *secure.*

Proof. We use the Game hopping technique.

GAME $\mathbf{G}_0$: The simulator runs the real IND-CCA attack game. The key generation algorithm in Damgård's ElGamal encryption scheme generates a secret key $\mathsf{sk} = (x, y)$ and a corresponding public key $\mathsf{pk} = (H, g, c = g^x, d = g^y)$. The simulator is given both the secret key $\mathsf{sk} = (x, y)$ and the public key pk and it generates random coins $R[\mathcal{A}]$ and $R[\mathcal{A}^\star]$. The simulator runs the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ on input pk and coins $R[\mathcal{A}]$. When $\mathcal{A}_1$ outputs a pair of messages (m_0, m_1), the simulator produces a challenge ciphertext by flipping a coin b and producing a ciphertext of m_b. This ciphertext $(u_1^\star, u_2^\star, e^\star)$ comes from a random string $r^\star \stackrel{R}{\leftarrow} \mathbb{Z}_q$:

$$u_1^\star = g^{r^\star}, u_2^\star = c^{r^\star}, K^\star = H(d^{r^\star}), e^\star = E_{K^\star}(m_b)$$

On input $(u_1^\star, u_2^\star, e^\star)$, $\mathcal{A}_2$ outputs bit b'. We note that the adversary can submit decryption queries in both stages, before and after receiving the challenge ciphertext.

We denote by S_0 the event $b' = b$ and use the same notation S_n in any game $\mathbf{G}_n$ below. Note that the adversary is given access to the decryption oracle $\mathcal{D}_{\mathsf{sk}}$ during both steps of the attack.

$$\Pr[\mathsf{S}_0] = \frac{1}{2} \times \left(\mathsf{Adv}_{\pi}^{\mathsf{ind\text{-}cca}}(\mathcal{A}) + 1\right).$$

GAME $\mathbf{G}_1$: In this game, we simulate the decryption oracle without making use of the secret key. The input of the simulator is just the public key. In order to simulate the decryption queries, the simulator will use the DHK Extractor in the first stage (before receiving the challenge) and the EDHK Extractor in the second stage (after receiving the challenge $(u_1^\star, u_2^\star, e^\star)$). The state of the extractor $\mathcal{A}^\star$ in the first stage is set to be $St[\mathcal{A}^\star] \leftarrow ((g, c), R[\mathcal{A}])$ and the state of the extractor $\mathcal{A}^\star$ in the second stage will be set to be $St[\mathcal{A}^\star] \leftarrow ((g, c, u_1^\star, u_2^\star), R[\mathcal{A}])$. We now describe the simulation of the decryption oracle:

- Decryption queries in the first stage (the queries submitted by $\mathcal{A}_1$): whenever the adversary submits a query (u_1, u_2, e), the simulator runs the DHK Extractor $\mathcal{A}^\star((u_1, u_2), St[\mathcal{A}^\star], R[\mathcal{A}^\star])$. If the $\mathsf{Exp}^{\mathsf{dhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$ outputs 0 or 1, the simulator rejects this ciphertext. If the $\mathsf{Exp}^{\mathsf{dhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$ outputs r, the simulator computes $K = H(d^r)$ and outputs $m = D_K(m)$.
- Decryption queries in the second stage: whenever the adversary submits a query (u_1, u_2, e), the simulator uses EDHK Extractor $\mathcal{A}^\star((u_1, u_2), St[\mathcal{A}^\star], R[\mathcal{A}^\star])$. If the $\mathsf{Exp}^{\mathsf{edhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$ outputs 0, the simulator rejects this ciphertext. If the $\mathsf{Exp}^{\mathsf{edhk}}_{\mathcal{G},\mathcal{A},\mathcal{A}^\star}(\lambda)$ outputs $(r_1||r_2)$:
  - $r_2 = 0$, the simulator simply computes $K = H(d^{r_1})$ and outputs $m = D_K(e^\star)$
  - if $r_2 \neq 0$: $u_1 = g^{r_1}(u_1^\star)^{r_2}$ and $u_2 = c^{r_1}(u_2^\star)^{r_2} = DH(c, u_1)$. The session key $K = H(DH(d, u_1)) = H(d^{r_1}(u_2^\star)^{r_2})$. The simulator decrypts e under a hashed random session key (hash of a random group element of $\mathcal{G}_q$) and returns the resulted plaintext to the adversary. Under EHDDH, it's easy to see that the adversary cannot distinguish $H(d^{r_1}(u_2^\star)^{r_2})$ from $H(R)$, where $R \stackrel{R}{\leftarrow} \mathcal{G}_q$.

Therefore, under the DHK, EDHK and EHDDH assumptions, the adversary cannot distinguish the two games $\mathbf{G}_1$ and $\mathbf{G}_0$:

$$|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_0]| \leq \mathsf{Adv}^{\mathsf{dhk}}_{\mathcal{G}}(\lambda) + \mathsf{Adv}^{\mathsf{edhk}}_{\mathcal{G}}(\lambda) + n_q \mathsf{Adv}^{\mathsf{ehddh}}_{\mathcal{G}}(\lambda),$$

where n_q is the maximum number of decryption queries that the adversary could make.

GAME $\mathbf{G}_2$: One now replaces the challenge $(u_1^\star = g^{r^\star}, u_2^\star = g^{r^\star x}, K^\star = H(g^{yr^\star}))$ by a random challenge $(u_1^\star, u_2^\star = g^{r^\star x}, K^\star = H(V^\star))$, where $V^\star \stackrel{R}{\leftarrow} \mathcal{G}$. The simulation of the decryption oracle is kept unchanged. Therefore, under the MHDDH assumption, *i.e.* it is hard to distinguish $(g, u_1^\star = g^{r^\star}, d = g^y, c = g^x, u_2^\star = g^{r^\star x}, K^\star = H(g^{yr^\star}))$ from a random challenge $(g, u_1^\star, d =$

$g^y, c = g^x, u_2^\star = g^{r^\star x}, K^\star = H(V^\star)$), the two games $\mathbf{G}_2$ and $\mathbf{G}_1$ are indistinguishable. Because of the equivalence between MHDDH and HDDH assumptions(Proposition 3), we have:

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \mathsf{Adv}_{\mathcal{G}}^{\mathsf{hddh}}(\lambda),$$

GAME $\mathbf{G}_3$: In this Game, we replace $K^\star$ by a random key in $\mathcal{K}_D$. In the previous game, the key $K^\star$ is chosen as $K^\star = H(V^\star)$ for a random $V^\star$, the distance between two games $\mathbf{G}_3$ and $\mathbf{G}_2$ is thus:

$$|\Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_2]| \leq \mathsf{Adv}_{\mathcal{H}}^{\mathsf{ind}}(\lambda)$$

GAME $\mathbf{G}_4$: Finally, we replace m_b in $e^\star = E_{K^\star}(m_b)$ by a random message $m^\star$. As the key $K^\star$ is randomly chosen, the distance between the two games $\mathbf{G}_4$ and $\mathbf{G}_3$ is:

$$|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3]| \leq \mathsf{Adv}_{\mathsf{dem}}^{\mathsf{ind-cca}}(\lambda)$$

In this game, the output of $\mathcal{A}_2$ follows a distribution that does not depend on b. Accordingly, $\Pr[\mathsf{S}_4] = 1/2$. □

6 Hardness of the EDHK Problem in the Generic Group Model

The hardness of the DHK problem has been proven by Dent [6] in generic groups. In this section, we prove that the EDHK assumption also holds in generic groups. In the generic group model [19], elements of $\mathcal{G}_q$ appear to be encoded as unique random strings, so that no property other than equality can be directly tested by the adversary. An oracle is assumed to perform operations between group elements, *i.e.* the group action in the group $\mathcal{G}_q$. The encoding of the elements of $\mathcal{G}_q$ is modeled as an injective function $\xi : \mathbb{Z}_p \to \Sigma$, where $\Sigma \subset \{0,1\}^*$, which maps all $a \in \mathbb{Z}_p$ to the string representation $\xi(g^a)$ of $g^a \in G$.

Let us first recall a lemma [18,19] that proofs in generic groups often rely on.

Lemma 8 ([18,19]). *Let $F(x_1, x_2, ..., x_m)$ be a polynomial of total degree $d \geq 1$. Then the probability that $F(x_1, x_2, ..., x_m) = 0$ mod n for randomly chosen values $(x_1, x_2, ..., x_m)$ in $\mathbb{Z}_n$ is bounded above by d/p where p is the largest prime dividing n.*

We now show that the EDHK assumption holds in a generic group.

Theorem 9. *The* EDHK *assumption holds in a generic group.*

Proof. The extractor $\mathcal{A}^\star$ keeps track of the oracle queries of $\mathcal{A}$ as polynomials. $\mathcal{A}^\star$ maintains a list of pair $L = \{(F_i, \xi_i) : i = 0, 1, ..., \tau - 1)\}$, where F_i are polynomials of degree ≤ 2 in $\mathbb{Z}_q[x, y, z_1, .., z_m]$. We set $F_0 = 1, F_1 = x, F_2 = y, F_3 = xy$ and $\tau = 4, m = 0$. The corresponding $\xi_0, \xi_1, \xi_2, \xi_3$ are set to be arbitrary distinct strings in $\{0,1\}^*$. $\mathcal{A}^\star$ starts the game by providing $\mathcal{A}$ with the strings $\xi_0, \xi_1, \xi_2, \xi_3$. Each query of $\mathcal{A}$ is a group action.

$\mathcal{A}^\star$ answers group action queries as follows:

When $\mathcal{A}$ makes queries on strings that have not been in the list, those strings will be added to the list and assigned new variables: each time a new string appears, we denote it by ξ_τ and assign a new variable z_{m+1} to the element ξ_τ, we add $(F_\tau = z_m, \xi_\tau)$ to the list L and then increment τ and m by one.

We can now assume that $\mathcal{A}$ makes queries on strings in the list. Given a multiply/divide selection bit and two operands ξ_i, ξ_j with $0 \leq i, j \leq \tau - 1$, we compute $F_\tau = F_i + F_j \in \mathbb{Z}_q[x, y, z_1, .., z_m]$ or $F_\tau = F_i - F_j \in \mathbb{Z}_q[x, y, z_1, .., z_m]$ depending on whether a multiplication or a division is requested. If $F_\tau = F_\ell$ for some $\ell < \tau$, we set $\xi_\tau = \xi_\ell$; otherwise, we set ξ_τ to a string in $\{0,1\}^*$ distinct from $\xi_0, \xi_1, ..., \xi_\tau$. We add (F_τ, ξ_τ) to L and give ξ_τ to $\mathcal{A}$, then increment τ by one.

$\mathcal{A}$ terminates and returns a pair ξ_i, ξ_j where $0 \leq i, j < \tau$. Let F_i, F_j be the corresponding polynomial in the list L. Note that if $\mathcal{A}$'s answer is correct then necessarily:

$$F_i(x, y, xy, z_1, z_2, ..., z_m) = xF_j(x, y, xy, z_1, z_2, ..., z_m)$$

Denote $F^\star(x, y, z_1, z_2, ..., z_\tau) = F_i(x, y, z_1, z_2, ..., z_\tau) - xF_j(x, y, xy, z_1, z_2, ..., z_\tau)$.

Case 1: $F^\star$ is identical to 0. From the above simulation, F_i, F_j should have the following form:

- $F_i(x, y, xy, z_1, z_2, ..., z_m) = c_{i_0} + \alpha_i x + \beta_i y + \gamma_i xy + c_{i_1} z_1 + ... + c_{i_m} z_m$
- $F_i(x, y, xy, z_1, z_2, ..., z_m) = c_{j_0} + \alpha_j x + \beta_j y + \gamma_j xy + c_{j_1} z_1 + ... + c_{j_m} z_m$

Therefore: $F^\star(x, y, z_1, z_2, ..., z_\tau) = c_{i_0} + (\alpha_i - c_{j_0})x + (\gamma_i - \beta_j)xy + \beta_i y - \alpha_j x^2 - \gamma_j xz + c_{i_1} z_1 + ... + c_{i_m} z_m - c_{j_1} xz_1 - ... - c_{j_m} xz_m - \gamma_j x^2 y$. We deduce thus:

- $\beta_i = \gamma_j = c_{i_1} = ... = c_{i_m} = c_{j_1} = ... = c_{j_m} = 0$
- $\alpha_i = c_{j_0} (= r_1)$
- $\gamma_i = \beta_j (= r_2)$

$F_i = r_1 x + r_2 xy, F_j = r_1 + r_2 y$. As F_i, F_j have this form, $\mathcal{A}^\star$ could easily extract and outputs $r_1 || r_2$.

Case 2: $F^\star$ is not identical to 0. In this case, $F^\star(x, y, z_1, z_2, ..., z_\tau) = 0$ is a non-trivial equation. At this point, $\mathcal{A}^\star$ chooses randomly $x^\star, y^\star, z_1^\star, z_2^\star, ..., z_m^\star$. The simulation provided by B is perfect unless the instantiation $x \leftarrow x^\star, y \leftarrow y^\star, z_1 \leftarrow z_1^\star, ..., z_m \leftarrow z_m^\star$ creates an equality relation between the simulated group elements that was not revealed to $\mathcal{A}$. The success probability of $\mathcal{A}$ is thus bounded by the probability that any of the following holds:

- $F_i(x^\star, y^\star, z_1^\star, ..., z_m^\star) = F_j(x^\star, y^\star, z_1^\star, ..., z_m^\star)$, for some $0 \leq i, j < \tau$. From Lemma 8, this occurs with a probability bounded by $\tau^2 2/q$.
- $F^\star(x^\star, y^\star, z_1^\star, ..., z_m^\star) = 0$. From Lemma 8, this occurs with a probability bounded by $1/q$.

□

7 Conclusion

We propose a hybrid Damgård's ElGamal encryption that is CCA secure. The proposed scheme is very efficient but its security should be based on an extension of the DHK assumption, which is quite strong. There are however some reasons that could convince us about the usefulness of this scheme. Firstly, we proved that the Extended DHK assumption holds in generic groups. Secondly, Gjøsteen[10] proposed a new technique of security proof for Damgård's ElGamal scheme (against non-adaptive chosen ciphertext) in which the DHK assumption can be replaced by an assumption of hardness of a new problem, the gap subgroup membership problem, which is somewhat similar to conventional problems such as the Gap Diffie-Hellman problem. Therefore, it could be possible that the Gjøsteen's technique (or another new technique) would help to replace the EDHK assumption by a more conventional one[1]. Finally, note that the HDDH assumption is generally weaker than DDH assumption and might hold even in groups where DDH problem is easy [8] and that EHDDH, EDHK assumption, though strong, might also hold in groups where the DDH problem is easy (it seems easy to prove the hardness of these assumptions in generic groups with pairings, for example). Therefore, it might happen that our proposed scheme is still secure in some non-DDH groups.

Acknowledgments

We would like to thank Eike Klitz for helpful discussions.

References

1. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
2. Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
3. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
4. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
5. Damgård, I.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
6. Dent, A.W.: The Hardness of the DHK Problem in the Generic Group Model. Cryptology ePrint Archive, Report 2006/156 (2006), http://eprint.iacr.org/

[1] An independent work recently posted on ePrint [12] shows indeed that hybrid Damgård's ElGamal encryption can be proved under standard assumptions.

7. ElGamal, T.: A Public-key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Trans. on Information Theory IT-31(4), 469–472 (1985)
8. Gennaro, R., Krawczyk, H., Rabin, T.: Secure Hashed Diffie-Hellman over Non-DDH Groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027. Springer, Heidelberg (2004)
9. Gennaro, R., Shoup, V.: A note on an encryption scheme of Kurosawa and Desmedt. Cryptology ePrint Archive, Report 2004/194 (2004), http://eprint.iacr.org/
10. Gjøsteen, K.: A new security proof for damgård's ElGamal. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 150–158. Springer, Heidelberg (2006)
11. Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and System Sciences 28, 270–299 (1984)
12. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: Randomness extraction: A new paradigm for hybrid encryption. Cryptology ePrint Archive, Report 2008/304 (2008), http://eprint.iacr.org/
13. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
14. Lipmaa, H.: On CCA1-Security of Elgamal And Damgård Cryptosystems. Cryptology ePrint Archive, Report 2008/234 (2008), http://eprint.iacr.org/
15. Naor, M., Yung, M.: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attack. In: 22nd Annual ACM Symposium on Theory of Computing, pp. 427–437. ACM Press, New York (1990)
16. Rackoff, C., Simon, D.R.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
17. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
18. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM 27(4), 701–717 (1980)
19. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
20. Shoup, V.: Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)
21. Wu, J., Stinson, D.: On The Security of The ElGamal Encryption Scheme and Damgards Variant. Cryptology ePrint Archive, Report 2008/200 (2008), http://eprint.iacr.org/

Construction of Yet Another Forward Secure Signature Scheme Using Bilinear Maps

Jia Yu[1], Fanyu Kong[2], Xiangguo Cheng[1], Rong Hao[1], and Guowen Li[3]

[1] College of Information Engineering,
Qingdao University, Qingdao 266071, China
{yujia,chengxg,hr}@qdu.edu.cn
[2] Institute of Network Security,
Shandong University, Jinan 250100, China
fanyukong@sdu.edu.cn
[3] School of Computer Science and Technology,
Shandong Jianzhu University, Jinan 250101, China
guowenli@gmail.com

Abstract. Forward secure signatures are proposed to deal with the key exposure problem. Compared to regular signatures, forward secure signatures can protect the security of signatures previous to the time period of key exposure. The efficiency is an important issue of forward secure signatures. In this paper, we construct yet another forward secure signature scheme using bilinear maps. In this scheme, all performance parameters have complexities of log magnitude in terms of the total time periods. In addition, our scheme needs very few pairing operations in verifying algorithm, which is very important because the pairing operation is very time-consuming. At last, we prove that our scheme is forward secure in random oracle model assuming CDH problem is hard.

Keywords: forward security, computation Diffie-Hellman problem, bilinear maps, digital signature.

1 Introduction

The secret key exposure has arisen much attention of cryptography researchers because it threatens the security of digital signatures greatly. Therefore, how to deduce the damage of key exposure for digital signatures is an important work. Forward security for digital signatures is one kind of methods widely used. In forward secure signatures, the whole time is divided into several time periods. Each secret key is not only used to sign the message in the current time period but also used to compute the secret key of the next time period by a one-way function. So the adversary is computationally infeasible to forge any signature previous the period of key exposure even if the current key is exposed.

Anderson [1] firstly proposed to apply forward security to digital signatures. Bellare and Miner [2] further formalized forward secure signatures and proposed a practical scheme. They also provided the definitions of forward secure signatures and its security. Subsequently a lot of constructions of forward secure

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 83–97, 2008.

signature schemes [3,4,5,6] were proposed. The scheme [5] had optimal signing and verifying algorithms at the expense of slower key update. While the scheme [6] could achieve fast key update with another method but had slower signing and verifying algorithms. In these schemes, at least one of operations of key generation, key update, signing and verifying algorithms were linear to the total number of time periods T. Malkin *et al.* [7] proposed a forward secure signature scheme in which the operations of all sub-algorithms were independent of T but linear to the current time period, where T was the total number of time periods. With the time period increasing, the efficiency of Malkin's scheme was still greatly influenced by T. How to construct a forward secure signature scheme with higher efficiency has been a hot topic in research for a long time, which is an open problem presented by Itkis [8]. Hierarchical ID-based cryptography advocated by Gentry and Silverberg [9] could be used to construct efficient forward secure signature schemes. Based on hierarchical ID-based cryptography [9] and ke-PKE [10], the first forward secure signature scheme using bilinear maps was proposed in [11], whose efficiency was balanced across all its aspects because each parameter in the scheme had a complexity no larger than $O(\log T)$. Because the pairing operation is time-consuming, reducing the times of pairing operations in forward secure signature from bilinear maps is very important to improve efficiency. Another forward secure signature scheme with similar construction was proposed in [12]. Vo and Kim [13] proposed yet another forward secure signature from bilinear pairings in 2005. They claimed that the operations of all sub-algorithms in their scheme didn't increase with T increasing. Unfortunately, it was proved that the scheme didn't satisfy the forward security [14]. A fine-grained forward-secure signature scheme was proposed in [15], which allowed the signer to specify which signatures of the current time period remained valid when revoking the public key. Boyen *et al.* presented a forward secure signature with untrusted update in [16]. Libert *et al.* [17] gave generic constructions of forward secure signatures in untrusted update environments.

Forward secure symmetric-key encryption and forward secure public key encryption were studied in [18] and [10]. Forward secure threshold signature was also researched in [19,20,21,22]. Key-insulation [23,24,25,26] and intrusion-resilient cryptography [27,28,29,30] were proposed to achieve a high level of security. However, these methods needed synchronization and the signer's communication with a safe device for each time period. So they were not able to apply to many scenarios.

Our contribution. We construct yet another forward secure signature scheme using bilinear maps in this paper. The scheme has nice average performance. We makes use of the binary tree structure similar to [11], as a result, the scheme has an advantage that all the complexities of the running times of key generation, key update, signing and verifying algorithms and the sizes of public key, secret key, and signature are no more than $O(\log T)$ in terms of the total number of time periods T. Because the scheme is operating over a certain elliptic curve, the signature size and the secret key size are relatively short. In addition, there are only triple pairing operations in its verifying algorithm because a new approach

is performed to derive the local secret key. It is very important because the pairing operation influences the efficiency of verifying algorithm greatly. The security of our scheme is based on CDH assumption. It is proved to be forward secure in random oracle model assume CDH problem is hard.

Organization. In Section 2, we introduce the preliminaries of our work, including cryptographic assumptions, forward secure signature scheme and its security. A concrete description of our scheme is given in Section 3. In addition, the performance analysis and security analysis are given in Section 4 and 5, respectively. Finally, we conclude the paper in Section 6.

2 Preliminaries

2.1 Cryptographic Assumptions

We give some cryptographic assumptions that are similar to the descriptions in [9]. Let G_1 and G_2 be two cyclic groups of some prime order q, where G_1 is represented additively and G_2 is represented multiplicatively. And let $P \in G_1$ be a generator of G_1. A bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ satisfies:

1. Bilinear: For all $P, Q \in G_1$ and $a, b \in Z$, there is $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degenerate: The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2.
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

Computation Diffie-Hellman (CDH) problem: Given (P, aP, bP), where $a, b \in Z_q$, compute abP. The advantage of an algorithm A in solving the CDH problem in a group G_1 is

$$AdvCDH_A = Pr[A(P, aP, bP) = abP | a, b \xleftarrow{R} Z_q]$$

Definition 1 (*CDH Assumption*). *A probabilistic algorithm A is said (t, ε)-break CDH problem in G_1 if A runs at most time t, computes CDH problem with an advantage of at least ε . We say that G_1 is a (t, ε)-secure CDH group if no probabilistic algorithm A (t, ε)-break CDH problem in G_1.*

IG is a CDH parameter generator if only it takes security parameter k, outputs the description of two groups G_1 and G_2 of the same prime order q and an admissible pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$. We say that *IG* satisfies the CDH assumption if for all PPT algorithms A the following probability is negligible:

$$Pr[A(G_1, G_2, \hat{e}, P, aP, bP) = abP | (G_1, G_2, \hat{e}) \leftarrow IG(1^k), P \leftarrow G_1^*, a, b \leftarrow Z_q]$$

2.2 Forward Secure Signature Scheme

A forward secure signature scheme is a key-evolving signature scheme with special security. Different from standard signatures, key-evolving signatures have an additional algorithm—key update algorithm which takes the current secret

key as input and derives a new secret key for the next period. So a key-evolving signature scheme consists of a key generation algorithm, a key update algorithm, a signing algorithm and a verifying algorithm. We firstly review the definition of the key-evolving signature scheme.

Definition 2 (*Key-evolving Signature Scheme*). *A key-evolving signature scheme is an quadruple of algorithms* FSIG=(FSIG.key, FSIG.update, FSIG.sign, *FSIG.verify), where:*

1. FSIG.key: *the key generation algorithm, is a probabilistic algorithm which takes as input a security parameter $k \in N$ and the total number of time periods T, and generates a public key PK and the initial secret key SK_0.*
2. FSIG.update: *the key update algorithm, is a probabilistic algorithm which takes as input the secret key SK_j of the current period j, and generates the new secret key SK_{j+1} for the next period.*
3. FSIG.sign: *the signing algorithm, takes as input the secret key SK_j of the current time period j and a message M, and generates a signature $< j, sign >$ of M for period j. This algorithm may be probabilistic.*
4. FSIG.verify: *the verifying algorithm, is a deterministic algorithm which takes as input the public key PK, a message M and a candidate signature $< j, sign >$and output 1 when $< j, sign >$ is a valid signature or 0, otherwise.*

If $< j, \sigma >$ is a signature of message M generated by FSIG.sign algorithm, then $\texttt{FSIG.verify}_{\texttt{PK}}(M, < j, \sigma >) = 1$. Assume SK_j always contains both values j and T.

2.3 Security Definition

If a key-evolving signature scheme is a forward secure signature scheme, then an adversary is computationally difficult to forge any signature of the previous period even if she gets the current secret key. Below we indicate the experiment to evaluate the security of forward secure signature in random oracle model:

From below experiment, the adversary knows the total number of time periods and the current time period. Adversary F runs in three phases: The first is the chosen message attack phase (cma). In this phase, F has access to a signature oracle to query any signature of the message she selects with respect to the current secret key. At the end of each time period, F can choose to stay in this phase or go to the next phase. The second phase is the break-in phase. In this phase, the adversary is given the secret key SK_{i^*} for the specific time period i^* she decides to break in. The last phase is the forgery phase (forge). In this phase, the adversary needs to output a forgery. The adversary is considered to be successful if she forges a signature of some new message (that is, not queried previously) for some time period prior to i^*. The hash function H is viewed as a random oracle and F can query random oracle H in the whole procedure.

Definition 3 (***Forward-security in Random Oracle Model***). *Let* KE-SIG *is a key-evolving signature scheme with security parameter k, number of time period T. Let H be a random oracle and F be an adversary above described. Let*

Experiment. F-Forge-RO(FSIG, F)

Select $H : \{0,1\}^* \rightarrow \{0,1\}^l$ at random
$(PK, SK_0) \xleftarrow{R} \texttt{FSIG.key}^H(k, \cdots, T)$
$j = 0$
Repeat
    $d \leftarrow F^{H,\texttt{FSIG.sign}^H_{SK_j}(\cdot)}(cma, PK)$
    $SK_{j+1} \leftarrow \texttt{FSIG.update}^H(SK_j)$
    $j \leftarrow j + 1$
Until (d=break) or ($j = T$)
$i^* \leftarrow j$
$(M, < j, sign >) \leftarrow F^H(\texttt{forge}, SK_{i^*})$
If $\texttt{FSIG.verify}^H_{PK}(M, < i, sign >) = 1$ and $0 \leq i < i^*$ and M was not queried of $\texttt{FSIG.sign}^H_{SK_i}(\cdot)$ in period i
then
    return 1
else
    return 0

$Succ^{fsig}(\texttt{KE-SIG}[k,T], F)$ denote the probability that above experiment returns 1. Then the insecurity of KE-SIG *scheme is function*

$$Insec^{fsig}(\texttt{KE-SIG}[k,T], t, q_{sig}, q_{hash}) = max_F\{Succ^{fsig}(\texttt{KE-SIG}[k,T], F)\}$$

Where the maximum is taken over all adversaries F satisfies the following conditions: the time for executing the above experiment is at most t; F makes at most q_{sig} signing queries to the signing oracle and q_{hash} hash queries to the H-oracle.

Above definitions and experiment are taken from [2,3,11] with slight modifications.

3 The Proposed Forward Secure Signature Scheme

3.1 Notations and Constructions

Our scheme makes use of a binary tree structure, which has been used in many cryptographic schemes. This structure is firstly suggested to form forward secure signature schemes by Bellare and Miner [2]. It also has been adopted in [10,11,12,28]. As in [11], for convenience, we define the total number of time periods T to be a power of 2 ($T = 2^l$). We use a full binary tree with depth l and present each node with a binary string ζ (the root node is an empty string $\zeta = \varepsilon$). If the depth of node ζ is less than l, then its left and right children are represented with $\zeta 0$ and $\zeta 1$, respectively. Each time period $i (0 \leq i \leq T-1)$ can be represented using a binary string in l bits $< i >= i_1 i_2 \cdots i_l$, that is, each leaf $< i >$ from leftmost leaf to the rightmost leaf can represent each period i.

The leftmost leaf represents period 0 and the rightmost of leaf represents period $T-1$. The binary presentation of internal node is $\zeta = \zeta_1\zeta_2\cdots\zeta_j (1 \le j < l)$, where j is the depth of nod ζ.

The root nod ϵ contains root secret s_ϵ and root verification point Q. Each internal nod ζ contains a node secret s_ζ, a local secret $S_\zeta \in G_1$, and a verification point Q_ζ. The local secret of each leaf nod $< i >$ is $sk_{<i>} = (S_{<i>}, \Phi_{<i>})$, where $\Phi_{<i>} = (Q_{i_1}, \cdots, Q_{i_1 i_2 \cdots i_{l-1}}, Q_{<i>})$. All the verification points are not real secrets, but will be published as a part of signature.

Denote the leaf nod with $< i >= i_0 i_1 i_2 \cdots i_l$ (where $i_0 = \epsilon$). The full secret SK_i in period consists of $(1)s_{<i>}$ $(2)sk_{<i>}$ $(3)\{sk_{i_0 i_1 \cdots i_{j-1}1}\}$ (for each $i_j = 0, 1 \le j \le l$). The first two parts are leaf secret and local secret. For each $i_j = 0, 1 \le j \le l$, internal node $\zeta 0 = i_0 i_1 \cdots i_j$ has a right sibling node $\zeta 1 = i_0 i_1 \cdots i_{j-1} 1$, and we use $\Psi(i)$ denote the set of all right sibling nodes of nod i. Represent SK_i as $\{s_{<i>}, S_{<i>}, Set_{<i>}, \Phi_{<i>}\}$, where $Set_{<i>} = \{S_\zeta | \zeta \in \Psi(i)\}$. $Set_{<i>}$ can evolve the full secrets $SK_{i'}$ for all periods $i' > i$, but can't evolve the full secrets for previous periods. The signature can be generated from secret $s_{<i>}$,$S_{<i>}$ and $\Phi_{<i>}$. Let the signer secret be $\{s_{<i>}, S_{<i>}, Set_{<i>}, \Phi_{<i>}\}$, where $Set_{<i>} = \{S_\zeta | \zeta \in \Psi(i)\}$. The secrets $s_{<i>}$,$S_{<i>}$ and $\Phi_{<i>}$ in signer secrets are used to generate signatures, and the secret $Set_{<i>}$ is used to evolve the secret of the next period.

3.2 Description of the Scheme

Below we give the detailed description of our forward secure signature scheme using bilinear maps.

(1) `algorithm FSIG.key`(k, l, T)
Begin
  Run IG(1^k) to generate additive group G_1 and multiplicative group G_2
  with same prime order q and an admissible pairing $\hat{e} : G_1 \times G_1 \to G_2$.
  Select cryptographic hash functions $H_1 : \{0,1\}^* \times G_1 \to Z_q^*$,
  $H_2 : G_1 \to G_1$, $H_3 : \{0,1\}^* \times G_1 \to G_1$.
  Select generator $P \in_R G_1$ and secret $s_\epsilon \in_R Z_q^*$, and let
  $Q = s_\epsilon P$, $S_\varepsilon = s_\epsilon H_2(Q)$.
  Let the public key be $PK = \{G_1, G_2, \hat{e}, H_1, H_2, H_3, P, Q\}$.
  Select $s_0, s_1 \in_R Z_q^*$, and compute $Q_0 = s_0 P$, $Q_1 = s_1 P$.
  Compute $S_0 = S_\epsilon + s_0 H_1(0, Q_0) H_2(Q)$ and $S_1 = S_\epsilon + s_1 H_1(1, Q_1) H_2(Q)$.
  For $j = 1$ to $l - 1$
    Select $s_{0^j 0}, s_{0^j 1} \in_R Z_q^*$, and compute $Q_{0^j 0} = s_{0^j 0} P$, $Q_{0^j 1} = s_{0^j 1} P$.
    Compute $S_{0^j 0} = S_{0^j} + s_{0^j 0} H_1(0^j 0, Q_{0^j 0}) H_2(Q)$,
    and $S_{0^j 1} = S_{0^j} + s_{0^j 1} H_1(0^j 1, Q_{0^j 1}) H_2(Q)$.
  Set $SK_0 = \{S_{0^l}, Set_{0^l}, \Phi_{0^l}\}$, where $\Phi_{0^l} = (Q_0, \cdots, Q_{0^{l-1}}, Q_{0^l})$, and
  $Set_{0^l} = (< S_1, Q_1 >, < S_{01}, Q_{01} >, \cdots, < S_{0^{l-1}1}, Q_{0^{l-1}1} >)$.
  Erase all interim data, and return PK, SK_0.
End

(2) `algorithm FSIG.update`(SK_i)

Begin

If $i = T - 1$ then

Let $SK_T = \phi$.

Else

Parse $< i >= i_0 i_1 i_2 \cdots i_l$, $(i_0 = \epsilon)$, $SK_i = \{S_{<i>}, Set_{<i>}, \Phi_{<i>}\}$,
and $\Phi_{<i>} = (Q_{i_1}, \cdots, Q_{i_1 i_2 \cdots i_{l-1}}, Q_{<i>})$.

If $i_l = 0$ then

Find $< S_{<i+1>}, Q_{<i+1>} >$ from $Set_{<i>}$,
and set $Set_{<i+1>} = Set_{<i>} - \{< S_{<i+1>}, Q_{<i+1>} >\}$.
Set $\Phi_{<i+1>} = (\Phi_{<i>} - \{Q_{<i>}\}) \cup \{Q_{<i+1>}\}$.
Set $SK_{i+1} = (S_{<i+1>}, Set_{<i+1>}, \Phi_{<i+1>})$.

Else

Find the maximal $j (1 \leq j < l)$ satisfying $i_j = 0$,
and let $\eta = i_0 i_1 \cdots i_{j-1} 1$, $(i_0 = \epsilon)$. Here $< i + 1 >= \eta 0^{l-j}$.
Find $< S_\eta, Q_\eta >$ from $Set_{<i>}$,
and set $Set_{<i+1>} = Set_{<i>} - \{< S_\eta, Q_\eta >\}$.
Set $\Phi_{<i+1>} = \Phi_{<i>} - \{Q_{i_1 i_2 \cdots i_j}, Q_{i_1 i_2 \cdots i_j 1}, \cdots, Q_{i_1 i_2 \cdots i_j 1^{l-j-1}}, Q_{<i>}\}$.
For $m = 1$ to $l - j$

Select $s_{\eta 0^m}, s_{\eta 0^{m-1} 1} \in_R Z_q$, and compute
$Q_{\eta 0^m} = s_{\eta 0^m} P$,
$Q_{\eta 0^{m-1} 1} = s_{\eta 0^{m-1} 1} P$,
$S_{\eta 0^m} = S_{\eta 0^{m-1}} + s_{\eta 0^m} H_1(\eta 0^m, Q_{\eta 0^m}) H_2(Q)$,
and $S_{\eta 0^{m-1} 1} = S_{\eta 0^{m-1}} + s_{\eta 0^{m-1} 1} H_1(\eta 0^{m-1} 1, Q_{\eta 0^{m-1} 1}) H_2(Q)$,
Set $Set_{<i+1>} = Set_{<i+1>} \cup \{< S_{\eta 0^{m-1} 1}, Q_{\eta 0^{m-1} 1} >\}$,
and $\Phi_{<i+1>} = \Phi_{<i+1>} \cup \{Q_{\eta 0^m}\}$.

Set $SK_{i+1} = \{S_{<i+1>}, Set_{<i+1>}, \Phi_{<i+1>}\}$.

Erase all interim data, and return SK_{i+1}.

End

(3) `algorithm FSIG.sign`(i, SK_i, M)

Begin

Parse $< i >= i_1 i_2 \cdots i_l$, $SK_i = \{S_{<i>}, Set_{<i>}, \Phi_{<i>}\}$.
Select $r \in_R Z_q^*$, and compute $U = rP$.
Compute $V = S_{<i>} + r H_3(i_1 i_2 \cdots i_l || M, U)$.
Return signature $< i, sign = (U, V, \Phi_{<i>}) >$.

End

(4) `algorithm FSIG.verify`$(M, PK, < i, sign >)$

Begin

Parse $< i >= i_1 i_2 \cdots i_l$,$sign = (U, V, \Phi_{<i>})$,
and $\Phi_{<i>} = (Q_{i_1}, \cdots, Q_{i_1 i_2 \cdots i_{l-1}}, Q_{<i>})$.
Verify

$$\hat{e}(P, V) = \hat{e}(Q + \textstyle\sum_{j=1}^{l} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) Q_{i_1 i_2 \cdots i_j}, H_2(Q)) \cdot \hat{e}(U, H_3(i_1 i_2 \cdots i_l || M, U))$$

If it holds, return "valid";
Otherwise, return "invalid".

End

4 Performance Analysis

The proposed scheme is based on the binary tree structure that has been used in [10,11,28], so it enjoys a particular advantage of this structure. The scheme has a nice average performance, that is, there is no cost parameters including key generation time, key update time, signing time, verifying time, and signature size, public key size, secret storage size has a complexity more than $O(\log T)$ in terms of the total number of time periods T in this scheme. In schemes [2,3,5,6,7] that don't use this structure, at least one in key generation, key update, signing and verifying algorithms have a complexity of $O(T)$ with the current period increasing. Our scheme is more efficient in verifying algorithm because there are only triple pairing operations compared with $O(\log T)$ pairing operations in scheme [11] with the similar binary tree structure. It is very important because the pairing operation is very time-consuming. Furthermore, the verifying algorithm can be further optimized by precomputing and storing $\hat{e}(Q + \sum_{j=1}^{l} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) Q_{i_1 i_2 \cdots i_j}, H_2(Q))$ after the verifier receives the first signature from the signer at the corresponding time period. In addition, the scheme is operating over a certain elliptic curve, thus, it possesses the traits of short signature based on bilinear pairing. When T is not very large, the signature size and the secret key size are very short. The following table gives a brief performance comparison with scheme [11] which is based on the same construction and computation assumption with our scheme. We consider the cost in terms of the number of scalar multiplications and pairing computations in the group G_1. In the following table, we name our FSIG scheme before precomputation as FSIG_A and the scheme after precomputation as FSIG_B. Use S to present the scalar multiplication in G_1 and use P to present the pairing computation.

Table 1. Performance comparisons (in terms of T)

Parameters	Scheme in [11]	FSIG_A	FSIG_B
Key generation time	$O(\log T)S$	$O(\log T)S$	$O(\log T)S$
Key update time	$O(\log T)S$	$O(\log T)S$	$O(\log T)S$
Sign time	S	$2S$	S
Verify time	$O(\log T)P$	$O(\log T)S + 3P$	$2P$
Signature size (bits)	$O(\log T)k$	$O(\log T)k$	$O(\log T)k$
Public Key size (bits)	$O(\log T)k$	$O(\log T)k$	$O(\log T)k$
Secret Key size (bits)	$O(\log T)k$	$O(\log T)k$	$O(\log T)k$

5 Security Analysis

Theorem 1. *Assume* $< i, sign = (U, V, \Phi_{<i>}) >$ *is a signature of message* M *for period* i *generated by signing algorithm. Then* $\text{FSIG.verify}(M, PK, < i, sign = (U, V, \Phi_{<i>}) >) =$ *"valid".*

Proof

$$
\begin{aligned}
&\hat{e}(P,V)\\
&=\hat{e}(P,S_{<i>}+rH_3(i_1i_2\cdots i_l||M,U))\\
&=\hat{e}(P,S_{<i>})\cdot\hat{e}(P,rH_3(i_1i_2\cdots i_l||M,U))\\
&=\hat{e}(P,s_\epsilon H_2(Q)+\sum_{j=1}^{l}s_{i_1i_2\cdots i_j}H_1(i_1i_2\cdots i_j,Q_{i_1i_2\cdots i_j})H_2(Q))\\
&\quad\cdot\hat{e}(rP,H_3(i_1i_2\cdots i_l||M,U))\\
&=\hat{e}(P,(s_\epsilon+\sum_{j=1}^{l}s_{i_1i_2\cdots i_j}H_1(i_1i_2\cdots i_j,Q_{i_1i_2\cdots i_j}))H_2(Q))\\
&\quad\cdot\hat{e}(rP,H_3(i_1i_2\cdots i_l||M,U))\\
&=\hat{e}(s_\epsilon P+\sum_{j=1}^{l}s_{i_1i_2\cdots i_j}H_1(i_1i_2\cdots i_j,Q_{i_1i_2\cdots i_j})P,H_2(Q))\\
&\quad\cdot\hat{e}(U,H_3(i_1i_2\cdots i_l||M,U))\\
&=\hat{e}(Q+\sum_{j=1}^{l}H_1(i_1i_2\cdots i_j,Q_{i_1i_2\cdots i_j})Q_{i_1i_2\cdots i_j},H_2(Q))\\
&\quad\cdot\hat{e}(U,H_3(i_1i_2\cdots i_l||M,U))
\end{aligned}
$$

Let the length of prime q be k bits. In order to simplify the efficiency analysis, we assume a group operation on G_1 is at most the time of $O(k^n)$ bit operation.

Theorem 2. *If there is a forger F which runs in time at most t, asking at most q_{sig} signing queries and q_{hash} random oracle H_3-oracle queries, such as $\mathtt{Succ}^{fsig}(\mathtt{KE\text{-}SIG}[k,T],F)>\epsilon$, then there exists an adversary A that (t',ϵ')-break CDH problem in group G_1, where*

$$
\begin{aligned}
t' &= t+O(max\ \{q_{sig}\log T,q_{hash}\}\cdot k^n+max\{\log T,q_{sig}\}\cdot k^2)\\
\epsilon' &= \frac{1}{T}(\epsilon-\frac{q_{sig}(q_{hash}-1)}{q-1})
\end{aligned}
$$

Proof. As discussed in [3,11], we assume that hash query $(i_1i_2\cdots i_l||M,U)$ must be made simultaneously when F makes signing query of message M for period i, where $<i>=i_1i_2\cdots i_l$. It may make the number of hash queries be increased to $q_{hash}+q_{sig}$. Assume F maintains all necessary records and can't query the same hash twice.

We view H_2 as an ordinary hash function and H_3 as a random oracle in the proof. Furthermore, replace H_1 by $(l+1)$-wise independent hash function in function family [31].

Firstly, the algorithm A is given parameters $(G_1,G_2,\hat{e})$ and $(P,Q=s_\epsilon P=\alpha P,P'=H_2(Q)=\beta P)$ generated by IG, and the goal of A is to compute $\alpha\beta P$, where $\alpha=s_\epsilon,\beta\in Z_q$ are unknown to A and F. A runs F as a subroutine. A selects a total time periods T and guesses the time period i^* randomly at which F asks the break-in queries. The probability that her guess is right is $1/T$. Let $<i^*>=i_1^*i_2^*\cdots i_l^*$.

A selects at random $h_{i_1^*i_2^*\cdots i_l^*},y_{i_1^*i_2^*\cdots i_l^*}\in Z_q^*$, and $h_{i_1^*i_2^*\cdots i_{j-1}^*1},y_{i_1^*i_2^*\cdots i_{j-1}^*1}$ in Z_q^* where $1\le j\le l$ and $i_j^*=0$. A randomly selects hash function H_1 from a

$(l+1)$-wise independent hash function family with the following constraints for $1 \le j \le l$ and $i_j^* = 0$.

$H_1(i_1^* i_2^* \cdots i_{j-1}^* 1, Q_{i_1^* i_2^* \cdots i_{j-1}^* 1}) = h_{i_1^* i_2^* \cdots i_{j-1}^* 1}$,

$H_1(i_1^* i_2^* \cdots i_l^*, Q_{i_1^* i_2^* \cdots i_l^*}) = h_{i_1^* i_2^* \cdots i_l^*}$, where

$Q_{i_1^* i_2^* \cdots i_{j-1}^* 1} = 1/h_{i_1^* i_2^* \cdots i_{j-1}^* 1}(y_{i_1^* i_2^* \cdots i_{j-1}^* 1} P - P')$,

$Q_{i_1^* i_2^* \cdots i_l^*} = 1/h_{i_1^* i_2^* \cdots i_l^*}(y_{i_1^* i_2^* \cdots i_l^*} P - P')$.

A provides $PK = (G_1, G_2, \hat{e}, P, Q)$ and the total number of time period T to F.

A maintains two tables: H_3 table, and signing oracle table to answer the queries from F. The element in H_3 oracle table is tuple $(< j > ||M, U, h, \lambda, \varphi)$, where $< j > ||M$ and U represent the input values of the query, h represents the output value of this query and the values λ, φ present two temporary variables.

The simulation of H_3 queries. When F queries H_3 hash oracle, A must guarantee the answer is random. If F queries H_3 oracle at a point $(< i > ||M, U)$. A does as following:

1. If $(< i > ||M, U)$ has already appeared on a tuple $(< i > ||M, U, h, \lambda, \varphi)$ in H_3 table, then A responds $H_3(i_1 i_2 \cdots i_l || M, U) = h \in G_1$ to F.

2. Else selects $\lambda \in_R Z_q$ computes $h = \lambda P$ and adds $(< i > ||M, U, h, \lambda, *)$ to H_3 table. A responds $H_3(i_1 i_2 \cdots i_l || M, U) = h \in G_1$ to F.

Signing requires. When A requires the signature of $< M, i >$, F does as follows:

1. If $i \neq i^*$, A selects $\lambda, \varphi \in_R Z_q^*$, and computes $h = \lambda P - (1/\varphi)P'$, $U = \varphi Q$. If $H_3(i_1 i_2 \cdots i_l || M, U)$ has been defined, then A aborts.

A adds $(< i > ||M, U, h, \lambda, \varphi)$ to the H_3 table. A selects $s_{i_1 i_2 \cdots i_j} \in_R Z_q^* (1 \le j \le l)$(If hasn't selected), and sets $\Phi_{<i>} = \{s_{i_1} P, s_{i_1 i_2} P, \cdots, s_{i_1 i_2 \cdots i_l} P\}$, then computes $V = \varphi \lambda Q + \sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) P'$.

It is because:

$$\begin{aligned} V &= s_\epsilon H_2(Q) + \textstyle\sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) H_2(Q) + r H_3(i_1 i_2 \cdots i_l || M, U) \\ &= s_\epsilon P' + \textstyle\sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) P' + s_\epsilon \varphi(\lambda P - (1/\varphi) P') \\ &= \varphi \lambda Q + \textstyle\sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) P' \end{aligned}$$

Finally, A sends $sign = (U, V, \Phi_{<i>})$ to F. Obviously, A can provide the signature to F though she can't compute $\alpha P' = \alpha\beta P$.

2. If $i = i^*$, selects $\lambda, \varphi \in_R Z_q^*$, and computes $h = \lambda P$, $U = \varphi Q$, and adds $(< i^* > ||M, U, h, \lambda, \varphi)$ to the H_3 table. A has set $Q_{i_1^* i_2^* \cdots i_l^*} = 1/h_{i_1^* i_2^* \cdots i_l^*}(y_{i_1^* i_2^* \cdots i_l^*} P - P')$. She selects $s_{i_1^* i_2^* \cdots i_j^*} \in_R Z_q^* (1 \le j \le l-1)$ (if hasn't selected) and sets $\Phi_{<i^*>} = \{s_{i_1^*} P, s_{i_1^* i_2^*} P, \cdots, s_{i_1^* i_2^* \cdots i_{l-1}^*} P, Q_{i_1^* i_2^* \cdots i_l^*}\}$, then computes

$V = \sum_{j=1}^{l-1} s_{i_1^* i_2^* \cdots i_j^*} H_1(i_1^* i_2^* \cdots i_j^*, Q_{i_1^* i_2^* \cdots i_j^*}) P' + y_{i_1^* i_2^* \cdots i_l^*} P' + \lambda U$

It is because:

$$\begin{aligned} V &= S_{<i^*>} + r H_3(i_1^* i_2^* \cdots i_l^* || M, U) \\ &= s_\epsilon H_2(Q) + \textstyle\sum_{j=1}^{l} s_{i_1^* i_2^* \cdots i_j^*} H_1(i_1^* i_2^* \cdots i_j^*, Q_{i_1^* i_2^* \cdots i_j^*}) H_2(Q) \\ &\quad + r H_3(i_1^* i_2^* \cdots i_l || M, U) \\ &= s_\epsilon H_2(Q) + \textstyle\sum_{j=1}^{l-1} s_{i_1^* i_2^* \cdots i_j^*} H_1(i_1^* i_2^* \cdots i_j^*, Q_{i_1^* i_2^* \cdots i_j^*}) H_2(Q) \\ &\quad + s_{i_1^* i_2^* \cdots i_l^*} H_1(i_1^* i_2^* \cdots i_l^*, Q_{i_1^* i_2^* \cdots i_l^*}) H_2(Q) + rh \end{aligned}$$

$= s_\epsilon P' + \sum_{j=1}^{l-1} s_{i_1^* i_2^* \cdots i_j^*} H_1(i_1^* i_2^* \cdots i_j^*, Q_{i_1^* i_2^* \cdots i_j^*}) P'$
$+1/h_{i_1^* i_2^* \cdots i_l^*}(y_{i_1^* i_2^* \cdots i_l^*} - s_\epsilon) h_{i_1^* i_2^* \cdots i_l^*} P' + r\lambda P$
$= \sum_{j=1}^{l-1} s_{i_1^* i_2^* \cdots i_j^*} H_1(i_1^* i_2^* \cdots i_j^*, Q_{i_1^* i_2^* \cdots i_j^*}) P' + y_{i_1^* i_2^* \cdots i_l^*} P' + \lambda U$
Finally, A sends $sign = (U, V, \Phi_{<i>})$ to F.

Break-in Phase. When F finishes the cma phase and comes to the break-in phase, A does as follows in order to provide SK_{i^*} to F:

A has known $S_{<i^*>} = \sum_{j=1}^{l-1} s_{i_1^* i_2^* \cdots i_j^*} H_1(i_1^* i_2^* \cdots i_j^*, Q_{i_1^* i_2^* \cdots i_j^*}) P' + y_{i_1^* i_2^* \cdots i_l^*} P'$, $\Phi_{<i^*>} = \{s_{i_1^*} P, s_{i_1^* i_2^*} P, \cdots, s_{i_1^* i_2^* \cdots i_{l-1}^*} P, Q_{i_1^* i_2^* \cdots i_l^*}\}$. For each j such as $i_j = 0$ and $(1 \le j \le l)$, A has set $Q_{i_1^* i_2^* \cdots i_{j-1}^* 1} = 1/h_{i_1^* i_2^* \cdots i_{j-1}^* 1}(y_{i_1^* i_2^* \cdots i_{j-1}^* 1} P - P')$, and then sets $S_{i_1^* i_2^* \cdots i_{j-1}^* 1} = \sum_{m=1}^{j-1} s_{i_1^* i_2^* \cdots i_m^*} H_1(i_1^* i_2^* \cdots i_m^*, Q_{i_1^* i_2^* \cdots i_m^*}) P' + y_{i_1^* i_2^* \cdots i_{j-1}^* 1} P'$, A sets $Set_{<i^*>} = \bigcup_{1 \le j \le l, i_j = 0} \{< S_{i_1^* i_2^* \cdots i_{j-1}^* 1}, Q_{i_1^* i_2^* \cdots i_{j-1}^* 1} >\}$.

Since

$$\begin{aligned}
S_{i_1^* i_2^* \cdots i_{j-1}^* 1} = \; & s_\epsilon H_2(Q) + \sum_{m=1}^{j-1} s_{i_1^* i_2^* \cdots i_m^*} H_1(i_1^* i_2^* \cdots i_m^*, Q_{i_1^* i_2^* \cdots i_m^*}) H_2(Q) \\
& + s_{i_1^* i_2^* \cdots i_{j-1}^*} H_1(i_1^* i_2^* \cdots i_{j-1}^* 1, Q_{i_1^* i_2^* \cdots i_{j-1}^* 1}) H_2(Q) \\
= \; & s_\epsilon P' + \sum_{m=1}^{j-1} s_{i_1^* i_2^* \cdots i_m^*} H_1(i_1^* i_2^* \cdots i_m^*, Q_{i_1^* i_2^* \cdots i_m^*}) P' \\
& + 1/h_{i_1^* i_2^* \cdots i_{j-1}^* 1}(y_{i_1^* i_2^* \cdots i_{j-1}^* 1} - s_\epsilon) h_{i_1^* i_2^* \cdots i_{j-1}^* 1} P' \\
= \; & \sum_{m=1}^{j-1} s_{i_1^* i_2^* \cdots i_m^*} H_1(i_1^* i_2^* \cdots i_m^*, Q_{i_1^* i_2^* \cdots i_m^*}) P' + y_{i_1^* i_2^* \cdots i_{j-1}^* 1} P'
\end{aligned}$$

Therefore, she can provide the secret key $SK_{i^*} = (S_{<i^*>}, Set_{<i^*>}, \Phi_{<i^*>})$ in this period to F.

Forge Phase. After above procedure, if A guesses the correct break-in period i^*, F outputs a valid forgery $sign = (U, V, \Phi_{<i>})$ for $< i, M >$. The probability to output the valid forgery without querying H_3 hash oracle at a point $(< i > ||M, U)$ is negligible. So F can find the tuple $(< i > ||M, U, h, \lambda, *)$ in H_3 table. Then

$\hat{e}(P, V) = \hat{e}(Q + \sum_{j=1}^{l} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) Q_{i_1 i_2 \cdots i_j}, H_2(Q))$
$\cdot \hat{e}(U, H_3(i_1 i_2 \cdots i_l || M, U))$ $\Rightarrow$

$\hat{e}(P, V) = \hat{e}(s_\epsilon P + \sum_{j=1}^{l} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) s_{i_1 i_2 \cdots i_j} P, H_2(Q))$
$\cdot \hat{e}(rP, H_3(i_1 i_2 \cdots i_l || M, U))$ $\Rightarrow$

$\hat{e}(P, V) = \hat{e}(P, s_\epsilon H_2(Q) + \sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) H_2(Q))$
$\cdot \hat{e}(P, rH_3(i_1 i_2 \cdots i_l || M, U))$ $\Rightarrow$

$V = s_\epsilon H_2(Q) + \sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) H_2(Q)$
$+ rH_3(i_1 i_2 \cdots i_l || M, U)$ $\Rightarrow$

$V = s_\epsilon P' + \sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) P' + rH_3(i_1 i_2 \cdots i_l || M, rP)$ $\Rightarrow$

$V = \alpha\beta P + \sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) P' + rh$ $\Rightarrow$

$V = \alpha\beta P + \sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) P' + r\lambda P$ $\Rightarrow$

$\alpha\beta P = V - \sum_{j=1}^{l} s_{i_1 i_2 \cdots i_j} H_1(i_1 i_2 \cdots i_j, Q_{i_1 i_2 \cdots i_j}) P' + \lambda U$

Thus, A can succeed in computing $\alpha\beta P$. The construction completes.

The running time analysis. The total running time of A contains the running time t of F plus the following time.

(1) The time of selecting H_1 function: there are $O(l)$ times of inverse operations on Z_q^* , multiplicative and minus operations in G_1. Therefore, the total time is at most $O((k^2+k^n)\texttt{log}T)$.

(2) The time of direct H_3 oracle queries: there are at most q_{hash} multiplicative operations on G_1 in the total. So the total time is at most $O(q_{hash} \cdot k^n)$.

(3) The time of signing oracle queries: There need q_{sig} times of inverse operations on Z_q^*, multiplicative and minus operations in G_1 for the indirect H_3 oracle queries generated by signing queries. So it needs $O((k^2+k^n)q_{sig})$ time. In addition, $q_{sig}O(l)$ times of multiplicative and additive operations in G_1 are needed. It needs $O(q_{sig} \cdot \texttt{log}T \cdot k^n)$ time. So the total time in these cases is $O(q_{sig} \cdot \texttt{log}T \cdot k^n + q_{sig}k^2)$.

(4) The time of break-in queries: There need $O(l)$ times of inverse operations on Z_q^* , multiplicative, plus and minus operations in G_1. So the total time is at most $O((k^2+k^n)\texttt{log}T)$.

(5) The time of solving CDH problem: The total time is no more than $O(\texttt{log}T \cdot k^n)$.

As we have analyzed, the total running time of A is at most $t' = t + O(max\{q_{sig}logT, q_{hash}\} \cdot k^n + max\{logT, q_{sig}\} \cdot k^2)$.

The success probability analysis. We analyze the following three events and compute the probability for A to succeed.

Event E_1**:** When F queries the signature oracle, A aborts. There is $Pr[E_1] \leq (q_{hash}-1) \cdot q_{sig}/(q-1)$.

In H_3 table A maintains, the number of queries generated not by signing algorithm is $q_{hash} - q_{sig}$. Therefore, when the k-th signature query happens, in the worst case, there are at most $q_{hash} - q_{sig} + k - 1$ of H_3 queries defined. The probability for A to abort the k-th ($k \in \{1, 2, \cdots, q_{sig}\}$) signature query is at most $(q_{hash}-q_{sig}+k-1)/(q-1)$, where $q-1$ is the size of the domain from which U (actually φ) is selected (that is the elements number of Z_q^*). Let ε_k denote the event that A aborts the k-th signature query. The following description is right:

$Pr[E_1] = Pr[\varepsilon_1 \cup \varepsilon_2 \cdots \cup \varepsilon_{q_{sig}}] \leq \sum_{k=1}^{q_{sig}} Pr[\varepsilon_k] = \sum_{k=1}^{q_{sig}} \frac{(q_{hash}-q_{sig}+k-1)}{q-1} = \frac{q_{sig}(q_{hash}-\frac{1}{2}q_{sig}-\frac{1}{2})}{q-1} \leq \frac{q_{sig}(q_{hash}-1)}{q-1}$

Event E_2**:** F outputs $d = break$ and the break-in phase is period i^*. There is $Pr[E_2] = 1/T$.

F can't distinguish the simulation given by A from the real world, so the probability that period b which A guesses is equal to the period in which F enters her break-in phase is $1/T$.

Event E_3**:** When A doesn't abort, F succeeds to forge a valid signature for a new message in period j, where $1 \leq j < b$. Obviously, there is $Pr[E_3] \geq \varepsilon$.

Therefore, the probability for A to solve CDH problem is at least:

$$\begin{aligned} Pr[E_2] \cdot Pr[\bar{E_1}] \cdot Pr[E_3] &\geq \frac{1}{T}(\varepsilon - \varepsilon Pr[E_1]) \geq \frac{1}{T}(\varepsilon - Pr[E_1]) \\ &\geq \frac{1}{T}(\varepsilon - \frac{q_{sig}(q_{hash}-1)}{q-1}) = \varepsilon' \end{aligned}$$

Therefore, the theorem follows.

Theorem 3. *Let* KE-SIG$[k,T]$ *present our forward secure signature scheme with security parameter* k *and the total number of time periods* T*. For any* t*,* q_{sig} *and* q_{hash}*, the following relation holds:*

$$\text{Insec}^{\text{fsig}}(\text{KE-SIG}[k,T],t,q_{sig},q_{hash}) \leq T \cdot \text{Insec}^{\text{CDH}}(k,t') + \frac{q-1}{q_{sig}(q_{hash}-1)}$$

where $t' = t + O(max\{q_{sig}\log T, q_{hash}\} \cdot k^n + max\{\log T, q_{sig}\} \cdot k^2)$.

Proof. Let $\texttt{Insec}^{\texttt{CDH}}(k,t') = \epsilon'$, that is, there is not an adversary A that can (t',ϵ')-break CDH problem in group G_1, From theorem 2, we can know that for any adversary F, the following relation holds:

$$\begin{aligned} &\texttt{Succ}^{\texttt{fsig}}(\texttt{KE-SIG}[k,T],F) \leq \epsilon = T\varepsilon' + \frac{q-1}{q_{sig}(q_{hash}-1)} \\ &= T \cdot \texttt{Insec}^{\texttt{CDH}}(k,t') + \frac{q-1}{q_{sig}(q_{hash}-1)} \end{aligned}$$

So we can get

$$\texttt{Insec}^{\texttt{fsig}}(\texttt{KE-SIG}[k,T],t,q_{sig},q_{hash}) \leq T \cdot \texttt{Insec}^{\texttt{CDH}}(k,t') + \frac{q-1}{q_{sig}(q_{hash}-1)}$$

$\texttt{where}\ t' = t + O(max\{q_{sig}\texttt{log}T, q_{hash}\} \cdot k^n + max\{\texttt{log}T, q_{sig}\} \cdot k^2)$.

6 Conclusions

Forward secure signature can protect the security of signatures pertaining to previous periods even if the current secret key is exposed. This paper constructs yet another forward secure signature scheme using bilinear maps, which has a nice average performance, that is, there is no cost parameters having a complexity more than $O(\log T)$. In addition, the verifying algorithm only needs three times of pairing operations. The security of the scheme is based on CDH assumption. We prove that the constructed scheme is forward secure in random oracle assuming CDH problem is hard.

Acknowledgments. We would like to thank anonymous referees of the second international conference on provable security (ProvSec 2008) for the suggestions to improve this paper. This research is supported by National Natural Science Foundation of China (60703089) and the National High-Tech R & D Program (863 Program) of China (2006AA012110).

References

1. Anderson, R.: Two remarks on public key cryptology. Invited Lecture. In: The 4th ACM Conference on Computer and Communications Security (1997)
2. Bellare, M., Miner, S.: A forward-secure digital signature scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
3. Abdalla, M., Reyzin, L.: A new forward-secure digital signature scheme. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 116–129. Springer, Heidelberg (2000)
4. Krawczyk, H.: Simple forward-secure signatures for any signature scheme. In: the 7th ACM Conference on Computer and Communications Security, pp. 108–115. ACM Press, New York (2000)
5. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 499–514. Springer, Heidelberg (2001)
6. Kozlov, A., Reyzin, L.: Forward-secure signatures with fast key update. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 247–262. Springer, Heidelberg (2003)
7. Maklin, T., Micciancio, D., Miner, S.: Efficient generic forward-secure signatures with an unbounded number of time periods. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 400–417. Springer, Heidelberg (2002)
8. Itkis, G.: Forward Security: Adaptive Cryptography-Time Evolution. The Handbook of Information Security (2005), http://www.cs.bu.edu/faculty/itkis/pap/forward-secure-survey.pdf
9. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
10. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
11. Hu, F., Wu, C.H., Irwin, J.D.: A new forward secure signature scheme using bilinear maps. Cryptology ePrint Archive, Report 2003/188 (2003)
12. Kang, B.G., Park, J.H., Halm, S.G.: A new forward secure signature scheme. Cryptology ePrint Archive, Report 2004/183 (2004)
13. Vo, D.L., Kim, K.: Yet another forward secure signature from bilinear pairings. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 441–455. Springer, Heidelberg (2006)
14. Yu, J., Kong, F.Y., Cheng, X.G., Hao, R., Li, G.W.: Cryptanalysis of Vo-Kim Forward Secure Signature in ICISC2005. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324. Springer, Heidelberg (2008)
15. Camenisch, J., Koprowski, M.: Fine-grained forward-secure signature schemes without random oracles. Discrete Applied Mathematics 154(2), 175–188 (2006)
16. Boyen, X., Shacham, H., Shen, E., Waters, B.: Forward Secure Signatures with Untrusted Update. In: The 13th ACM conference on Computer and communications security, pp. 191–200. ACM Press, New York (2006)
17. Libert, B., Jacques, J., Yung, M.: Forward-Secure Signatures in Untrusted Update Environments: Efficient and Generic Constructions. In: The 14th ACM conference on Computer and communications security, pp. 266–275. ACM Press, New York (2007)
18. Bellare, M., Yee, B.: Forward-security in private-key cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (2003)

19. Abdalla, M., Miner, S., Namprempre, C.: Forward-secure threshold signature schemes. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 441–456. Springer, Heidelberg (2001)
20. Tzeng, Z.J., Tzeng, W.G.: Robust forward signature schemes with proactive security. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 264–276. Springer, Heidelberg (2001)
21. Wang, H., Qiu, G., Feng, D., Xiao, G.: Cryptanalysis of Tzeng-Tzeng Forward-Secure Signature Schemes. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E89-A(3), 822–825 (2006)
22. Yu, J., Kong, F., Hao, R.: Forward Secure Threshold Signature Scheme from Bilinear Pairings. In: Wang, Y., Cheung, Y.-m., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 587–597. Springer, Heidelberg (2007)
23. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
24. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong key-insulated signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Heidelberg (2002)
25. Zhou, Y., Cao, Z., Chai, Z.: Identity Based Key Insulated Signature. In: Chen, K., Deng, R., Lai, X., Zhou, J. (eds.) ISPEC 2006. LNCS, vol. 3903, pp. 226–234. Springer, Heidelberg (2006)
26. Libert, B., Quisquater, J., Yung, M.: Parallel Key-Insulated Public Key Encryption Without Random Oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 298–314. Springer, Heidelberg (2007)
27. Itkis, G., Reyzin, L.: SiBIR: Signer-base intrusion-resilient signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 499–514. Springer, Heidelberg (2002)
28. Dodis, Y., Franklin, M., Katz, J., Miyaji, A., Yung, M.: Intrusion resilient public-key encryption. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 19–32. Springer, Heidelberg (2003)
29. Dodis, Y., Franklin, M., Katz, J., Miyaji, A., Yung, M.: A generic construction for intrusion-resilient public-key encryption. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 81–98. Springer, Heidelberg (2004)
30. Itkis, G.: Intrusion-resilient signature: Generic constructions, or Defeating a strong adversary with minimal assumption. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 102–118. Springer, Heidelberg (2003)
31. Indyk, P.: A Small Approximately Min-Wise Independent Family of Hash Functions. Journal of Algorithms 38(1), 84–90 (2001)

Optimal Online/Offline Signature: How to Sign a Message without Online Computation

Fuchun Guo[1] and Yi Mu[2]

[1] Key Lab of Network Security and Cryptology
School of Mathematics and Computer Science
Fujian Normal University, Fuzhou, China
fuchunguo1982@gmail.com
[2] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Wollongong NSW 2522, Australia
ymu@uow.edu.au

Abstract. We propose a novel notion of signature named Optimal Online/Offline Signature. The new notion can be seen as an extension to the notion of online/offline signature, where our signature scheme allows all necessary computations to be carried out in the offline phase before the message is available and the signer does not need to conduct any computation to construct the final signature in the online phase. Although the same feature can be achieved from a one-time signature scheme, the large signature size of a one-time signature is a disadvantage. In this paper, we provide a solution that allows our signature to be aggregated into a short length (about 320 bits); hence it demonstrates a better applicability. We also give a generic construction and then extend it to an identity-based scenario.

1 Introduction

Digital signature schemes play an important role in information security and have exhibited many applications. In a usual signature generation, the signer inputs his private signing key and a message into a secure computer that then outputs the corresponding signature with some computation according to the signing algorithm. Such computation requires a private computer with a reasonable computation power. In practice, there could be a situation, where such a computer is not available.

Let us take a look at the following scenario. Suppose Bob is a signer and uses his secure private PC as the secure device for signing messages. When he has to travel, he can carry his private signing key along with the signing software stored in a USB drive. His signing could be conducted on a public computer. Unfortunately, the public computer can not be trusted. What Bob wishes is that he could conduct the signing without the help of a powerful computer.

If we allow Bob to carry a low-power device such as a PDA, then Bob could product a signature efficiently using a general online/offline signature scheme

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 98–111, 2008.

where the signing only required a simple computation through the PDA. The notion of online/offline signature was first introduced by Even, Goldreich and Micali [7], and the idea is to perform the signature generation in two phases. The first phase is performed offline, before the message to be signed is given, and the second phase is performed online, after the message to be signed becomes available. Online/offline signature schemes are useful in many applications that the signer has a very limited response time or limited computation power. In 2001, Shamir and Tauman [14] introduced a "hash-sign-switch" paradigm using a "chameleon hash function" or chameleon hash and proposed a generic online/offline signature scheme. The online phase is typically very fast and can be based on modular multiplications only.

Using such a general online/offline signature scheme, Bob only has to carry out modular multiplications after messages are presented. This is not a problem if Bob has been equipped with a private PDA with a reasonable computational power. However, if Bob's low-power device cannot even handle modular arithmetic, the problem will not be resolved. It is not an easy task for Bob to conduct the modular computation to generate his secure signature by hand. It would be desirable if even modular arithmetic is not required to generate a secure signature.

We notice the nice property of one-time signature, which could be a candidate to our solution. The notion of one-time signature was first introduced by Lamport in [11]. Even *et al.* [7] later adopted the idea to construct online/offline signatures. Similarly to a normal online/offline signature, a signing operation is split into two phases, but the online phase requires no computation!

We outline the one-time signature as follows. The generation of a one-time signature requires to use cryptographic hash functions. Let $H : \{0,1\}^l \rightarrow \{0,1\}^n$ $(l \geq n)$ be a secure one-way hash function. For each time, the signer randomly chooses n pair of l-bit strings $(x_{10}, x_{11}), (x_{20}, x_{21}), \cdots, (x_{n0}, x_{n1})$ and sets the public key to be

$$\begin{pmatrix} H(x_{10})\ H(x_{20}) \cdots H(x_{n0}) \\ H(x_{11})\ H(x_{21}) \cdots H(x_{n1}) \end{pmatrix}.$$

To sign a n-bit message $M \in \{0,1\}^n$, let $M[i]$ be the ith bit of M. The signature is

$$\sigma_M = \Big(x_{1M[1]}, x_{2M[2]}, \cdots, x_{nM[n]}\Big).$$

The signature length is about $3n$ elements for a message of n bits. Therefore, it becomes impractical when the message size is large.

In this paper, we propose a new online/offline signature scheme named Optimal Online/Offline Signature. For simplicity, we denote by O-3 the new signature scheme. Similarly to other online/offline signatures, the signature scheme can be divided into the offline phase (before the message is presented) and the online phase (after the message is known to the signer). Like a one-time signature, the final signature can be extracted from the pre-computed parameters and the bit-structure of the message requiring no computation in the online phase. The major difference with a traditional one-time signature is that the size of O-3 is very short.

The contributions of this paper are as follows:

- We construct an O-3 signature scheme and reduce its security to the Computational Diffie-Hellman (CDH) assumption in the standard model. Our construction follows the framework of the Waters signature scheme [15]. In his scheme, he used one n-length vector with different combinations to denote different messages, while we choose two n-length vectors in order to achieve a fixed number signature elements to satisfy the property of our O-3 signature.
- We show how to construct an O-3 signature from any provably-secure signature scheme. If the generic signature scheme is the short signature scheme [2], then, the O-3 signature will be as short as 320 bits.
- We show how to extend an O-3 signature to Identity-Based O-3 Signature. In 2006, Paterson and Schuldt [13] proposed a direct construction of Identity-Based Signature from the Waters signature scheme. Using their idea, we extend the O-3 signature to identity-based O-3 signature.

The rest of the paper is organized as follows. The detailed definition of O-3 signature and some preliminaries are provided in Section 2. A general description and construction of O-3 signature and its security proof are presented in Section 3. The generic construction of O-3 signature is described in Section 4. An identity-based O-3 signature is presented in Section 5. The conclusion is given in Section 6.

2 Definitions

In this section, we give the definition of O-3 signature scheme and define its security and complexity assumption.

2.1 O-3 Signature

Definition 1. The O-3 signature scheme can be described as the following three algorithms: KeyGen, Sign and Verify.

KeyGen: Take as input a security parameter and output a random pair of public and private keys (PK, SK).

Sign: This algorithm is divided into two phases:

- Offline Phase: On input SK, output a set of elements, denoted by $\mathcal{W}$.
- Online Phase: On input the message M and the set $\mathcal{W}$, select a set of elements $\mathcal{U}_M$ from $\mathcal{W}$ according to the message and delete unselected elements in $\mathcal{W}$. The signature on M is $\mathcal{U}_M$.

Verify: On input the signature $(M,\mathcal{U}_M)$ and the public key PK, output **accept** if $\mathcal{U}_M$ correctly maps to M; otherwise output **reject**.

Similarly to a normal signature scheme, the security of the O-3 signature scheme is modeled with unforgeability under a chosen message attack and a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ which is described as follows.

Setup: The challenger $\mathcal{C}$ runs algorithm **KeyGen** to obtain a pair of public key and secret key (PK, SK). The adversary $\mathcal{A}$ is given PK.

Queries: $\mathcal{A}$ makes a signature query on message M_i of her choice. $\mathcal{C}$ responds to each query with a signature $\mathcal{U}_{M_i}$. $\mathcal{A}$ can ask q_s message queries at most.

Forgery: $\mathcal{A}$ outputs a signature pair $(M^*, \mathcal{U}_{M^*})$ and wins the game if

1. $\mathcal{A}$ did not make a signature query on M^*;
2. $\mathcal{U}_{M^*}$ is a valid signature on M^* for PK.

We define $Adv_{\mathcal{A}}$ as the probability that $\mathcal{A}$ wins in the above game.

Definition 2. *An O-3 signature scheme is (ϵ, t, q_s)-secure against forgery under a chosen message attack if no forger (ϵ, t, q_s)-breaks it, where the forger $\mathcal{A}$ runs in time at most t and makes at most q_s signature queries with advantage $Adv_{\mathcal{A}} \leq \epsilon$.*

2.2 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. A map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is called a bilinear map if this map satisfies the following properties:

- Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$;
- Non-degeneracy: $e(g, g) \neq 1$. In other words, if g be a generator of $\mathbb{G}$, then $e(g, g)$ generates $\mathbb{G}_T$;
- Computability: It is efficient to compute $e(u, v)$ for all $u, v \in \mathbb{G}$.

2.3 Complexity

The security of our O-3 signature scheme will be reduced to the hardness of the Computational Diffie-Hellman (CDH) problem in the group in which the signature is constructed. We briefly review the definition of the CDH problem here:

Definition 3. *Let $\mathbb{G}$ be the group defined as above with a generator g and elements $g^a, g^b \in \mathbb{G}$ where a, b are selected uniformly at random from $\mathbb{Z}_p$, the CDH problem in $\mathbb{G}$ is to compute g^{ab}.*

Definition 4. *We say that the (ϵ, t)-CDH assumption holds in the group of $\mathbb{G}$ if there is no algorithm running in time t at most can solve the CDH problem in $\mathbb{G}$ with the probability at least ϵ.*

3 O-3 Signature

In this section, we show the construction of O-3 signature scheme and then prove its security under the CDH assumption in the standard model.

3.1 Scheme

Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be the bilinear map, g be the corresponding generator in $\mathbb{G}$ and p be the order of $\mathbb{G}, \mathbb{G}_T$. The O-3 signature scheme can be described as follows:

KeyGen: Randomly choose an integer $\alpha \in \mathbb{Z}_p$ and generators $g_2, u_0 \in \mathbb{G}$, and set $g_1 = g^\alpha$. Randomly choose n pair of different generators of $\mathbb{G}$:

$$\Big[(u_{10}, u_{11}), (u_{20}, u_{21}), \cdots, (u_{n0}, u_{n1})\Big].$$

The public key PK and the secret key SK are defined as the following tuple

$$PK = \Big(g, g_1, g_2, u_0, (u_{10}, u_{11}), (u_{20}, u_{21}), \cdots, (u_{n0}, u_{n1})\Big), \qquad SK = g_2^\alpha.$$

Sign: The signing is divided into the offline phase for pre-computation and the online phase for signature extraction.

- **Offline Phase**
  - Randomly choose two n-length vectors $(\mu_1, \mu_2, \cdots, \mu_n) \in \mathbb{Z}_p^n$ and $(\nu_1, \nu_2, \cdots, \nu_n) \in \mathbb{Z}_p^n$ such that

$$\mu_1 + \mu_2 + \cdots + \mu_n = \nu_1 + \nu_2 + \cdots + \nu_n = 1 \pmod{p};$$

  - Pick a random $r \in \mathbb{Z}_p$, and compute $\tau = g^r$ and the following $2n$ values:

$$(\sigma_{i0}, \sigma_{i1}) = \Big(g_2^{\mu_i \alpha}(u_0^{\nu_i} u_{i0})^r, g_2^{\mu_i \alpha}(u_0^{\nu_i} u_{i1})^r\Big), \text{for all } i = 1, 2, \cdots, n;$$

  - Store the following $2n + 1$ elements:

$$\mathcal{W} = \Big((\sigma_{10}, \sigma_{11}), (\sigma_{20}, \sigma_{21}), \cdots, (\sigma_{n0}, \sigma_{n1}), \tau\Big).$$

- **Online Phase:** Given the message $M \in \{0,1\}^n$, let $M[i]$ be the ith bit of $M \in \{0,1\}^n$, the signature is generated as the follows:
  - Output the following $n + 1$ elements as the signature on M:

$$\mathcal{U}_M = \Big(\sigma_{1M[1]}, \sigma_{2M[2]}, \cdots, \sigma_{nM[n]}, \tau\Big);$$

  - Clear the other n elements.

Verify: Given the signature $(M, \mathcal{U}_M)$ and the public key PK, the verification is as the following:

– Output

$$\sigma_\tau = \prod_{i=1}^{n} \sigma_{iM[i]} = \prod_{i=1}^{n} g_2^{\mu_i \alpha} (u_0^{\nu_i} u_{iM[i]})^r = g_2^{\alpha} \Big(u_0 \prod_{i=1}^{n} u_{iM[i]} \Big)^r;$$

– Accept the signature if the following equation holds

$$e(\sigma_\tau, g) = e(g_1, g_2) e(u_0 \prod_{i=1}^{n} u_{iM[i]}, \tau);$$

– The final signature on M is (σ_τ, τ).

Efficiency: The signing cost is $2n$ multi-exponentiations and one exponentiation in the offline phase but null in the online phase. The verification cost is n multiplications and two bilinear pairings when $e(g_1, g_2)$ is one of the PK parameters. Furthermore, all public parameters in PK excluding g_1 can be shared by all users in a signature system and the individual public key for a single user in this system will be as short as one element of g_1. When the bilinear pairing defined in [2] is applied in our scheme, the signature size of the two elements (σ_τ, τ) will be as short as 320 bits.

3.2 Security

We now show the security of our scheme in the standard model. The main idea of the proof is similar to the Waters IBE [15,13] scheme but more complicated. The challenge is how to simulate the pieces of signature $\sigma_{1M[1]}, \sigma_{2M[2]}, \cdots, \sigma_{nM[n]}$.

Theorem 1. *Our O-3 signature scheme is (ϵ, t, q_s)-secure, assuming that the (ϵ', t')-CDH assumption holds, where*

$$\epsilon' \geq \frac{1}{4q_s(n+1)}, \qquad t' = t + O(q_s n t_e),$$

and n is the length of message and t_e is the time for an exponentiation.

Proof. Suppose there exists a (ϵ, t, q_s)-forger $\mathcal{A}$ against our scheme. We construct an algorithm $\mathcal{B}$ that solves the CDH problem. Algorithm $\mathcal{B}$ is given as input a random triple $(g, g^a, g^b) \in \mathbb{G}^3$ and $\mathcal{B}$'s goal is to output g^{ab}. $\mathcal{B}$ works by interacting with $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ sets $m = 2q_s$ and randomly chooses an integer k between 0 and n. We assume that $m(n+1) < p$ for the given q_s and n. It then chooses two n-length vectors $\boldsymbol{x}_0 = (x_{10}, x_{20}, \cdots, x_{n0})$ and $\boldsymbol{x}_1 = (x_{11}, x_{21}, \cdots, x_{n1})$, where all the elements of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ are chosen uniformly at random from the integers between 0 and $m-1$ and a value x', chosen uniformly at random between 0

and $m-1$. Additionally, $\mathcal{B}$ chooses a random $y' \in \mathbb{Z}_p$ and two n-length vectors $\boldsymbol{y}_0 = (y_{10}, y_{20}, \cdots, y_{n0})$ and $\boldsymbol{y}_1 = (y_{11}, y_{21}, \cdots, y_{n1})$, where all the elements of $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$ are chosen at random in $\mathbb{Z}_p$.

For a message $M \in \{0,1\}^n$, let $M[i]$ be the ith bit of M. Define the following two functions

$$F(M) = x' + \sum_{i=1}^{n} x_{iM[i]} - mk,$$
$$K(M) = y' + \sum_{i=1}^{n} y_{iM[i]}.$$

Now, $\mathcal{B}$ sets the public key PK to be: $g_1 = g^a, g_2 = g^b, u_0 = g_1^{x'-mk} g^{y'}$ and

$$(u_{i0}, u_{i1}) = \left(g_1^{x_{i0}} g^{y_{i0}}, g_1^{x_{i1}} g^{y_{i1}}\right), \quad \text{for all } i = 1, 2, \cdots, n.$$

From the perspective of the adversary, the distribution of PK is identical to the real construction. Given the message M, the equation

$$u_0 \prod_{i=1}^{n} u_{iM[i]} = g_1^{F(M)} g^{K(M)}$$

holds. The PK parameters are all sent to the forger $\mathcal{A}$.

Queries: The forger $\mathcal{A}$ issues q_s signature queries in an adaptive choice. On receiving the signature query on M, if $F(M) = 0 \pmod m$, abort; otherwise, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_p$ and does the following:

- Randomly choose the n-length vector $(\nu_1, \nu_2, \cdots, \nu_n) \in \mathbb{Z}_p^n$ such that

$$\nu_1 + \nu_2 + \cdots + \nu_n = 1 \pmod p;$$

- Set another n-length vector $(\mu_1, \mu_2, \cdots, \mu_n) \in \mathbb{Z}_p^n$ to be

$$\mu_i = \frac{\nu_i(x' - mk) + x_{iM[i]}}{F(M)} \pmod p,$$

and we have

$$\begin{aligned} & \mu_1 + \mu_2 + \cdots + \mu_n \pmod p \\ = & \frac{(x' - mk)\sum_{i=1}^{n} \nu_i + \sum_{i=1}^{n} x_{iM[i]}}{F(M)} \pmod p \\ = & \frac{x' - mk + \sum_{i=1}^{n} x_{iM[i]}}{F(M)} \pmod p \\ = & 1 \pmod p; \end{aligned}$$

– $\mathcal{B}$ computes and sends the tuple of $\mathcal{U}_M = (\sigma_{1M[1]}, \sigma_{2M[2]}, \cdots, \sigma_{nM[n]}, \tau)$ as the signature of M to the forger, where

$$\sigma_{iM[i]} = g_1^{(\nu_i(x'-mk)+x_{iM[i]})r} \cdot g_2^{-\frac{y'\nu_i+y_{iM[i]}}{F(M)}} \cdot g^{(y'\nu_i+y_{iM[i]})r},$$
$$\tau = g^r \cdot g_2^{-\frac{1}{F(M)}}.$$

Let $\widetilde{r} = r - \frac{b}{F(M)}$. We have

$$\begin{aligned}
& g_1^{(\nu_i(x'-mk)+x_{iM[i]})r} \cdot g_2^{-\frac{y'\nu_i+y_{iM[i]}}{F(M)}} \cdot g^{(y'\nu_i+y_{iM[i]})r} \\
&= \left(g_1^{\nu_i(x'-mk)+x_{iM[i]}}\right)^r \cdot \left(g^{y'\nu_i+y_{iM[i]}}\right)^r \cdot g_2^{-\frac{y'\nu_i+y_{iM[i]}}{F(M)}} \\
&= g^{\frac{\nu_i(x'-mk)+x_{iM[i]}}{F(M)} \cdot ab} \cdot \left(g_1^{\nu_i(x'-mk)+x_{iM[i]}}\right)^{r-\frac{b}{F(M)}} \cdot \left(g^{y'\nu_i+y_{iM[i]}}\right)^{r-\frac{b}{F(M)}} \\
&= g_2^{\frac{\nu_i(x'-mk)+x_{iM[i]}}{F(M)} \cdot a} \left(g_1^{\nu_i(x'-mk)+x_{iM[i]}} g^{y'\nu_i+y_{iM[i]}}\right)^{r-\frac{b}{F(M)}} \\
&= g_2^{\frac{\nu_i(x'-mk)+x_{iM[i]}}{F(M)} \cdot a} \left(g_1^{(x'-mk)\nu_i} g^{y'\nu_i} g_1^{x_{iM[i]}} g^{y_{iM[i]}}\right)^{\widetilde{r}} \\
&= g_2^{\mu_i \alpha} (u_0^{\nu_i} u_{iM[i]})^{\widetilde{r}} \\
&= \sigma_{iM[i]}
\end{aligned}$$

and

$$g^r \cdot g_2^{-\frac{1}{F(M)}} = g^{r-\frac{b}{F(M)}} = g^{\widetilde{r}} = \tau.$$

So, the tuple $(\sigma_{1M[1]}, \sigma_{2M[2]}, \cdots, \sigma_{nM[n]}, \tau)$ is a correct signature on M and $\mathcal{B}$ sends it to the forger $\mathcal{A}$. There are n uniformly random and independent values in total $(\nu_1, \nu_2, \cdots, \nu_{n-1}, r)$ in the $\sigma_{1M[1]}, \sigma_{2M[2]}, \cdots, \sigma_{nM[n]}$ simulation, so the simulation is identical to the real construction from the adversary's perspective.

Forgery: If $\mathcal{B}$ does not abort in the query phase, $\mathcal{A}$ will output a valid signature (σ_τ^*, τ^*) on a message M^* with probability ϵ at least. If $F(M^*) \neq 0 \pmod p$, abort; otherwise, $F(M^*) = 0 \pmod p$ and $\mathcal{B}$ computes and outputs the challenge value g^{ab} by

$$\begin{aligned}
\frac{\sigma_\tau^*}{(\tau^*)^{K(M^*)}} &= \frac{g_2^\alpha (u_0 \prod_{i=1}^n u_{iM^*[i]})^r}{g^{rK(M^*)}} \\
&= \frac{g_2^a (g_1^{F(M^*)} g^{K(M^*)})^r}{g^{rK(M^*)}} \\
&= g^{ab},
\end{aligned}$$

which is the solution to the CDH assumption.

We have completed the simulation of the scheme. In the following step, we will analyze the bound of probability that $\mathcal{B}$ does not abort following the way

of [13]. In the phase of signature queries, it requires that for each query on M have $F(M) \neq 0 \pmod m$ and $F(M^*) = 0 \pmod p$ in the phase of forgery. Let $M_1, M_2, \cdots, M_{q_s}$ be the signature queries and M^* be the forged message, define the events A_i and A^* as

$$A_i : F(M_i) \neq 0 \pmod m$$
$$A^* : F(M^*) = 0 \pmod p.$$

According to the definition of our simulation, the probability of $\mathcal{B}$ not aborting is

$$\Pr[\neg \text{ abort}] = \Pr\Big[\bigwedge_{i=1}^{q_s} A_i \bigwedge A^*\Big].$$

The definition of $m(n+1) \leq p$ implicates the transformation from $F(M) = 0 \pmod p$ to $F(M) = 0 \pmod m$, and the definition of $F(M) = x' + \sum_{i=1}^{n} x_{iM[i]} - mk$ gives that if $F(M) = 0 \pmod m$ ($x' + \sum_{i=1}^{n} x_{iM[i]} - mk = tm$ for some t) , there exists an unique choice of k such that $F(M) = 0 \pmod p$. Since $x', \boldsymbol{x}_0, \boldsymbol{x}_1$ and k are randomly chosen, we have

$$\begin{aligned}\Pr[A^*] &= \Pr\Big[F(M^*) = 0 \pmod p \wedge F(M^*) = 0 \pmod m\Big] \\ &= \Pr\Big[F(M^*) = 0 (\text{mod } m)\Big] \Pr\Big[F(M^*) = 0 (\text{mod } p) | F(M^*) = 0 (\text{mod } m)\Big] \\ &= \frac{1}{m} \cdot \frac{1}{n+1}\end{aligned}$$

and the inequality

$$\begin{aligned}\Pr[\bigwedge_{i=1}^{q_s} A_i | A^*] &= 1 - \Pr[\bigvee_{i=1}^{q_s} \neg A_i | A^*] \\ &\geq 1 - \sum_{i=1}^{q_s} \Pr[\neg A_i | A^*].\end{aligned}$$

When A_i and A_j are evaluated in two different messages, the probabilities of $F(M_i) = 0 \pmod m$ and $F(M_j) = 0 \pmod m$ are independent since the sum of $F(M_i)$ and $F(M_j)$ will be different in one random value at least. So, we have $\Pr[\neg A_i | A^*] = \frac{1}{m}$ and then

$$\begin{aligned}\Pr[\bigwedge_{i=1}^{q_s} A_i \bigwedge A^*] &= \Pr[A^*] \Pr[\bigwedge_{i=1}^{q_s} A_i | A^*] \\ &\geq \frac{1}{m(n+1)} \cdot (1 - \frac{q_s}{m})\end{aligned}$$

and due to $m = 2q_s$, we have that

$$\Pr[\neg \text{ abort}] = \Pr[\bigwedge_{i=1}^{q_s} A_i \bigwedge A^*] \geq \frac{1}{4q_s(n+1)}.$$

If the simulation does not abort, $\mathcal{A}$ will create a valid forged signature with probability ϵ at least and $\mathcal{B}$ can compute g^{ab} from the forged signature as shown above with the probability at least $\epsilon' = \frac{\epsilon}{4q_s(n+1)}$.

The time complexity of $\mathcal{B}$ is mainly dominated by the exponentiations in signature queries. Since there are $O(n)$ exponentiations for each query, and then for the number of q_s queries, the time complexity of $\mathcal{B}$ is $t + O\Big(q_s n t_e\Big)$.

This completes the full proof. □

4 Generic Construction

In this section, we show how to achieve the generic construction of O-3 signature from any provably-secure signature scheme. The generic O-3 signature scheme consists of the four algorithms: **SysGen**, **KeyGen**, **Sign** and **Verify** for the system parameters generation, private signing key generation, signing message and signature verification, respectively.

4.1 Generic Scheme

SysGen: Let $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ be any provably secure signature scheme. Randomly choose n pair of different bit stings:

$$(u_{10}, u_{11}), (u_{20}, u_{21}), \cdots, (u_{n0}, u_{n1}) \in \{0,1\}^* \times \{0,1\}^*.$$

The system parameters are

$$SP = \Big\{\mathcal{G}, \mathcal{S}, \mathcal{V}, (u_{10}, u_{11}), (u_{20}, u_{21}), \cdots, (u_{n0}, u_{n1})\Big\}.$$

KeyGen: On input a secure parameter, run the key generation algorithm of the original signature scheme $\mathcal{G}$ to obtain a pair of public and private keys (PK, SK).

Sign: The signing is divided into the offline phase for pre-computation and the online phase for full signature extraction.

- **Offline Phase**
  - Randomly choose a bit sting $\tau \in \{0,1\}^n$ and output the following $2n$ signatures

$$(\sigma_{i0}, \sigma_{i1}) = \Big(\mathcal{S}_{SK}(\tau, u_{i0}), \mathcal{S}_{SK}(\tau, u_{i1})\Big), \text{ for all } i = 1, 2, \cdots, n;$$

  - Store the following $2n$ signatures and τ:

$$\mathcal{W} = \Big((\sigma_{10}, \sigma_{11}), (\sigma_{20}, \sigma_{21}), \cdots, (\sigma_{n0}, \sigma_{n1}), \tau\Big).$$

- **Online Phase:** Given the message $M \in \{0,1\}^n$, let $M[i]$ be the ith bit of $M \in \{0,1\}^n$, the signature is unfolded as the follows:

- Output the following n signatures and τ as the signature of M:

$$\mathcal{U}_M = \Big(\sigma_{1M[1]}, \sigma_{2M[2]}, \cdots, \sigma_{nM[n]}, \tau\Big);$$

- Clear the other n elements.

Verify: Given the signature $(M,\mathcal{U}_M)$, public key PK and system parameters SP, accept the signature if the ith signature $\sigma_{iM[i]}$ on $(\tau, u_{iM[i]})$ is correct for all $i = 1, 2, \cdots, n$.

4.2 Security and Efficiency

In the above signature construction, it requires that the signer should choose different bit strings for each signing process; otherwise, any adversary can forge a new signature easily. For example, there are two signatures on $\mathcal{U}_{M_1}$ and $\mathcal{U}_{M_2}$ with same bit string τ on 4-bit message $M_1 = 1111$ and $M_2 = 0000$, respectively, then the adversary can forge any signature because the elements in $\mathcal{U}_{M_1}$ and $\mathcal{U}_{M_2}$ are the same as the pre-computed signature elements stored in the offline phase.

From above, we know that the signer actually signs n different signatures first to combine a new final signature. Now, suppose a forger $\mathcal{A}$ can forge a valid signature $\mathcal{U}_{M^*}$ on new message M^* using random bit string τ^*. If τ^* is different from the random strings in queries phase, it is obvious that the forger outputs a forged signature on a new message; otherwise, there must have one element $\sigma_{jM[j]}$ (signature) at least in $\mathcal{U}_{M^*}$, such that it has never be unfolded to the forger. That is, it must contain a valid piece of signature on new message $(\tau^*, u_{jM[j]})$ forged by $\mathcal{A}$ herself, where τ^* is new or the signer (simulator) never sent its signature to the forger. Thus, the forgery on $\sigma_{jM[j]}$ will be contrary to the provably-secure signature and we have that the generic construction of O-3 signature scheme is secure too.

For its efficiency in verification, we can adopt a batch verification signature scheme [8,5], where the cost on verifying n signatures of different messages is less than conducting them one by one. For its length in store, we can adopt a signature scheme with the property of aggregation [4], where n different signatures on different messages can be aggregated into a single signature. A short provably-secure signature scheme with both properties can be found in [2] and our final signature length will be as short as 320 bits.

5 Identity-Based O-3 Signature

The identity-based O-3 signature scheme consists of the four algorithms: **Setup**, **KeyGen**, **Sign** and **Verify** for the setup of the system, key generation, signing and verification, respectively.

The construction of the identity-based O-3 signature is similar to the Paterson-Schuldt's identity-based signature from the Waters signature scheme. Because of this, it is not hard to achieve the security proof by following their idea and our proof in Section 3. Therefore, we omit the security proof.

Setup: Randomly choose an integer $\alpha \in \mathbb{Z}_p$ and generators $g_2, I_0, u_0 \in \mathbb{G}$, and set $g_1 = g^\alpha$. Choose at random $3n$ different generators of $\mathbb{G}$:

$$\Big[(I_1, I_2, \cdots, I_n), (u_{10}, u_{11}), (u_{20}, u_{21}), \cdots, (u_{n0}, u_{n1})\Big].$$

The public parameters *params* and the master secret key K are defined as:

$$params = \Big(g, g_1, g_2, I_0, I_1, \cdots, I_n, u_0, (u_{10}, u_{11}), \cdots, (u_{n0}, u_{n1})\Big), \quad K = g_2^\alpha.$$

KeyGen: Given the identity $ID = \{0,1\}^n$, let $I[i]$ be the ith bit of ID and let $\mathcal{I} \subset \{1, 2, \cdots, n\}$ to be the set that $I[i] = 1$. To construct the private key, randomly choose $s \in \mathbb{Z}_p$ and output

$$d_{ID} = (d_1, d_2) = \Big(g_2^\alpha (I_0 \prod_{i \in \mathcal{I}} I_i)^s, g^s\Big);$$

Sign: The signing is divided into the offline phase for pre-computation and the online phase for the signature extraction.

- **Offline Phase:**
  - Randomly choose two n-length vectors $(\mu_1, \mu_2, \cdots, \mu_n) \in \mathbb{Z}_p^n$ and $(\nu_1, \nu_2, \cdots, \nu_n) \in \mathbb{Z}_p^n$ such that

$$\mu_1 + \mu_2 + \cdots + \mu_n = \nu_1 + \nu_2 + \cdots + \nu_n = 1 \pmod{p}.$$

  - Given the private key $d_{ID} = (d_1, d_2)$ of $ID \in \{0,1\}^n$, pick a random $r \in \mathbb{Z}_p$, and compute $\tau = g^r$ and the following $2n$ values:

$$(\sigma_{i0}, \sigma_{i1}) = \Big(d_1^{\mu_i}(u_0^{\nu_i} u_{i0})^r, d_1^{\mu_i}(u_0^{\nu_i} u_{i1})^r\Big), \text{ for all } i = 1, 2, \cdots, n.$$

  - Store the following $2n + 2$ elements:

$$\mathcal{W} = \Big((\sigma_{10}, \sigma_{11}), (\sigma_{20}, \sigma_{21}), \cdots, (\sigma_{n0}, \sigma_{n1}), d_2, \tau\Big).$$

- **Online Phase:** Given the message $M \in \{0,1\}^n$, let $M[i]$ be the ith bit of $M \in \{0,1\}^n$, the signature is unfolded as the follows:
  - Output the following $n + 2$ elements as the signature on M:

$$\mathcal{U}_M = \Big(\sigma_{1M[1]}, \sigma_{2M[2]}, \cdots, \sigma_{nM[n]}, d_2, \tau\Big);$$

  - Clear the other n elements.

Verification: Given the signature $(M, \mathcal{U}_M)$ and the identity $ID \in \{0,1\}^n$, the verification is as the following:

- Output

$$\sigma_\tau = \prod_{i=1}^{n} \sigma_{iM[i]} = \prod_{i=1}^{n} d_1^{\mu_i}(u_0^{\nu_i} u_{iM[i]})^r = g_2^\alpha \Big(I_0 \prod_{i \in \mathcal{I}} I_i\Big)^s \Big(u_0 \prod_{i=1}^{n} u_{iM[i]}\Big)^r;$$

– Accept the signature if the following equation holds

$$e(\sigma_\tau, g) = e(g_1, g_2)e(I_0 \prod_{i \in \mathcal{I}} I_i, d_2)e(u_0 \prod_{i=1}^{n} u_{iM[i]}, \tau);$$

– The final signature on M is (σ_τ, d_2, τ).

6 Conclusion

In this paper, we presented the notion of O-3 signature. Similarly to a traditional online/offline signature, an O-3 signature is divided into two phases. The feature of O-3 signature is that no computation is required in the online phase. Although the same properties can be achieved from a one-time signature scheme, our signature can be very short. We also showed how to construct a generic O-3 signature and extended the notion to identity-based O-3 signature. Although the O-3 signature scheme requires to manage more parameters in the offline phase than a traditional online/offline signature, there is no doubt that it has great applicability.

Acknowledgement. The authors would like to thank the anonymous reviewers of ProvSec 2008 for their helpful comments on this work.

References

1. Boneh, D., Boyen, X.: Efficient selective-id secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
2. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
3. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
5. Camenisch, J., Hohenberger, S., Pedersen, M.: Batch Verification of Short Signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007)
6. Chen, X., Zhang, F., Susilo, W., Mu, Y.: Efficient Generic online/offline Signatures Without Key Exposure. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 18–30. Springer, Heidelberg (2007)
7. Even, S., Goldreich, O., Micali, S.: Online/offline digital signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 263–275. Springer, Heidelberg (1990)
8. Fiat, A.: Batch RSA. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 175–185. Springer, Heidelberg (1990)
9. Harn, L.: Batch verifying multiple DSA digital signatures. Electronics Letters 34(9), 870–871 (1998)

10. Harn, L.: Batch verifying multiple RSA digital signatures. Electronics Letters 34(12), 1219–1220 (1998)
11. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (October 1979)
12. Naor, D., Shenhavy, A., Woolz, A.: One-Time Signatures Revisited:Have They Become Practical?, http://eprint.iacr.org/2005/442
13. Paterson, K., Schuldt, J.: Efficient identity-based signatures secure in the standard model. In: Batten, L., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006)
14. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
15. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 320–329. Springer, Heidelberg (2005)

Round-Optimal Blind Signatures from Waters Signatures

Kristian Gjøsteen and Lillian Kråkmo

Dept. of Mathematical Sciences, NTNU

Abstract. We present a round-optimal blind signature scheme based on Waters' signature scheme. Our construction resembles that of Fischlin [10], but does not rely on generic non-interactive zero-knowledge proofs. In addition to a common reference string, our scheme requires a registered public key for the signer.

Keywords: Blind signatures, Waters signatures.

1 Introduction

The idea of blind signatures was proposed by Chaum [8] as a key ingredient for anonymous electronic cash applications. Blind signatures allow a bank to issue signatures without seeing the content of the signed documents, and at the same time prevent users from forging signatures. The security of blind signatures was first formalized by Pointcheval and Stern [17] and later by Juels, Luby and Ostrovsky [13], resulting in the notions *blindness* and *non-forgeability*. Since then, a number of blind signature schemes have been proposed, some in the random oracle model [1,4,5,17], and some without random oracles [7,10,12,14,15]. Most of the above mentioned schemes use three or more moves, and proving security under concurrent executions of the signature generation protocol has often been difficult. Notably, this problem is avoided in schemes requiring only two moves, i.e. round-optimal schemes.

Recently, Fischlin [10] proposed a round-optimal blind signature scheme in the common reference string model. This scheme uses generic non-interactive zero-knowledge (NIZK) proofs, which makes it quite impractical. Our contribution is a concrete round-optimal scheme based on Waters' signature scheme [18]. Waters' scheme is weakly unforgeable, in the sense that signatures may easily be randomized. This property makes Waters' scheme a natural starting point for constructing a blind signature scheme. In our scheme, to obtain a blind signature on a message, the user computes a commitment to the message, based on Waters' hash function. The signer's response is essentially a signature on the commitment. Finally, the user obtains a blind signature from the signer's response, by simultaneously unblinding the commitment and randomizing the resulting Waters signature.

In order to obtain provable security, the user is also required to compute a NIZK proof that the commitment was honestly generated. The proof is obtained

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 112–126, 2008.

by applying Linear encryption [6] as an extractable commitment scheme, and by compiling a suitable Σ-protocol using the technique developped by Damgård et al. [9]. Consequently, we need a common reference string and a registered public key for the signer. In typical applications of blind signature schemes, where the signer is a bank, it seems reasonable to assume that the user has access to some reliable public information about the bank. Hence we do not consider the requirement of a registered public key to be a significant inconvenience.

The main drawback with our scheme is that a moderate number of NIZK proofs must be generated and verified as part of the signature generation protocol. A major advantage, however, is that verifying signatures is no more expensive than for Waters signatures.

2 Preliminaries

2.1 Bilinear Groups

Let $\mathbf{G}$ be a multiplicative cyclic group of prime order p, and let g be a generator of $\mathbf{G}$. We say that $\mathbf{G}$ is a *bilinear group* if the group operation in $\mathbf{G}$ is efficiently computable, and if there exists a multiplicative cyclic group $\mathbf{G}_1$ of order p and an efficiently computable non-degenerate bilinear map $e : \mathbf{G} \times \mathbf{G} \rightarrow \mathbf{G}_1$, i.e. for all $u, v \in \mathbf{G}$ and $a, b \in \{0, \ldots, p-1\}$, we have $e(u^a, v^b) = e(u, v)^{ab}$, and $e(g, g) \neq 1$.

2.2 Signature Schemes and Their Security

We refer to [13] for a formal definition of a signature scheme $\mathcal{S}$, and note that we use the following notation: $\mathcal{S} = (\mathit{Gen}, \mathit{Sign}, \mathit{Verify})$, where $\mathit{Gen}(1^\tau)$ outputs (sk, pk), $\mathit{Sign}(sk, m)$ outputs σ, and $\mathit{Verify}(pk, m, \sigma)$ outputs *accept*/*reject*.

In our work we need the notion *existential unforgeability under an adaptive chosen message attack* (UF-CMA) as defined by Goldwasser, Micali and Rivest in [11], which is defined by the experiment $\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{S},A}(\tau)$, given in Figure 1. In this experiment the adversary A has access to the signing oracle $\mathcal{O}_{\mathcal{S}}$, which takes a message as input and outputs a signature on the message under sk. It is required that $\mathcal{O}_{\mathcal{S}}$ was never queried with the message m.

We define the *success rate* of A in breaking $\mathcal{S}$ with respect to UF-CMA as

$$\mathbf{Succ}^{\text{uf-cma}}_{\mathcal{S},A}(\tau) = \Pr\left[\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{S},A}(\tau) = 1\right].$$

Definition 1. *The scheme $\mathcal{S}$ is said to be (t, q, ϵ)-secure with respect to UF-CMA if no A running in time t and making at most q oracle queries has success rate at least ϵ.*

Waters' Signature Scheme: The security of Waters' signature scheme is based on the Computational Diffie Hellmann (CDH) assumption and does not rely on random oracles [18]. We review the scheme below.

Key Generation: Let $\mathbf{G}$ be a bilinear group of order p, where p has length τ, and let $e : \mathbf{G} \times \mathbf{G} \to \mathbf{G}_1$ be the corresponding efficiently computable bilinear map. To generate the public key, the algorithm Gen_W chooses a random generator $g \in \mathbf{G}$ and a random $\alpha \in \{0, \ldots, p-1\}$ and lets $g_1 = g^\alpha$. Additionally, for some appropriate n, it chooses random $g_2, u', u_1, \ldots, u_n \in \mathbf{G}$ and lets $U = (u_1, \ldots, u_n)$. The public key is (g, g_1, g_2, u', U) and the secret key is g_2^α.

Signature Generation: Upon input of a message m of length n, the algorithm $Sign_W$ chooses a random $r \in \{0, \ldots, p-1\}$ and computes the signature σ as $\sigma = (\sigma_1, \sigma_2) = (g_2^\alpha (u' \prod_{i=1}^n u_i^{m_i})^r, g^r)$, where m_i denotes the i'th bit of m.

Signature Verification: To verify a signature $\sigma = (\sigma_1, \sigma_2)$ on a message m, the algorithm $Verify_W$ checks that $e(\sigma_1, g) = e(\sigma_2, u' \prod_{i=1}^n u_i^{m_i}) \cdot e(g_1, g_2)$. If this holds, it outputs *accept*, otherwise it outputs *reject*.

Assume that a signature (σ_1, σ_2) on a message m is generated according to the above scheme. Note that, if we randomly choose $r^* \in \{0, \ldots, p-1\}$ and let $(\sigma_1^*, \sigma_2^*) = (\sigma_1 (u' \prod_{i=1}^n u_i^{m_i})^{r^*}, \sigma_2 g^{r^*})$, (σ_1^*, σ_2^*) is a new, uniformly distributed signature on m. This property is exploited in our blind signature scheme.

We refer to [18] for a formal definition of the CDH problem and the related complexity assumption. Waters' scheme is known to be (t, q, ϵ)-secure with respect to UF-CMA if the $(t, \frac{\epsilon}{16(n+1)q})$-CDH assumption holds in $\mathbf{G}$.

2.3 Public Key Encryption Schemes and Their Security

We refer to [3] for a formal definition of a public key encryption scheme $\mathcal{PKE}$, and note that we use the following notation: $\mathcal{PKE} = (Gen, Enc, Dec)$, where $Gen(1^\tau)$ outputs (sk, pk), $Enc(pk, m)$ outputs c, and $Dec(sk, c)$ outputs $m/\perp$.

We need the security notion *real-or-random indistinguishability under a chosen plaintext attack* (ROR-CPA), which is defined by the experiment $\mathbf{Exp}^{\text{ror-cpa}}_{\mathcal{PKE},A}(\tau)$ given in Figure 1. In this experiment, the adversary A has access to the oracle $\mathcal{O}^b_{\text{ror}}$ (initialized with a hidden bit b) which takes as input a message m. If $b = 0$, it outputs an encryption of a randomly chosen string of length $|m|$ under pk. A new random string is chosen for each query. If $b = 1$, it outputs an encryption of m under pk. We define the *advantage* of A in breaking $\mathcal{PKE}$ with respect to ROR-CPA as

$$\mathbf{Adv}^{\text{ror-cpa}}_{\mathcal{PKE},A}(\tau) = \left| \Pr\left[\mathbf{Exp}^{\text{ror-cpa}}_{\mathcal{PKE},A}(\tau) = 1 | b = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ror-cpa}}_{\mathcal{PKE},A}(\tau) = 1 | b = 0\right] \right|.$$

Definition 2. *The scheme $\mathcal{PKE}$ is said to be (t, q, ϵ)-secure with respect to ROR-CPA if no A running in time t and making at most q oracle queries has advantage at least ϵ.*

Linear Encryption: Linear encryption was proposed by Boneh, Boyen and Shacham in [6] as a natural extension of ElGamal encryption. While ElGamal encryption relies on the Decision Diffie Hellman (DDH) problem, Linear encryption relies on the Decision Linear Diffie Hellman (DLDH) problem, which is believed to be hard even in bilinear groups where the DDH problem is easy.

$\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{S},A}(\tau)$:

1. $(sk, pk) \leftarrow Gen(1^\tau)$.
2. $(m, \sigma) \leftarrow A^{\mathcal{O}_S}(pk)$.
3. If $Verify(pk, m, \sigma) = accept$ then return 1, otherwise return 0.

$\mathbf{Exp}^{\text{nf}}_{\mathcal{BS},A}(\tau)$:

1. $crs \leftarrow D(1^\tau)$.
2. $(pk_{KS}, sk_{KS}) \leftarrow KS(1^\tau)$.
3. $(pk, sk) \leftarrow Gen(1^\tau)$.
4. Let $A(1^\tau, crs, pk_{KS}, pk)$ engage in polynomially many (in τ) parallel interactive protocols, with polynomially many (in τ) copies of $Signer(pk, sk)$, where A decides in an adaptive manner when to stop. Let l be the number of executions, where *Signer* outputs *completed*.
5. A outputs a collection $\{(m_1, \sigma(m_1)), \ldots, (m_k, \sigma(m_k))\}$, subject to the constraint that $m_i \neq m_j$ for $1 \leq i < j \leq k$, and $Verify(pk, m_i, \sigma(m_i))$ outputs *accept* for $1 \leq i \leq k$.

$\mathbf{Exp}^{\text{ror-cpa}}_{\mathcal{PKE},A}(\tau)$

1. $(sk, pk) \leftarrow Gen(1^\tau)$
2. $b \leftarrow \{0, 1\}$
3. $b' \leftarrow A^{\mathcal{O}^b_{\text{ror}}}(pk)$
4. Return b'.

$\mathbf{Exp}^{\text{b}}_{\mathcal{BS},A}(\tau)$:

1. $crs \leftarrow D(1^\tau)$.
2. Run A on input $(1^\tau, crs)$.
3. $pk, r, (m_0, m_1) \leftarrow A$.
4. $(sk_{KS}, pk_{KS}) \leftarrow KS(1^\tau, r)$.
5. $b \leftarrow \{0, 1\}$.
6. Let A engage in two parallel interactive protocols, the first with $User(pk, m_b)$ and the second with $User(pk, m_{1-b})$.
7. If the first *User* outputs $\sigma(m_b)$ and the second *User* outputs $\sigma(m_{1-b})$, then give $\{\sigma(m_0), \sigma(m_1)\}$ to A as additional input.
8. A outputs a bit b'.

Fig. 1. Experiments for security definitions

We refer to [6] for a formal definition of the DLDH problem and the related complexity assumption.

In the Linear encryption (LE) scheme, Gen_L outputs (sk_L, pk_L), where pk_L is a triple of randomly chosen generators $\alpha_1, \alpha_2, \beta \in \mathbf{G}$, where $\mathbf{G}$ is a group of prime order p, and p has length τ. sk_L is the exponents $a_1, a_2 \in \{0, \ldots, p-1\}$ such that $\alpha_1^{a_1} = \alpha_2^{a_2} = \beta$. Upon input of a message $m \in \mathbf{G}$, Enc_L chooses random values $r, s \in \{0, \ldots, p-1\}$, and outputs the triple $(y_1, y_2, y_3) = (\alpha_1^r, \alpha_2^s, \beta^{r+s}m)$. Upon input of a ciphertext (y_1, y_2, y_3), Dec_L outputs $y_3 y_1^{-a_1} y_2^{-a_2}$.

It can be shown that, for all t, q polynomial in τ, the LE scheme is (t, q, ϵ)-secure with respect to ROR-CPA, for some ϵ negligible in τ, if the corresponding holds for the DLDH problem in $\mathbf{G}$.

2.4 Setup Assumptions

The Common Reference String Model: Let D be a probabilistic polynomial-time algorithm which takes a security parameter 1^τ as input and outputs a value *crs*, chosen according to some publicly known distribution. In the *common reference string (CRS) model*, we assume that all parties have access to a trusted

functionality $\mathcal{F}_{\text{crs}}^{D}$, which initially runs $D(1^\tau)$ and obtains *crs*, and later gives *crs* to any party asking for it.

The Registered Public Key Model: We also review the *registered public-key model*, according to [9], and note that its relation to the common reference string model is discussed in [2].

Let KS be a probabilistic polynomial-time algorithm which takes a security parameter 1^τ as input and outputs a pair (sk, pk) of private and public keys.

In the registered public-key model, we assume that all parties have access to a trusted functionality $\mathcal{F}_{\text{reg}}^{KS}$, which can be invoked to register own key pairs and to retrieve the public keys of others. In order to register a key pair, the registrant privately sends $\mathcal{F}_{\text{reg}}^{KS}$ the random coins r used to create his key pair. $\mathcal{F}_{\text{reg}}^{KS}$ then runs $KS(1^\tau, r)$, stores the resulting public key together with the identity of the registrant, and later gives the public key to any party asking for it.

2.5 Compilation of Σ-Protocols in the Registered Public Key Model

According to [9], a *Σ-protocol for a relation R* is an interactive proof-system for the language $L_R = \{x | \exists w : (x, w) \in R\}$. The conversations are on the form (a, e, z), where a and z are messages sent by the prover P, while e is a random challenge sent by the verifier V. Additionally, a Σ-protocol has the properties *relaxed special soundness* and *special honest-verifier zero-knowledge*. We refer to [9] for a formal definition of these properties, and note that *perfect honest-verifier zero-knowledge* is a stronger variant of honest-verifier zero-knowledge, where the conversations output by the simulator are identically distributed as conversations between P and V.

We now briefly review the technique developed by Damgård et al. [9] for compiling Σ-protocols into non-interactive zero-knowledge arguments. Their technique works in the registered public-key model, and we refer to [9] for a formal definition of a *non-interactive system for the relation R with key setup KS*, along with the desired properties of such a system: *correctness, zero-knowledge* and *soundness*.

We note that the compilation technique only applies to Σ-protocols with the additional property of *linear answer*, i.e. it requires that P's final message z is a sequence of integers which are linearly obtained from the challenge e.

The high-level idea of the technique is the following: It is assumed that V has initially registered a public key pk_{KS} with the functionality $\mathcal{F}_{\text{reg}}^{KS}$ described above. pk_{KS} is on the form (pk, c), where pk is a public key of a homomorphic encryption scheme, and c is an encryption under pk of a randomly chosen challenge e. The corresponding private key is (sk, e), where sk is the private key corresponding to pk. To compute a proof, P first obtains pk_{KS} from $\mathcal{F}_{\text{reg}}^{KS}$, and computes the first message a according to the Σ-protocol. Then, P exploits the homomorphic property of the encryption scheme, and the fact that the Σ-protocol has linear answer, to obtain an encrypted response to the challenge e encrypted in c. The encrypted response may in turn be decrypted and checked

as usual by V. This technique is illustrated in Chapter 4.2, where we describe the compiled Σ-protocol used in our blind signature scheme.

As for showing that the compiled protocol has the desired properties, we note that correctness of the above system follows directly from completeness of the involved Σ-protocol. Furthermore, since a simulator running V obtains the random coins intended for $\mathcal{F}_{\text{reg}}^{KS}$, he obtains V's private key, and in particular the challenge e. Hence, to simulate a proof for a statement x, he may run an honest-verifier simulator for the Σ-protocol on input (x, e) to obtain a conversation (a, e, z) from which a correctly distributed proof is directly obtained. This means that zero-knowledge (for arbitrary verifiers) of the compiled protocol follows from honest-verifier zero-knowledge of the original Σ-protocol. Proving soundness is more involved, but it basically boils down to the assumed security of the involved encryption scheme. We refer to [9] for detailed proofs of the above properties for the general construction. As for the particular construction used in our blind signature scheme, proofs are omitted due to space limitations.

3 Blind Signature Schemes and Their Security

Our definition of a blind signature scheme corresponds to the one given by JLO in [13], modified to fit our model, where all parties are assumed to have access to the trusted functionalities $\mathcal{F}_{\text{crs}}^{D}$ and $\mathcal{F}_{\text{reg}}^{KS}$ defined earlier, and where the signer is initially required to register a public key pk_{KS} with $\mathcal{F}_{\text{reg}}^{KS}$.

Definition 3 (Blind Signature Scheme). *A blind signature scheme $\mathcal{BS}$ is a tuple $(Gen, Signer, User, Verify, KS, D)$ with the following properties:*

- *Gen is a probabilistic polynomial time algorithm, which takes as input a security parameter τ (encoded as 1^τ), and outputs a pair (pk, sk) of public and secret keys.*
- *Signer and User are a pair of polynomially-bounded probabilistic interactive Turing machines, given as common input a public key pk. In addition, Signer is given a corresponding secret key sk, and User is given a message m. The length of all inputs must be polynomial in the security parameter τ. Signer and User interact according to the protocol. At the end of the interaction, Signer outputs either completed or not completed and User outputs either fail or $\sigma(m)$.*
- *Verify is a deterministic polynomial time algorithm, which takes as input a public key pk, a message m and a signature $\sigma(m)$, and outputs either accept or reject, indicating whether $\sigma(m)$ is a valid signature on the message m.*
- *D and KS are the algorithms parameterizing $\mathcal{F}_{crs}^{D}$ and $\mathcal{F}_{reg}^{KS}$.*

It is required that for any message m, and for all key pairs (pk, sk) output by Gen, if both Signer and User follow the protocol, then Signer(pk, sk) outputs completed, User(pk, m) outputs $\sigma(m)$, and Verify$(pk, m, \sigma(m))$ outputs accept.

JLO formalized the security of blind signature schemes using the notions *blindness* and *non-forgeability*. Informally, a scheme has blindness if it is infeasible

for a malicious signer to determine the order of which two messages are signed by interaction with an honest user. A scheme has non-forgeability if, given l interactions with an honest signer, it is infeasible for a malicious user to produce more than l valid signatures.

Non-forgeability for blind signature schemes in our model is formally defined using the experiment $\mathbf{Exp}^{\mathrm{nf}}_{\mathcal{BS},A}(\tau)$ given in Figure 1. We point out that this is *weak* non-forgeability, since we require that the k signatures output by the adversary correspond to different messages. A is said to win the experiment if $k > l$. We define the *success rate* of the adversary A in breaking $\mathcal{BS}$ with respect to non-forgeability as

$$\mathbf{Succ}^{\mathrm{nf}}_{\mathcal{BS},A}(\tau) = \Pr\left[k > l\right].$$

Definition 4 (Non-forgeability). *The scheme $\mathcal{BS}$ is said to be (t, q, ϵ)-secure with respect to non-forgeability if no A running in time t, engaging in at most q protocols where Signer outputs completed, has success rate at least ϵ.*

As for blindness in our model, we need the experiment $\mathbf{Exp}^{\mathrm{b}}_{\mathcal{BS},A}(\tau)$ given in Figure 1. We define the *advantage* of A in breaking $\mathcal{BS}$ with respect to blindness as

$$\mathbf{Adv}^{\mathrm{b}}_{\mathcal{BS},A}(\tau) = \left|\Pr\left[b' = 1 | b = 1\right] - \Pr\left[b' = 1 | b = 0\right]\right|.$$

Definition 5 (Blindness). *The scheme $\mathcal{BS}$ is said to be (t, ϵ)-secure with respect to blindness if no A running in time t has advantage at least ϵ.*

4 Our Blind Signature Scheme

In this section we present our blind signature scheme, which is based on Waters' signature scheme. We note that, since the signatures generated by our scheme are in fact Waters signatures, we only obtain weak non-forgeability, as signatures may easily be randomized.

A construction similar to ours was earlier proposed by Okamoto [15]. However, Okamoto's scheme is less concrete, and the possibility of turning the scheme into a round-optimal one is not addressed.

4.1 A Sketch of Our Scheme

We start by briefly outlining the idea of our scheme. To obtain a blind signature on a message m, the user commits to m and sends the resulting commitment c to the user, along with a proof π that c was honestly generated. The signer responds with (σ'_1, σ'_2), which is essentially a signature on the commitment. In the final step, the user obtains a blind signature (σ_1, σ_2) on m, by simultaneously unblinding the commitment and randomizing the resulting signature. The signature generation protocol is sketched in Figure 2, where we note that the key pair (sk_{BS}, pk_{BS}) is generated exactly as (sk_W, pk_W) in Waters' signature scheme.

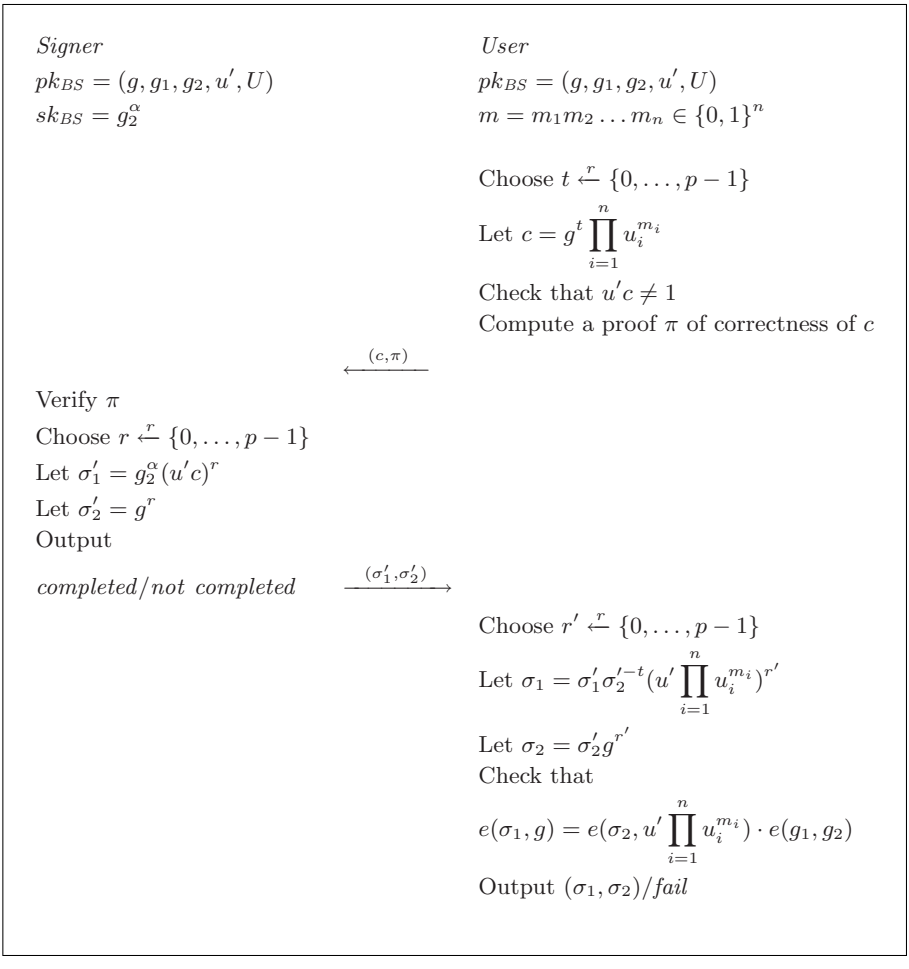

Fig. 2. The signature generation protocol

We proceed by explaining the high-level idea of the proof π. Let A^{nf} be an adversary trying to break the non-forgeability of our scheme. In order to achieve provable security, π is constructed such that a simulator running a copy of A^{nf} can extract the message m and the exponent t. This extractability is obtained by having the user commit to m and t by encrypting them, using a public key obtained from the common reference string. Given m and t, the simulator can use a signing oracle for Waters' signature scheme to obtain a correctly distributed response (σ_1', σ_2'), hence the non-forgeability of our scheme reduces to the UF-CMA security of Waters' scheme.

The proof π should convince the verifier (in this case *Signer*) that c was honestly generated. If we let $t_{\tau-1} \dots t_o$ be the bit representation of t, that is, $t = \sum_{i=0}^{\tau-1} t_i 2^i$, this amounts to proving that $c = g^{\sum_{i=0}^{\tau-1} t_i 2^i} \prod_{i=1}^{n} u_i^{m_i}$ for known

bits t_i, $0 \leq i \leq \tau - 1$, and m_i, $1 \leq i \leq n$. This is obtained by having the prover (in this case *User*) commit to each of the values g^{t_i}, $0 \leq i \leq \tau - 1$, and $u_i^{m_i}$, $1 \leq i \leq n$, prove correctness of each of the involved commitments, and then prove that c in fact contains the committed values.

Let $Enc_L(m, r, s)$ denote the Linear encryption of the message $m \in G$ with randomness (r, s) under the public key $(\alpha_1, \alpha_2, \beta)$. Recall that, with this notation, $Enc_L(m, r, s) = (\alpha_1^r, \alpha_2^s, \beta^{r+s} m)$, and note that this encryption scheme is homomorphic, i.e.

$$Enc_L(m, r, s) \cdot Enc_L(m', r', s') = Enc_L(mm', r + r', s + s).$$

The prover commits to g^{t_i}, $0 \leq i \leq \tau - 1$, by choosing random values $r_i, s_i \in \{0, \ldots, p - 1\}$ and computing

$$T_i = Enc_L(g^{t_i}, r_i, s_i) = (\alpha_1^{r_i}, \alpha_2^{s_i}, \beta^{r_i + s_i} g^{t_i}).$$

Similarly, the prover commits to $u_i^{m_i}$, $1 \leq i \leq n$, by choosing random values $r_i', s_i' \in \{0, \ldots, p - 1\}$ and computing

$$M_i = Enc_L(u_i^{m_i}, r_i', s_i') = (\alpha_1^{r_i'}, \alpha_2^{s_i'}, \beta^{r_i' + s_i'} u_i^{m_i}).$$

Correctness of each of the above commitments can be proved using a suitable Σ-protocol. Moreover, since we want our proof π to be non-interactive, we apply the technique developped by Damgård et al. for compiling Σ-protocols into NIZK proofs.

As for proving that c in fact contains the committed values, note that, by letting $r^* = \sum_{i=0}^{\tau-1} r_i 2^i + \sum_{i=1}^{n} r_i'$ and $s^* = \sum_{i=0}^{\tau-1} s_i 2^i + \sum_{i=1}^{n} s_i'$, we have

$$\prod_{i=0}^{\tau-1} T_i^{2^i} \prod_{i=1}^{n} M_i = (\alpha_1^{r^*}, \alpha_2^{s^*}, \beta^{r^* + s^*} g^t \prod_{i=1}^{n} u_i^{m_i}),$$

i.e. $\prod_{i=0}^{\tau-1} T_i^{2^i} \prod_{i=1}^{n} M_i$ is a commitment to $g^t \prod_{i=1}^{n} u_i^{m_i}$. This means that the prover can prove the correctness of c by opening this commitment, that is, by including r^* and s^* in π. In this way, assuming that the verifier has accepted all of the above NIZK proofs, he can conclude that c was honestly generated if and only if

$$Enc_L(c, r^*, s^*) = (\alpha_1^{r^*}, \alpha_2^{s^*}, \beta^{r^* + s^*} c) = \prod_{i=0}^{l-i} T_i^{2^i} \prod_{i=1}^{n} M_i.$$

4.2 The Protocol *compile*(Σ_{OR})

The proof π in our blind signature scheme includes proofs of correctness of several commitments, all on the form $(y_1, y_2, y_3) = (\alpha_1^r, \alpha_2^s, \beta^{r+s} u^b)$, where α_1, α_2, β and u are publicly known elements of a bilinear group $\mathbf{G}$ of known prime order p, $r, s \in \{0, \ldots, p - 1\}$ are secret exponents, and b is a secret bit. We

proceed by constructing a Σ-protocol for proving correctness of a commitment on the above form. To this end, we apply the so-called *OR-construction*, briefly reviewed here according to [9]. Given Σ-protocols Σ_l and Σ_r for relations R_l and R_r, the OR-construction yields a Σ-protocol Σ_{OR} for the relation R_{OR} defined by

$$((x_l, x_r), (w_l, w_r)) \in R_{OR} \Leftrightarrow (x_l, w_l) \in R_l \vee (x_r, w_r) \in R_r.$$

Let R_L be the relation defined by

$$(x, w) = ((x_1, x_2, x_3), (r, s)) \in R_L \Leftrightarrow x_1 = \alpha_1^r, x_2 = \alpha_2^s, x_3 = \beta^{r+s}.$$

We note that, for a commitment (y_1, y_2, y_3), we have

$$(y_1, y_2, y_3) = (\alpha_1^r, \alpha_2^s, \beta^{r+s}) \Leftrightarrow ((y_1, y_2, y_3), (r, s)) \in R_L$$

and

$$(y_1, y_2, y_3) = (\alpha_1^r, \alpha_2^s, \beta^{r+s} u) \Leftrightarrow ((y_1, y_2, \frac{y_3}{u}), (r, s)) \in R_L.$$

This means that a suitable protocol Σ_{OR} for our purpose is obtained by composing Σ-protocols for R_L, where we let $x_3 = y_3$ in one protocol and $x_3 = \frac{y_3}{u}$ in the other.

A Σ-protocol Σ_L for R_L is presented in Figure 3. It is straight-forward to show that Σ_L has completeness, relaxed special soundness and perfect honest-verifier Zero-Knowledge.

The protocol Σ_{OR} is constructed in a standard way from the two suitable instances of Σ_L, and it is not hard to show that completeness, relaxed special soundness and perfect honest-verifier zero-knowledge of Σ_{OR} follow from the corresponding properties of Σ_L. Due to space limitations, we do not give a complete description of Σ_{OR}. We simply note that Σ_{OR} has conversations on the form $(a, e, (z_1, z_2, \ldots, z_6))$, where z_i is linearly obtained from e for all i, $1 \leq i \leq 6$. This means that Σ_{OR} has linear answer, so we may apply the compilation technique of Damgård et al.

We now describe the compilation of the protocol Σ_{OR}. The homomorphic encryption scheme used in the compilation is Paillier's encryption scheme [16]. We refer to [16] for a description of this scheme. It is assumed that the verifier has initially registered a public key with $\mathcal{F}_{\text{reg}}^{KS}$, where the key setup algorithm KS is defined as follows:

$KS(1^\tau)$: Generate a keypair (sk_P, pk_P) for Paillier encryption, by running the key generation algorithm with 2τ as input. Choose a random challenge $e \in \{0, \ldots, p-1\}$. Also, let c be a Paillier encryption of e under pk_P. The public key is (pk_P, c) and the private key is (sk_P, e).

Due to the homomorphic property of Paillier encryption, and the fact that Σ_{OR} is with linear answer, it is possible to execute the prover's side of the protocol given only the encryption c of the challenge e. First, the prover computes his first message a. Then, denoting by $Enc_P(pk_P, m)$ a random Paillier encryption

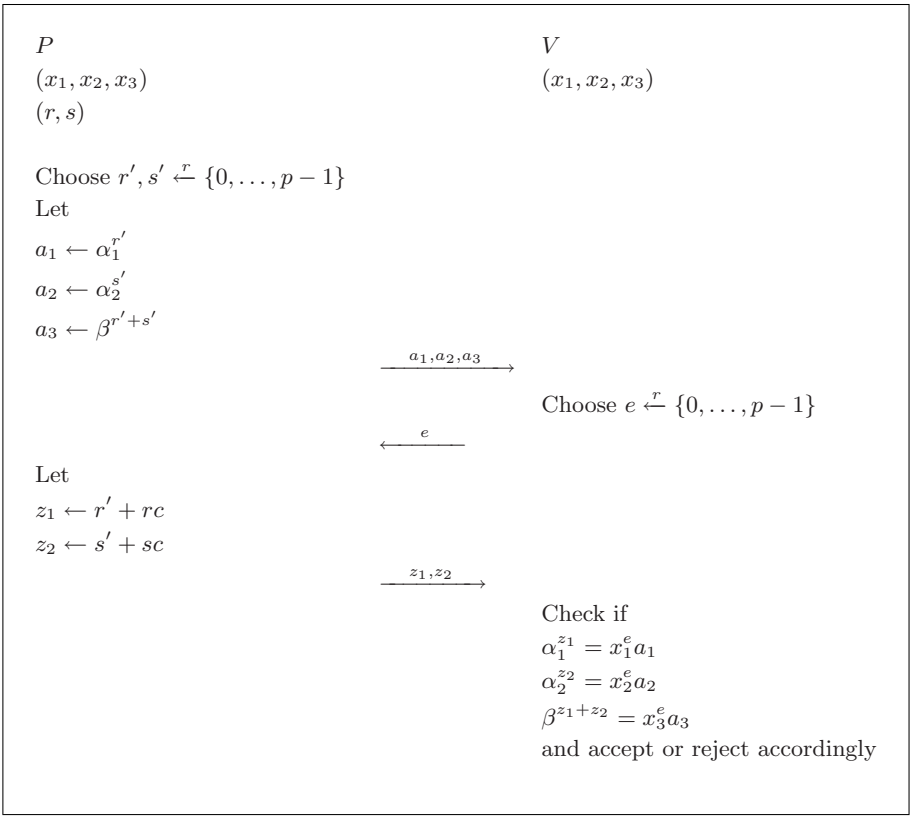

Fig. 3. The protocol Σ_L

of m under pk_P, and assuming that the component z_i of his response z is given by $z_i = u_i + v_i e$, he can compute an encryption of z_i as $Enc_P(pk_P, u_i) \cdot c^{v_i}$.

The compiled protocol is defined below. We remind the reader that Σ_{OR} has conversations on the form $(a, e, (z_1, z_2, \ldots, z_6))$, where, for each i, $1 \leq i \leq 6$, $z_i = u_i + v_i e$ for some $u_i, v_i \in \{0, \ldots, p-1\}$.

Protocol *Compile*(Σ_{OR}):

1. On input of a statement/witness pair (x, w), P gets V's public key (pk_P, c) from $\mathcal{F}_{\text{reg}}^{KS}$ and computes the first message a according to Σ_{OR}. Then, for i such that $1 \leq i \leq 6$, P computes $Enc_P((pk_P, u_i) \cdot c^{v_i})$, and lets c_i be a randomization of the resulting encryption. P sends x, π to V, where $\pi = (a, (c_1, \ldots, c_6))$.
2. On input of a statement x and a proof $\pi = (a, (c_1, \ldots, c_6))$, for i such that $1 \leq i \leq 6$, V lets z_i' be the Paillier decryption of c_i under sk. Then V verifies that the conversation $(a, e, (z_1', \ldots, z_6'))$ would be accepted by the verifier of Σ_{OR} upon input x, and accepts or rejects accordingly.

We need a result from [9] based on the following assumption: '$\mathcal{H}_{\text{Paillier}}$ is 2-harder than $\mathcal{G}_{\text{dlog}}$'. Due to space limitations, we refer to [9] for formal definitions of the involved terms. Loosely speaking, it is assumed that, given an algorithm A that solves the discrete logarithm (DLOG) problem for moduli of length τ, there is no algorithm that breaks Paillier encryption for moduli of length 2τ, with runtime comparable to that of A. We note that, as defined in [9], $\mathcal{G}_{\text{dlog}}$ addresses the DLOG problem in the subgroup of $\mathbf{Z}_p$ of order p', where p and p' are primes, and $p = 2p' + 1$. Since our Σ-protocol involves a bilinear group, we need a modified version of $\mathcal{G}_{\text{dlog}}$, addressing the DLOG problem in a general bilinear group of prime order. We call this modified version $\mathcal{G}^*_{\text{dlog}}$. By arguing as in [9], the following assumption seems reasonable.

Assumption 1: $\mathcal{H}_{\text{Paillier}}$ is 2-harder than $\mathcal{G}^*_{\text{dlog}}$.

We obtain the following result, analogous to Theorem 1, Theorem 2 and Corollary 1 in [9].

Theorem 1. *compile*(Σ_{OR}) *has correctness and perfect zero-knowledge (in the registered public-key model). Furthermore, under Assumption 1,* compile(Σ_{OR}) *is sound for* $\mathcal{O}(\log \tau)$ *executions.*

The proof of the above theorem is omitted due to space limitations. We note that, when it comes to proving soundness, we have slightly modified the definition in [9], so it better suits our application. That is, we consider the following experiment, where $\tilde{P}$ is a probabilistic, polynomial-time adversary:

1. $(sk_{KS}, pk_{KS}) \xleftarrow{r} KS(1^\tau)$
2. Run $\tilde{P}$ on input $(1^\tau, pk_{KS})$.
3. Repeat until $\tilde{P}$ stops: $\tilde{P}$ outputs (x_i, π_i), $1 \leq i \leq m(k)$, for some polynomial m. Run $V(1^\tau, x, \pi_i, sk_{KS})$, $1 \leq i \leq m(k)$. If $V(1^\tau, x_i, \pi_i, sk_{KS}) = 1$ for all i, then give 1 to $\tilde{P}$, otherwise give 0 to $\tilde{P}$.

In the original experiment, $\tilde{P}$ is only allowed to output one statement/proof pair at a time. The reason why we allow for a polynomial number of pairs is that the prover in our blind signature scheme does not learn whether each pair is accepted or not.

4.3 Our Scheme

A detailed description of our blind signature scheme is given below. Recall that, in our model, it is assumed that *Signer* and *User* have access to the trusted functionalities $\mathcal{F}^D_{\text{crs}}$ and $\mathcal{F}^{KS}_{\text{reg}}$ defined earlier. Furthermore, *Signer* is initially required to register a public key pk_{KS} with the functionality $\mathcal{F}^{KS}_{\text{reg}}$.

Common Reference String: The algorithm D takes 1^τ as input, randomly chooses $\alpha_1, \alpha_2, \beta \leftarrow \mathbf{G}$ and lets $crs \leftarrow \alpha_1 || \alpha_2 || \beta$.

Key Setup: The algorithm KS works exactly as in $compile(\Sigma_{OR})$, i.e. the public and secret keys are $pk_{KS} = (pk_P, c)$ and $sk_{KS} = (sk_P, e)$.

Key Generation: The algorithm Gen works exactly as in Waters' signature scheme, i.e. the public and secret keys are $pk_{BS} = (g, g_1, g_2, u', U)$ and $sk_{BS} = g_2^\alpha$.

Signature Generation: *User* takes (pk_{BS}, m) as input and lets $m_1 m_2 \dots m_n$ be the bit representation of m. He randomly chooses $t \leftarrow \{0, \dots, p-1\}$, and lets $c \leftarrow g^t \prod_{i=1}^n u_i^{m_i}$. He checks that $u'c \neq 1$. If this holds, he continues. Otherwise, he starts over, choosing a new t. Then he generates a proof π of correctness of c as follows: Let $t_{\tau-1} \dots t_o$ be the bit representation of t, that is, $t = \sum_{i=0}^{\tau-1} t_i 2^i$. *User* gets $crs = \alpha_1 || \alpha_2 || \beta$ from $\mathcal{F}_{\text{crs}}^D$, chooses random values $r_i, s_i \in \{0, \dots, p-1\}$ and computes, for all i, $0 \leq i \leq \tau - 1$,

$$T_i = (\alpha_1^{r_i}, \alpha_2^{s_i}, \beta^{r_i + s_i} g^{t_i}).$$

Similarly, he chooses random values $r_i', s_i' \in \{0, \dots, p-1\}$ and computes, for all i, $1 \leq i \leq n$,

$$M_i = (\alpha_1^{r_i'}, \alpha_2^{s_i'}, \beta^{r_i' + s_i'} u_i^{m_i}).$$

He also computes

$$r^* = \sum_{i=0}^{\tau-1} r_i 2^i + \sum_{i=1}^{n} r_i', \quad s^* = \sum_{i=0}^{\tau-1} s_i 2^i + \sum_{i=1}^{n} s_i'.$$

User then computes, for all i, $0 \leq i \leq \tau - 1$, a proof π_{T_i} according to $Compile(\Sigma_{OR})$ on input $(T_i, (r_i, s_i))$. Moreover, for all i, $1 \leq i \leq n$, he computes a proof π_{M_i} according to $Compile(\Sigma_{OR})$ on input $(M_i, (r_i', s_i'))$. Finally, he lets $\pi = ((T_0, \pi_{T_0}), \dots, (T_{\tau-1}, \pi_{T_{\tau-1}}), (M_1, \pi_{M_1}), \dots, (M_n, \pi_{M_n}), r^*, s^*)$, sends (c, π) to *Signer* and waits.

Signer gets (pk_{BS}, sk_{BS}) as input. Upon receiving (c, π) from *User*, he verifies π by the following procedure: First, he verifies each of the pairs (T_i, π_{T_i}), $0 \leq i \leq \tau - 1$, and (M_i, π_{M_i}), $1 \leq i \leq n$, according to $Compile(\Sigma_{OR})$. If all pairs are accepted, he continues. Otherwise, he outputs *not completed* and stops. He gets $crs = \alpha_1 || \alpha_2 || \beta$ from $\mathcal{F}_{\text{crs}}^D$ and checks if

$$(\alpha_1^{r^*}, \alpha_2^{s^*}, \beta^{r^* + s^*} c) = \prod_{i=0}^{\tau-1} T_i^{2^i} \prod_{i=1}^{n} M_i.$$

If this holds, he concludes that c was honestly generated and continues. Otherwise, he outputs *not completed* and stops. He randomly chooses $r \leftarrow \{0, \dots, p-1\}$. He then lets $\sigma_1' = g_2^\alpha (u'c)^r$ and $\sigma_2' = g^r$, outputs *completed*, sends (σ_1', σ_2') to *User* and stops.

Upon receiving (σ_1', σ_2') from *Signer*, *User* randomly chooses $r' \leftarrow \{0, \dots, p-1\}$. He lets $\sigma_1 = \sigma_1' \sigma_2'^{-t} (u' \prod_{i=1}^n u_i^{m_i})^{r'}$ and $\sigma_2 = \sigma_2' g^{r'}$. He then checks if

$$e(\sigma_1, g) = e(\sigma_2, u' \prod_{i=1}^{n} u_i^{m_i}) \cdot e(g_1, g_2).$$

If this holds, he outputs (σ_1, σ_2) and stops. Otherwise, he outputs *fail* and stops.

Signature Verification: The algorithm *Verify* works exactly as in Waters' scheme, i.e. to verify a signature $\sigma = (\sigma_1, \sigma_2)$ on a message m, it checks that

$$e(\sigma_1, g) = e(\sigma_2, u' \prod_{i=1}^{n} u_i^{m_i}) \cdot e(g_1, g_2).$$

If this holds, it outputs *accept*, otherwise it outputs *reject*.

We obtain the following results.

Theorem 2. *Under Assumption 1, and if Waters' scheme is* (q', t', ϵ')*-secure with respect to UF-CMA, then our blind signature scheme is* (l, t, ϵ)*-secure with respect to non-forgeability, where* $q' = l = \mathcal{O}(\log \tau)$*,* $t' = t + poly(\tau)$ *and* $\epsilon' = (1 - \rho)\epsilon$*, where* ρ *is a negligible function in* τ*.*

Theorem 3. *If the LE scheme is* $(1, t', \epsilon')$*-secure with respect to ROR-CPA, then our blind signature scheme is* (t, ϵ)*-secure with respect to blindness, where* $t' = t + poly(\tau)$*, and* $\epsilon' = \frac{\epsilon}{2n+2\tau}$*.*

Due to space limitations, the proofs of the above results are not included. As explained earlier, the non-forgeability of our scheme reduces to the UF-CMA security of Waters' scheme, since a simulator can extract the message m and the exponent t from a proof π computed by the user. As for blindness, the proof uses a standard hybrid argument for reducing the blindness of our scheme to the ROR-CPA security of the LE-scheme. That is, we assume that there is an adversary A^{b} against the blindness of our blind signature scheme, and define a series of games. In the first game, A^{b} runs in the experiment $\mathbf{Exp}^{\mathrm{b}}_{\mathcal{BS},A}(\tau)$, where $b = 0$, and in the last game, A^{b} runs in the experiment $\mathbf{Exp}^{\mathrm{b}}_{\mathcal{BS},A}(\tau)$, where $b = 1$. We show that, if A^{b} can distinguish two consecutive games, then we can construct an adversary breaking the ROR-CPA security of the LE scheme.

We note that the restriction on l in Theorem 2 is due to the restriction to $\mathcal{O}(\log \tau)$ executions in Theorem 1. However, the authors of [9] show that, under a stronger non-standard assumption, their compiled protocol for proving equality of discrete logarithms is sound for an arbitrary polynomial number of executions. Their strategy also applies to the protocol $compile(\Sigma_{OR})$. Hence, under this assumption, we achieve non-forgeability for an arbitrary polynomial l. We refer to [9] for more details.

References

1. Abe, M.: A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 136–151. Springer, Heidelberg (2001)
2. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally Composable Protocols with Relaxed Set-Up Assumptions. In: FOCS 2004: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004), Washington, DC, USA, pp. 186–195. IEEE Computer Society Press, Los Alamitos (2004)

3. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
4. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 319–338. Springer, Heidelberg (2002)
5. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
6. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures, pp. 41–55 (2004)
7. Camenisch, J., Koprowski, M., Warinschi, B.: Efficient Blind Signatures Without Random Oracles. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 134–148. Springer, Heidelberg (2005)
8. Chaum, D.: Blind Signatures for Untraceable Payments. In: Advances in Cryptology-Crypto 1982, pp. 199–203 (1982)
9. Damgård, I., Fazio, N., Nicolosi, A.: Non-Interactive Zero-Knowledge from Homomorphic Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 41–59. Springer, Heidelberg (2006)
10. Fischlin, M.: Round-Optimal Composable Blind Signatures in the Common Reference String Model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117. Springer, Heidelberg (2006)
11. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 17(2), 281–308 (1988)
12. Hazay, C., Katz, J., Koo, C.-Y., Lindell, Y.: Concurrently-Secure Blind Signatures Without Random Oracles or Setup Assumptions. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
13. Juels, A., Luby, M., Ostrovsky, R.: Security of Blind Digital Signatures (Extended Abstract). In: McCurley, K.S., Ziegler, C.D. (eds.) Advances in Cryptology 1981 - 1997. LNCS, vol. 1440, pp. 150–164. Springer, Heidelberg (1999)
14. Kiayias, A., Zhou, H.-S.: Concurrent Blind Signatures Without Random Oracles. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 49–62. Springer, Heidelberg (2006)
15. Okamoto, T.: Efficient Blind and Partially Blind Signatures Without Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
16. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
17. Pointcheval, D., Stern, J.: Provably Secure Blind Signature Schemes. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 252–265. Springer, Heidelberg (1996)
18. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 114–127. Springer, Heidelberg (2008)

Secure Proxy Multi-signature Scheme in the Standard Model

Zhenhua Liu[1,2], Yupu Hu[2], and Hua Ma[1]

[1] Applied Mathematics Department,Xidian University,
Xi'an, Shaanxi 710071,China
[2] The Ministry of Education Key Laboratory of Computer Networks
and Information Security, Xidian University, Xi'an 710071, China
zhualiu@hotmail.com, yphu@mail.xidian.edu.cn

Abstract. In electronic world, proxy signature is a solution of delegation of signing capabilities. Proxy multi-signature schemes allow a proxy signer to generate a proxy signature on behalf of two or more original signers. However, the security of the known proxy multi-signature schemes is proven in the random oracle which does not imply security in the real world. In this paper, we present a proxy multi-signature scheme in the standard model. The size of a proxy multi-signature is independent of the number of the original signers. Our scheme is existentially unforgeable against chosen message attack and chosen warrant attack based on the hardness of the well known CDH problem in the standard model.

Keywords: proxy multi-signature, standard model, bilinear pairings, provable security.

1 Introduction

The concept of proxy signature was first introduced by Mambo, Usuda and Okamoto in 1996[12]. In the proxy signature scheme, an original signer is allowed to authorize a designated person as his proxy signer. Then the proxy signer is able to sign on behalf of the original signer. Since then, many proxy signature schemes have been proposed [2,4,9,11,13,14,17]. Proxy signatures can combine other special signatures to obtain some new types of proxy signatures. Till now, there are various kinds of proxy signature schemes have been proposed [6,10,19,21,22].

Proxy multi-signature was also introduced by Yi et al. in 2000 [20]. In a proxy multi-signature scheme, a designated proxy signer can generate the signature on behalf of a group of original signers. Proxy multi-signatures can play important role in the following scenario: A company releases a document that may involve the financial department, engineering department, and program office, etc. The document must be signed jointly by these entities, or signed by a proxy signer authorized by these entities. One solution to the later case of this problem is to use a proxy multi-signature scheme.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 127–140, 2008.

Although many new proxy multi-signature schemes and improvements have been proposed[7,5,3,8,18] since 2000, most of them do not fully meet the desired security requirement and their security are all argued by presenting attacks that fail, which only provide very weak guarantee. In 2003, Boldyreva, Palacio, and Warinschi[2] presented the formal definition and security notion for proxy signature, i.e., the existential unforgeability against an adaptive chosen-message attack, which was the first work on proxy signature in the provable security direction. In 2006, the formal security model for proxy multi-signature schemes is presented due to Cao and Cao[3], Wang and Cao [18], respectively. Their model is an adapted version of the model of Boldyreva, Palacio and Warinschi[2]. One drawback of both proxy multi-signature schemes is that they are provably secure only in the random oracle model and thus there is only a heuristic argument for their security. Consequently, to design a provably secure proxy multi-signature scheme based on standard intractability assumptions is both of theoretical and practical importance.

As a natural extension of the efforts to provide secure schemes without the use of random oracles, we give the first proxy multi-signature scheme that is provably secure based on the difficulty of CDH problem in the standard model. Our construction derive from novel adaptations of the signature scheme of Waters [16] and the ID-based signature scheme of Paterson and Schuldt [15].

The paper will proceed as follows. In Section 2 we will give some preliminary works. Section 3 recalls the general proxy multi-signature schemes and secure model. We will present our proxy multi-signature scheme in the standard model in Section 4 and provide its formal security analysis in Section 5. Finally, we conclude our paper in Sections 6.

2 Preliminaries

we review some fundamental backgrounds required in this paper, namely bilinear pairing and complexity assumption.

2.1 Bilinear Pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of prime order p and let g be a generator of $\mathbb{G}$. The map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear pairing if the following three conditions hold true:

(1) e is bilinear, i.e. $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
(2) e is non-degenerate, i.e. $e(g, g) = 1_{\mathbb{G}_T}$.
(3) e is efficiently computable.

We say that $(\mathbb{G}, \mathbb{G}_T)$ are bilinear groups if there exists the bilinear pairing, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ as above, and e, and the group action in $\mathbb{G}$ and $\mathbb{G}_T$ can be computed efficiently. See [1] for more details on the construction of such pairings.

2.2 Complexity Assumption

Definition 1. Computational Diffie Hellman (CDH) Problem in $\mathbb{G}$. Given $g, g^a, g^b \in \mathbb{G}$ for some unknown $a, b \in \mathbb{Z}_p$, compute $g^{ab} \in \mathbb{G}$. The success probability of a polynomial algorithm $\mathcal{C}$ in solving the CDH problem in $\mathbb{G}$ is denoted:

$$Succ_{\mathcal{C},\mathbb{G}}^{CDH} = Pr[C(g, g^a, g^b) = g^{ab} : a, b \in \mathbb{Z}_p]$$

Definition 2. Computational Diffie Hellman (CDH) Assumption in $\mathbb{G}$. Given $g, g^a, g^b \in \mathbb{G}$, for some unknown $a, b \in \mathbb{Z}_p$, $Succ_{\mathcal{C},\mathbb{G}}^{CDH}$ is negligible.

3 Formal Model of Proxy Multi-signature

We first define the general proxy multi-signature schemes. In a proxy multi-signature scheme, there are n original signers and a proxy signer. Let $A_1, A_2, \cdots, A_n$ be the n original signers, and B be a proxy signer designated by all original signers $A_i (i \in \{1, 2, \cdots, n\})$. The definition details the components of a proxy multi-signature scheme.

Definition 3. Proxy multi-signature scheme. A proxy multi-signature scheme is a tuple $PMS = (Gen, Sign, Verify, ProxyKeyGen, ProxyMultiSign, ProxyMultiVerify)$.

Gen: On input of a security parameter 1^k, the algorithm produces public/private key pairs. The public and private key pairs for the original signers and the proxy signer are $(pk_{A_1}, sk_{A_1}), \cdots, (pk_{A_n}, sk_{A_n}), (pk_B, sk_B)$.

Sign: This is a (possibly) randomized standard signing algorithm. On input of a secret key sk and a message $M \in \{0,1\}^*$, the algorithm outputs a signature σ.

Verify: This is a deterministic standard verification algorithm. On input of a message/signature pair (M, σ), the algorithm outputs 1 if σ is a valid signature for M relative to pk, and outputs 0 otherwise.

ProxyKeyGen: It is a protocol between all original signers and the proxy signer formed by a group of interactive randomized algorithms. All original signers and the proxy signer input the public keys $pk_{A_1}, pk_{A_2}, \cdots, pk_{A_n}, pk_B$, and the delegation warrant ω which includes the restrictions on the class of messages delegated, the identities of the original signers and the proxy signer, and the period of delegation, etc. Every original signer takes as input his secret key $sk_{A_i} (i \in \{1, 2, \cdots, n\})$. The proxy signer also takes as input his secret key sk_B. As a result, the proxy signer outputs a proxy signing key sk_p that the proxy signer uses it to produce proxy multi-signature on behalf of all original signers.

ProxyMultiSign: This is a (possibly) randomized algorithm. On input of a proxy signing key sk_p, a warrant $\omega \in \{0,1\}^*$ and a message $M \in \{0,1\}^*$ which satisfies ω, the algorithm outputs a proxy multi-signature $p\sigma$.

ProxyMultiVerify: This is a deterministic proxy multi-signature verification algorithm. On input of a proxy multi-signature $(pk_{A_1}, pk_{A_2}, \cdots, pk_{A_n}, pk_B, \omega, M, p\sigma)$, the algorithm outputs 1 if $p\sigma$ is a valid proxy multi-signature

for M by the proxy signer on behalf of all original signers, and outputs 0 otherwise.

3.1 Security Model

We now review the formal security model for proxy multi-signature schemes due to Cao and Cao[3]. Their model is an adapted version of the model of Boldyreva, Palacio and Warinschi[2].

In this model, we assume there are totally $n+1$ participants and there is no secure channel. In a seemingly extreme case the adversary is working against a single honest user, say user 1, and can select and register keys for all other users. Notice that this is without loss of generality since any attack that can be carried out in the presence of more honest users, can be performed by having some of the users under the control of the adversary behave honestly. So, in our model, the adversary $\mathcal{A}$ is given a public key pk_1 relative to user 1. Then $\mathcal{A}$ can choose the other n private keys $sk_2, \cdots, sk_{n+1}$ and register the corresponding public keys $pk_2, \cdots, pk_{n+1}$ to the certification authority. But he must output both of the public keys and the matching secret keys or prove his knowledge of the corresponding secret keys. The adversary is also given access to three oracles: a standard signing oracle, a delegation oracle, and a proxy multi-signature oracle. The adversary's goal is the existential forgery of a standard signature relative to pk_1 or a proxy multi-signature. His advantage, $Adv^{UF}_{PMS}(\mathcal{A})$, is defined to be his probability of success in the following game against a challenger $\mathcal{C}$.

Setup: $\mathcal{C}$ runs Gen on input 1^k to produce public/ secret key pair (pk_1, sk_1) relative to user 1, k is a security parameter, $\mathcal{C}$ keeps sk_1 secret and provides $\mathcal{A}$ with the public key pk_1.

Certification queries: $\mathcal{A}$ provides key pairs $(pk_2, sk_2), \cdots, (pk_{n+1}, sk_{n+1})$ in order to certify $pk_2, \cdots, pk_{n+1}$ which are all different from the challenged key pk_1.

Signing queries: $\mathcal{A}$ can query oracle $\mathcal{O}_S(sk_1, \cdot)$ on message M of his choice, and obtain a standard signature for M by user 1, $\sigma = Sign(sk_1, M)$. Then $\mathcal{C}$ adds the message M to a list S_{qu}.

Delegation queries

- $\mathcal{A}$ can request to interact with user 1, user 1 playing the role of one of the original signers. Without loss of generality, we assume the proxy signer is user $n+1$, and the original signers are all user $i (i \in \{1, 2, \cdots, n\})$. $\mathcal{C}$ responds by running algorithm $ProxyKeyGen$, the input of warrant ω is chosen by $\mathcal{A}$. Eventually outputs the corresponding proxy signing key skp, then (ω, skp) is added to a list $Warro$.
- $\mathcal{A}$ can request to interact with user 1, user 1 playing the role of a proxy signer. The original signers are all user $i (i \in \{2, \cdots, n+1\})$. $\mathcal{C}$ responds by running algorithm $ProxyKeyGen$, taken warrant ω of user 1 is chosen by $\mathcal{A}$ as input. Eventually, outputs a proxy private key skp, then adds (ω, skp) to a list $Warrp$. We emphasize that $\mathcal{A}$ does not have access to the element of $Warrp$.

Proxy multi-signature queries: $\mathcal{A}$ can make a query (ω, M) to the oracle $\mathcal{O}_{PMS}(skp, \cdot)$, where exist a skp such that $(\omega, skp) \in Warrp$ and M satisfies ω. Eventually output a proxy multi-signature $p\sigma$ on message M. Then the query (ω, M) is added to a list PMS_{qu}.

Output: Eventually, $\mathcal{A}$ halts, outputting any one the following forges:

- $\mathcal{A}$ outputs a valid standard signature (M^*, σ^*) where $Verify(pk_1, M^*, \sigma^*) = 1$ and M was not queried to oracle $\mathcal{O}_S(sk_1, \cdot)$,i.e. $M^* \notin S_{qu}$. So $\mathcal{A}$ forges user 1's standard signature.
- $\mathcal{A}$ outputs a valid proxy multi-signature $(pk_2, \cdots, pk_{n+1}, pk_1, \omega^*, M^*, p\sigma^*)$ on message M^* under warrant ω^* by user 1 on behalf of all users $i(i \in \{2, \cdots, n+1\})$, and $(\omega^*, M^*) \notin PMS_{qu}$. So $\mathcal{A}$ forges a valid proxy multi-signature and user 1 plays the role of proxy signer. But in fact user 1 didn't sign the message M^* under the warrant ω^* on behalf of the original signer group. This is a chosen-message attack.
- $\mathcal{A}$ outputs a valid proxy multi-signature $(pk_1, pk_2, \cdots, pk_{n+1}, \omega^*, M^*, p\sigma^*)$ on message M^* under warrant ω^*. Without loss of generality, we assume the proxy signer is user $n+1$, the original signers are users $i(i \in \{1, 2, \cdots, n\})$ (user 1 is one of the original signers), where $\omega^* \notin Warro$. So $\mathcal{A}$ forges a proxy multi-signature and user 1 plays the role of one of the original signers. But in fact user 1 didn't delegate his signing capability corresponding to ω^* to user $n+1$. This is a chosen-warrant attack.

Definition 4. Secure proxy multi-signature. A proxy multi-signature forger $\mathcal{A}(t, q_{PMS}, q_S, q_D, n, \epsilon)$ breaks an n-user proxy multi-signature scheme in an adaptive chosen-message attack and an adaptive chosen-warrant attack model if: $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_S queries to the signing queries, at most q_D queries to the delegation queries, and at most q_{PMS} queries to the proxy multi-signature queries; $Adv^{UF}_{PMS}(\mathcal{A})$ is at least ϵ; and the forged proxy multi-signature is by at most n users in which one is the proxy signers. A proxy multi-signature scheme is $(t, q_{PMS}, q_S, q_D, n, \epsilon)$-secure against existential forgery in an adaptive chosen-message attack and an adaptive chosen-warrant attack model if no forger breaks it. We formalize it as the existential unforgeability against an adaptive chosen-message attack and an adaptive chosen-warrant attack (PMS-UF-CMA-CWA).

4 A Secure Proxy Multi-signature Scheme

In this section, we will construct a new proxy multi-signature scheme in the standard model based on the schemes [15,16]. It is assumed that n original signers $A_1, A_2, \cdots, A_n$ jointly ask a proxy signer B to carry out signing a document M for them altogether, and a verifier Carol checks the validity of the created signatures.

Setup: Let $(\mathbb{G}, \mathbb{G}_T)$ be bilinear groups defined in Section 2.1, where $|\mathbb{G}| = |\mathbb{G}_T| = p$ for some prime p, g is the generator of $\mathbb{G}$. e denotes the bilinear pairing $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The messages M to be signed in this

scheme will be represented as bitstrings of length m. Furthermore, picks $2m+2$ random elements $u', v', u_1, u_2, \cdots, u_m, v_1, v_2, \cdots, v_m \in_R \mathbb{G}$ and set $\mathbf{u} = (u_1, u_2, \cdots, u_m)$, $\mathbf{v} = (v_1, v_2, \cdots, v_m)$. Then the common parameter $Para = (\mathbb{G}, \mathbb{G}_T, p, g, e, u', v', \mathbf{u}, \mathbf{v})$.For $1 \leq \forall i \leq n$, A_i has public key $pk_a^{(i)} = (pk_{ax}^{(i)}, pk_{ay}^{(i)}) = (g^{x_a^{(i)}}, g^{y_a^{(i)}})$ and secret key $sk_a^{(i)} = (sk_{ax}^{(i)}, sk_{ay}^{(i)}) = (x_a^{(i)}, y_a^{(i)})$ such that $x_a^{(i)}, y_a^{(i)} \in_R \mathbb{Z}_p^*$. Similarly, B's public key is $pk_b = (pk_{bx}, pk_{by}) = (g^{x_b}, g^{y_b})$ and the secret key is $sk_b = (sk_{bx}, sk_{by}) = (x_b, y_b)$.

Sign: Let M be an m-bit message to be signed and M_j denote the j-th bit of M, and $\mathcal{M} \subseteq \{1, 2, \cdots, m\}$ be the set of all j for which $M_j = 1$, the standard signature is generated as follows. First, a random $r \in \mathbb{Z}_p^*$ is chosen. Then the standard signature is constructed as: $\sigma = (\sigma_1, \sigma_2)$ where $\sigma_1 = g^{sk_x sk_y}(u' \prod_{j \in \mathcal{M}} u_j)^r, \sigma_2 = g^r$. Here sk_x, sk_y denote the secret key of the signer.

Verify: Check whether $\sigma = (\sigma_1, \sigma_2)$ is a signature for a message M. The signature is accepted if

$$e(\sigma_1, g) = e(pk_x, pk_y)e(u' \prod_{j \in \mathcal{M}} u_j, \sigma_2)$$

ProxyKeyGen

- *Warrant:* For delegating the signing capability to the proxy signer, the original signers first make a warrant ω which includes the restrictions on the class of messages delegated, the original signers and proxy signer's identities and public keys, the period of validity,etc.
- *SubProxyKeyGen:* Let ω be an m-bit message to be signed by the the original signer A_i and ω_j denote the j-th bit of ω, and $\mathcal{W} \subseteq \{1, \cdots, m\}$ be the set of all j for which $\omega_j = 1$, a signature is generated as follows. A random $r_i \in \mathbb{Z}_p^*$ is chosen, then the signature is constructed as:

$$\sigma_\omega^{(i)} = (\sigma_{\omega_1}^{(i)}, \sigma_{\omega_2}^{(i)}) = (g^{x_a^{(i)} y_a^{(i)}}(u' \prod_{j \in \mathcal{W}} u_j)^{r_i}, g^{r_i})$$

  then sends $(\omega, \sigma_\omega^{(i)})$ to the proxy signer B.
- *SubProKeyVerify:* After received $(\omega, \sigma_\omega^{(i)})$, B checks whether

$$e(\sigma_{\omega_1}^{(i)}, g) = e(pk_{ax}^{(i)}, pk_{ay}^{(i)})e(u' \prod_{j \in \mathcal{W}} u_j, \sigma_{\omega_2}^{(i)})$$

  If $(\omega, \sigma_\omega^{(i)})$ is valid, he accept it as a valid proxy and continues; otherwise, he requests a valid one from A_i, or he terminates this protocol.
- *ProxyKeyGen:* If B confirms the validity of all $(\omega, \sigma_\omega^{(i)})(i = 1, 2, \cdots, n)$, he computes $\sigma_\omega = (\sigma_{\omega_1}, \sigma_{\omega_2}) = (\prod_{i=1}^{n} \sigma_{\omega_1}^{(i)}, \prod_{i=1}^{n} \sigma_{\omega_2}^{(i)})$. Then picks a random number $r' \in \mathbb{Z}_p^*$, he computes the proxy signing key as:

$$sk_p = (sk_{p_1}, sk_{p_2}) = (g^{x_b y_b}\sigma_{\omega_1}(u' \prod_{j \in \mathcal{W}} u_j)^{r'}, \sigma_{\omega_2} g^{r'})$$

ProxyMultiSign: Let M be an m-bit message to be signed by the original signers $A_1, A_2, \cdots, A_n$ and M_k denote the k-th bit of M, and $\mathcal{M} \subseteq \{1, 2, \cdots, m\}$ be the set of all k for which $M_k = 1$, the proxy signature is generated as follows. A random value $r_m \in \mathbb{Z}_p$ is chosen, then the signature is constructed as:

$$p\sigma=(p\sigma_1, p\sigma_2, p\sigma_3) = (sk_{p_1}(v' \prod_{k\in\mathcal{M}} v_k)^{r_m}, sk_{p_2}, g^{r_m})$$
$$=(g^{\sum_{i=1}^{n} x_a^{(i)} y_a^{(i)}} g^{x_b y_b} (u' \prod_{j\in\mathcal{W}} u_j)^{\sum_{i=1}^{n} r_i + r'} (v' \prod_{k\in\mathcal{M}} v_k)^{r_m}, g^{\sum_{i=1}^{n} r_i + r'}, g^{r_m})$$

ProxyMultiVerify: Given the public keys $(pk_a^{(1)}, pk_a^{(2)}, \cdots, pk_a^{(n)}, pk_b)$, a warrant $\omega \in \{0,1\}^m$, a message $M \in \{0,1\}^m$, and a signature $p\sigma = (p\sigma_1, p\sigma_2, p\sigma_3)$, a verifier accept $p\sigma$ if the following equality holds:

$$e(p\sigma_1, g) = \prod_{i=1}^{n} e(pk_{ax}^{(i)}, pk_{ay}^{(i)}) e(pk_{bx}, pk_{by}) e(u' \prod_{j\in\mathcal{W}} u_j, p\sigma_2) e(v' \prod_{k\in\mathcal{M}} v_k, p\sigma_3)$$

It is easy to see that a signature constructed with the **ProxyMultiSign** algorithm will be accepted by a verifier. Thus the scheme is correct.

5 Security Analysis

We will prove that our proxy multi-signature scheme is existentially unforgeable under an adaptive chosen-message and adaptive chosen- warrant attack in the standard model, given that the computational Diffie-Hellman problem is hard, by using the approach of [16,15].

Theorem 1. The proxy multi-signature scheme of Section 4 is $(t, q_S, q_{PMS}, q_D, \epsilon)$-secure, assuming that (ϵ', t')-CDH assumption holds in $\mathbb{G}$, where:

$$\epsilon' \geq \frac{\epsilon}{16 q_{PMS}(q_S + q_{PMS} + q_D)(m+1)^2}$$

$$t' = t + \mathcal{O}((m(q_S + q_{PMS} + q_D) + n(q_{PMS} + q_D))\rho + (q_S + q_{PMS} + q_D)\tau)$$

and ρ and τ are the time of a multiplication and an exponentiation in $\mathbb{G}$, respectively.

Proof. Suppose there exists an $(t, q_S, q_{PMS}, q_D, \epsilon)$-forger $\mathcal{A}$ for our proxy multi-signature scheme. We will construct an algorithm $\mathcal{C}$ which will use $\mathcal{A}$ to solve the CDH problem. The algorithm $\mathcal{C}$ will be given a group $\mathbb{G}$, a generator g and the elements g^a and g^b. To be able to use $\mathcal{A}$ to compute g^{ab}, $\mathcal{C}$ must be able to simulate a challenger for $\mathcal{A}$. $\mathcal{C}$ will response $\mathcal{A}$'s queries as following.

Setup: Let $l_1 = 2(q_{PMS} + q_S + q_D)$ and $l_2 = 2q_{PMS}$. $\mathcal{C}$ randomly chooses (1) two integers k_1 and k_2 $(0 \leq k_1, k_2 \leq m)$. We will assume that

$l_1(m+1) < p$ and $l_2(m+1) < p$ for the given values of q_{PMS}, q_S and m.
(2) an integer $x' \in \mathbb{Z}_{l_1}$; an m-dimensional vector $\mathbf{X} = (x_j)(x_j \in \mathbb{Z}_{l_1})$
(3) an integer $z' \in \mathbb{Z}_{l_2}$; an m-dimensional vector $\mathbf{Z} = (z_k)(z_k \in \mathbb{Z}_{l_2})$
(4) two integers $y', w' \in \mathbb{Z}_p$; an m-dimensional vector $\mathbf{Y} = (y_j)(y_j \in \mathbb{Z}_p)$ and m-dimensional vector $\mathbf{W} = (w_k)(w_k \in \mathbb{Z}_p)$
For ease of analysis, we define the following functions:

$$F(M) = (p - l_1 k_1) + x' + \sum_{j \in \mathcal{M}} x_j \qquad \text{and} \qquad J(M) = y' + \sum_{j \in \mathcal{M}} y_j$$
$$K(M) = (p - l_2 k_2) + z' + \sum_{k \in \mathcal{M}} z_k \qquad \text{and} \qquad L(M) = w' + \sum_{k \in \mathcal{M}} w_k$$

Then the challenger assigns a set of public parameters as follows.

$$g_1 = g^a \qquad g_2 = g^b$$
$$u' = g_2^{(p-l_1k_1)+x'} g^{y'} \qquad u_j = g_2^{x_j} g^{y_j} (1 \le j \le m)$$
$$v' = g_2^{(p-l_2k_2)+z'} g^{w'} \qquad v_k = g_2^{z_k} g^{w_k} (1 \le k \le m)$$

Note that these public parameters will have the same distribution as in the game between the challenger and $\mathcal{A}$. $\mathcal{C}$ assigns the user 1's public key $pk_a^{(1)} = (pk_{ax}^{(1)}, pk_{ay}^{(1)}) = (g_1, g_2)$ where g^a, g^b are the input of the CDH problem. Hence for any message M, the following equations:

$$u' \prod_{j \in \mathcal{M}} u_j = g_2^{F(M)} g^{J(M)} \qquad \text{and} \qquad v' \prod_{k \in \mathcal{M}} v_k = g_2^{K(M)} g^{L(M)}$$

hold. Then $\mathcal{C}$ returns user 1's public key $pk_a^{(1)}$, all public parameters to $\mathcal{A}$. $\mathcal{C}$ runs $\mathcal{A}$ on input $pk_a^{(1)}$, answering all of $\mathcal{A}$'s queries in any order and any number of times as follow. The lists $Warro, S_{qu}, PMS_{qu}, Warrp$ are all empty at first and maintained by $\mathcal{C}$:

Certification queries: $\mathcal{A}$ wishes to certify some public key $pk_a^{(i)} (i \in \{2, \cdots, n+1\})$, providing also its corresponding private key $sk_a^{(i)} (i \in \{2, \cdots, n+1\})$. $\mathcal{C}$ checks that the private key is indeed the correct one and if so registers $(pk_a^{(i)}, sk_a^{(i)}) (i \in \{2, \cdots, n+1\})$ in its list of certified keypairs.

Signature queries: $\mathcal{A}$ issues standard signature queries with $pk_a^{(1)}$ on message M, and let M be an m-bit message and M_j denote the j-th bit of M, and $\mathcal{M} \subseteq \{1, \cdots, m\}$ be the set of all j such that $M_j = 1$. $\mathcal{C}$ responds to this query by Oracle $\mathcal{O}_S(sk_a^{(1)}, \cdot)$ as follows. Upon receiving an answer σ, $\mathcal{C}$ forwards the response to $\mathcal{A}$. The message M is added to a list S_{qu}.

- If $F(M) \neq 0 \bmod l_1$, $\mathcal{C}$ can construct a signature by choosing a random $r \in \mathbb{Z}_p$ and computing:

$$\sigma = (\sigma_1, \sigma_2) = (g_1^{-J(M)/F(M)} (u' \prod_{j \in \mathcal{M}} u_j)^r, g_1^{-1/F(M)} g^r)$$

Writing $\widetilde{r} = r - a/F(M)$, it can be verified that defining σ in this manner yields a valid signature on M, since:

$$\sigma_1 = g_1^{-J(M)/F(M)}(u' \prod_{j\in\mathcal{M}} u_j)^r = g_1^{-J(M)/F(M)}(g_2^{F(M)} g^{J(M)})^r$$
$$= g_2^a (g_2^{F(M)} g^{J(M)})^{-a/F(M)} (g_2^{F(M)} g^{J(M)})^r = g_2^a (g_2^{F(M)} g^{J(M)})^{r-a/F(M)}$$
$$= g_2^a (u' \prod_{j\in\mathcal{M}} u_j)^{\widetilde{r}}$$

and $\sigma_2 = g_1^{-1/F(M)} g^r = g^{r-a/F(M)} = g^{\widetilde{r}}$

– If $F(M) = 0 \bmod p, \mathcal{C}$, the above computation cannot be performed and the simulator will abort. To make the analysis of the simulation easier, we will force the simulator to abort whenever $F(M) = 0 \bmod l_1$. Since $l_1(m+1) < p$ implies $0 \le l_1 k_1 < p$ and $0 \le x' + \sum_{j\in\mathcal{M}} x_j < p$, it is easy to see that $F(M) = 0 \bmod p$ implies that $F(M) = 0 \bmod l_1$. Hence, $F(M) \neq 0 \bmod l_1$ implies $F(M) \neq 0 \bmod p$, so the former condition will be a sufficient requirement to ensure that a signature for M can be constructed.

Delegation queries

– If $\mathcal{A}$ requests to interact with user 1, user 1 playing the role of one of the original signers. We assume that user n + 1 is the proxy signer. $\mathcal{A}$ creates a warrant ω, and requests user 1 to sign the warrant ω. $\mathcal{C}$ queries ω to its signing oracle $\mathcal{O}_S(sk_a^{(1)}, \cdot)$. Upon receiving an answer σ, it forwards (ω, σ) to $\mathcal{A}$ and add the warrant ω to *Warro*.

– If $\mathcal{A}$ requests to interact with user 1, user 1 playing the role of the proxy signer, the original signers are user $i (i = 2, \cdots, n+1)$. $\mathcal{A}$ outputs a warrant ω and computes the signature $\sigma_\omega^{(i)} = (\sigma_{\omega_1}^{(i)}, \sigma_{\omega_2}^{(i)})$ for warrant ω under secret key $sk_a^{(i)} (i = 2, \cdots, n+1)$. Then sends $(\omega, \sigma_\omega^{(2)}, \cdots, \sigma_\omega^{(n+1)})$ to $\mathcal{C}$. After receiving $(\omega, \sigma_\omega^{(2)}, \cdots, \sigma_\omega^{(n+1)})$, $\mathcal{C}$ check the validity by $e(g, \sigma_{\omega_1}^{(i)}) = e(pk_{ax}^{(i)}, pk_{ay}^{(i)}) e(u' \prod_{j\in\mathcal{W}} u_j, \sigma_{\omega_2}^{(i)})$, and computes $\sigma_\omega = (\prod_{i=2}^{n+1} \sigma_{\omega_1}^{(i)}, \prod_{i=2}^{n+1} \sigma_{\omega_2}^{(i)})$. If so, $\mathcal{C}$ adds the warrant ω to list *Warrp*.

Proxy multi-signature queries: $\mathcal{A}$ requests a proxy multi-signature on (ω, M), where $\omega \in Warrp$, M satisfies ω, and user 1 is the proxy signer. $\mathcal{C}$ responds to this query as follows:

– If $F(\omega) \neq 0 \bmod l_1$, $\mathcal{C}$ can just construct a signature for ω as in a signature query, and then use the *ProxyKeyGen* algorithm to create a proxy signing key sk_p on ω as follows. $\mathcal{C}$ chooses a random $r_\omega \in \mathbb{Z}_p$ and computes:

$$\sigma'_\omega = (\sigma'_{\omega_1}, \sigma'_{\omega_2}) = (g_1^{\frac{-J(\omega)}{F(\omega)}} (u' \prod_{j\in\mathcal{W}} u_j)^{r_\omega}, g_1^{\frac{-1}{F(\omega)}} g^{r_\omega}) \quad \text{and} \quad sk_p = (\sigma_{\omega_1}\sigma'_{\omega_1}, \sigma_{\omega_2}\sigma'_{\omega_2})$$

where $\sigma_\omega = (\sigma_{\omega_1}, \sigma_{\omega_2})$. Then $\mathcal{C}$ can use the *ProxyMultiSign* algorithm to create a proxy signature on M.

– If $F(\omega) = 0 \bmod l_1$, $\mathcal{C}$ will try to construct a proxy multi-signature in a similar way to the construction of a standard signature in a standard signature query. If $K(M) \neq 0 \bmod l_2$, this implies $K(M) \neq 0 \bmod p$. $\mathcal{C}$ can construct the proxy multi-signature by choosing a random $r \in \mathbb{Z}_p$ and computing:

$$p\sigma = (p\sigma_1, p\sigma_2, p\sigma_3) = (\sigma_{\omega_1} g_1^{-L(M)/K(M)} (v' \prod_{k \in \mathcal{M}} v_k)^r, \sigma_{\omega_2}, g_1^{-1/K(M)} g^r)$$
$$= (g_2^a \sigma_{\omega_1} (v' \prod_{k \in \mathcal{M}} v_k)^{\widetilde{r}}, \sigma_{\omega_2}, g^{\widetilde{r}})$$

where $\widetilde{r} = r - a/K(M)$. Then $\mathcal{C}$ sends $(\omega, p\sigma)$ to $\mathcal{A}$ as the answer.
If $K(M) = 0 \bmod l_2$, the simulator will simply abort.
Then this query (ω, M) is added to a list PMS_{qu} which is maintained by $\mathcal{C}$.

Eventually, $\mathcal{A}$ outputs a forgery. If $\mathcal{C}$ does not abort as a consequence of one of the queries above, the simulation for $\mathcal{A}$ is perfect. Since $\mathcal{A}$ has a non-negligible advantage, at least one of the following events will occur with non-negligible probability.

– $\mathbf{E}_1$: $\mathcal{A}$ outputs a forgery of the form (M^*, σ^*), where $\sigma^* = (\sigma_1^*, \sigma_2^*)$, $Verify(pk_a^{(1)}, M^*, \sigma^*) = 1$, and M^* was not queried to Oracle $\mathcal{O}_S(sk_a^{(1)}, \cdot)$. If $\mathbf{E}_1$ occurs, $\mathcal{A}$ forges user 1's signature $\sigma^* = (\sigma_1^*, \sigma_2^*)$ on M^*. If $F(M^*) \neq 0 \bmod p$ then $\mathcal{C}$ will abort. If on the other hand, $F(M^*) = 0 \bmod p$, $\mathcal{C}$ computes and outputs

$$\frac{\sigma_1^*}{(\sigma_2^*)^{J(M^*)}} = \frac{g_2^a (u' \prod_{j \in \mathcal{M}} u_j)^r}{g^{J(M^*)r}} = g_2^a = g^{ab}$$

which is the solution to the given CDH problem.

– $\mathbf{E}_2$: $\mathcal{A}$ outputs a forgery of the form $(pk_a^{(2)}, \cdots, pk_a^{(n+1)}, pk_a^{(1)}, \omega_1^*, M_1^*, p\sigma^*)$, where $p\sigma^* = (p\sigma_1^*, p\sigma_2^*, p\sigma_3^*)$, $ProxyMultiVerify(pk_a^{(2)}, \cdots, pk_a^{(n+1)}, pk_a^{(1)}, \omega_1^*, M_1^*, p\sigma^*) = 1$ and $(\omega_1^*, M_1^*) \notin PMS_{qu}$. The user 1 is the proxy signer, $\mathcal{C}$ can compute $g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}}$ because $\mathcal{C}$ has stored the secret key $sk_a^{(i)} = (g^{x_a^{(i)}}, g^{y_a^{(i)}}), i \in \{2, \cdots, n+1\}$.
If $\mathbf{E}_2$ occurs, $\mathcal{A}$ forges user 1's proxy multi-signature. If $F(\omega_1^*) \neq 0 \bmod p$ or $K(M_1^*) \neq 0 \bmod p$ then $\mathcal{C}$ will abort. If on the other hand, $F(\omega_1^*) = 0 \bmod p$ and $K(M_1^*) = 0 \bmod p$, $\mathcal{C}$ computes and outputs

$$\frac{p\sigma_1^*}{g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}} (p\sigma_2^*)^{J(\omega_1^*)} (p\sigma_3^*)^{L(M_1^*)}} = \frac{g_2^a \cdot g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}} (u' \prod_{j \in \mathcal{W}_1} u_j)^{\sum_{i=2}^{n+1} r_i + r'} (v' \prod_{k \in \mathcal{M}_1} v_k)^{r_m}}{g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}} (g^{J(\omega_1^*)})^{\sum_{i=2}^{n+1} r_i + r'} g^{L(M_1^*) r_m}} = g_2^a = g^{ab}$$

– $\mathbf{E}_3$: $\mathcal{A}$ outputs a forgery of the form $(pk_a^{(1)}, pk_a^{(2)}, \cdots, pk_a^{(n+1)}, \omega_2^*, M_2^*, p\sigma')$, where $p\sigma' = (p\sigma_1', p\sigma_2', p\sigma_3')$, $ProxyMultiVerify(pk_a^{(1)}, pk_a^{(2)}, \cdots, pk_a^{(n+1)},$

$\omega_2^*, M_2^*, p\sigma_{M_2}^*) = 1$ and $\omega_2^* \notin Warro$. The user 1 is one of the original signers, $\mathcal{C}$ can compute $g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}}$ because $\mathcal{C}$ has stored the secret key $sk_a^{(i)} = (g^{x_a^{(i)}}, g^{y_a^{(i)}}), i \in \{2, \cdots, n+1\}$.

If $\mathbf{E}_3$ occurs, $\mathcal{A}$ forges a proxy multi-signature on behalf of user 1. If $F(\omega_2^*) \neq 0 \bmod p$ or $K(M_2^*) \neq 0 \bmod p$ then $\mathcal{C}$ will abort. If on the other hand, $F(\omega_2^*) = 0 \bmod p$ and $K(M_2^*) = 0 \bmod p$, $\mathcal{C}$ computes and outputs

$$\frac{p\sigma_1'}{g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}} (p\sigma_2')^{J(\omega_2^*)} (p\sigma_3')^{L(M_1^*)}} = \frac{g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}} (g_2^a)(u' \prod_{j \in \mathcal{W}_2} u_j)^{\sum_{i=2}^{n+1} r_i + r'} (v' \prod_{k \in \mathcal{M}_2} v_k)^{r_m}}{g^{\sum_{i=2}^{n+1} x_a^{(i)} y_a^{(i)}} (g^{J(\omega_2^*)})^{\sum_{i=2}^{n+1} r_i + r'} g^{L(M_2^*) r_m}} = g_2^a = g^{ab}$$

This completes the description of the simulation. It remains to analyze the probability of $\mathcal{C}$ not aborting. For the simulation to complete without aborting, we require that the following cases happen.

- A_1: $F(M) \neq 0 \bmod l_1$ during signature queries;
- A_2: $F(\omega) \neq 0 \bmod l_1$ during delegation queries;
- A_3: $F(\omega) \neq 0 \bmod l_1$ or $K(M) \neq 0 \bmod l_2$ during proxy multi-signature queries;
- B: $\mathcal{A}$ forges user 1's signature;
- B^*: $F(M^*) = 0 \bmod p$;
- C: $\mathcal{A}$ generates a valid proxy multi-signature forgery, where user 1 is the proxy signer;
- C^*: $F(\omega_1^*) = 0 \bmod p$ and $K(M_1^*) = 0 \bmod p$;
- D: $\mathcal{A}$ generates a valid proxy multi-signature forgery, where user 1 is one of all original signers;
- D^*: $F(\omega_2^*) = 0 \bmod p$ and $K(M_2^*) = 0 \bmod p$;

We denote $A = A_1 \wedge A_2 \wedge A_3$. Thus, the successful probability of $\mathcal{C}$ is $Pr[A \wedge B \wedge B^*] + Pr[A \wedge C \wedge C^*] + Pr[A \wedge D \wedge D^*]$, $\mathcal{A}$'s advantage is $Pr[B|A] + Pr[C|A] + Pr[D|A] \geq \epsilon$, and

$$\begin{aligned}
Pr[A] &= Pr[\bigwedge_{i=1}^{q_S} F(M_i) \neq 0 \bigwedge_{i=1}^{q_{PMS}+q_D} F(\omega_i) \neq 0 \bigwedge_{j=1}^{q_{PMS}} K(M_j) \neq 0] \\
&= Pr[\bigwedge_{i=1}^{q_S} F(M_i) \neq 0 \bigwedge_{i=1}^{q_{PMS}+q_D} F(\omega_i) \neq 0] Pr[\bigwedge_{j=1}^{q_{PMS}} K(M_j) \neq 0] \\
&= (1 - Pr[\bigvee_{i=1}^{q_S} F(M_i) = 0 \bigvee_{i=1}^{q_{PMS}+q_D} F(\omega_i) = 0])(1 - Pr[\bigvee_{j=1}^{q_{PMS}} K(M_j) = 0]) \\
&\geq (1 - \sum_{i=1}^{q_S} Pr[F(M_i) = 0] - \sum_{i=1}^{q_{PMS}+q_D} Pr[F(\omega_i) = 0])(1 - \sum_{j=1}^{q_{PMS}} Pr[K(M_j) = 0]) \\
&= (1 - \frac{q_S + q_{PMS} + q_D}{l_1})(1 - \frac{q_{PMS}}{l_2})
\end{aligned}$$

$$\begin{aligned}
Pr[B^*|A \wedge B] &= Pr[F(M^*) = 0 \bmod p | A \wedge B] \\
&= Pr[F(M^*) = 0 \bmod p \wedge F(M^*) = 0 \bmod l_1 | A \wedge B] \\
&= \frac{1}{l_1(m+1)} \\
Pr[C^*|A \wedge C] &= Pr[F(\omega_1^*) = 0 \bmod p \wedge K(M_1^*) = 0 \bmod p | A \wedge C] \\
&= Pr[F(\omega_1^*) = 0 \bmod p | A \wedge C] Pr[K(M_1^*) = 0 \bmod p | A \wedge C] \\
&= \frac{1}{l_1(m+1)} \frac{1}{l_2(m+1)} \\
Pr[D^*|A \wedge D] &= Pr[F(\omega_2^*) = 0 \bmod p \wedge K(M_2^*) = 0 \bmod p | A \wedge D] \\
&= Pr[F(\omega_2^*) = 0 \bmod p | A \wedge D] Pr[K(M_2^*) = 0 \bmod p | A \wedge D] \\
&= \frac{1}{l_1(m+1)} \frac{1}{l_2(m+1)}
\end{aligned}$$

Let $l_1 = 2(q_S + q_{PMS} + q_D)$ and $l_2 = 2q_{PMS}$. Hence, we have

$$\begin{aligned}
Pr[A \wedge B \wedge B^*] &= Pr[A] Pr[B|A] Pr[B^*|A \wedge B] \\
&\geq (1 - \frac{q_S + q_{PMS} + q_D}{l_1})(1 - \frac{q_{PMS}}{l_2}) \frac{1}{l_1(m+1)} Pr[B|A] \\
&= \frac{1}{8(q_S + q_{PMS} + q_D)(m+1)} Pr[B|A] \\
Pr[A \wedge C \wedge C^*] &= Pr[A] Pr[C|A] Pr[C^*|A \wedge C] \\
&\geq (1 - \frac{q_S + q_{PMS} + q_D}{l_1})(1 - \frac{q_{PMS}}{l_2}) \frac{1}{l_1(m+1)} \frac{1}{l_2(m+1)} Pr[C|A] \\
&= \frac{1}{16 q_{PMS}(q_S + q_{PMS} + q_D)(m+1)^2} Pr[C|A] \\
Pr[A \wedge D \wedge D^*] &= Pr[A] Pr[D|A] Pr[D^*|A \wedge D] \\
&\geq (1 - \frac{q_S + q_{PMS} + q_D}{l_1})(1 - \frac{q_{PMS}}{l_2}) \frac{1}{l_1(m+1)} \frac{1}{l_2(m+1)} Pr[D^*|A] \\
&= \frac{1}{16 q_{PMS}(q_S + q_{PMS} + q_D)(m+1)^2} Pr[D^*|A]
\end{aligned}$$

and we get that

$$\begin{aligned}
Pr[\mathcal{C} \text{ succeed}] &= Pr[A \wedge B \wedge B^*] + Pr[A \wedge C \wedge C^*] + Pr[A \wedge D \wedge D^*] \\
&\geq \frac{1}{8(q_S + q_{PMS} + q_D)(m+1)} Pr[B|A] + \frac{1}{16 q_{PMS}(q_S + q_{PMS} + q_D)(m+1)^2} Pr[C|A] \\
&\quad + \frac{1}{16 q_{PMS}(q_S + q_{PMS} + q_D)(m+1)^2} Pr[D|A] \\
&\geq \frac{1}{16 q_{PMS}(q_S + q_{PMS} + q_D)(m+1)^2} (Pr[B|A] + Pr[C|A] + Pr[D|A]) \\
&\geq \frac{\epsilon}{16 q_{PMS}(q_S + q_{PMS} + q_D)(m+1)^2}
\end{aligned}$$

Algorithm $\mathcal{C}$ is successful whenever $\mathcal{A}$ is. Algorithm $\mathcal{C}$'s running time is that of $\mathcal{A}$, plus the overhead in handling $\mathcal{A}$'s q_S Signature queries, q_D Delegation queries and q_{PMS} ProxyMultiSign queries. Since there are $\mathcal{O}(m)$ multiplications and $\mathcal{O}(1)$ exponentiations in the Signature queries, $\mathcal{O}(m+n)$ multiplications and $\mathcal{O}(1)$ exponentiations in the Delegation queries and ProxyMultiSign queries, the time complexity of $\mathcal{C}$ is

$$t' = t + \mathcal{O}((m(q_S + q_{PMS} + q_D) + n(q_{PMS} + q_D))\rho + (q_S + q_{PMS} + q_D)\tau)$$

where ρ and τ are the time of a multiplication and an exponentiation in $\mathbb{G}$, respectively.

6 Conclusions

In this paper, we proposed proxy multi-signature scheme in the standard model based on signature scheme [16,15]. We showed that our scheme is unforgeable against adaptively chosen message attacks and adaptively chosen warrant attacks. To our best knowledge, this is the first proxy multi-signature scheme that can be proven secure in the standard model. The size of a proxy multi-signature is independent of the number of the original signers.

Acknowledgment

The authors would like to thank anonymous reference of ProvSec2008 for their value comments and suggestions that improve the presentation of this paper.

This work was supported by National Natural Science Foundation of China under Grants No.60473029 and the National Basic Research Program (973 Program) of China under Grants No.2007CB311201.

References

1. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
2. Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights, http://eprint.iacr.org/2003/096
3. Cao, F., Cao, Z.: Security model of proxy-multi signature schemes. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 144–152. Springer, Heidelberg (2006)
4. Huang, X., Susilo, W., Mu, Y., Wu, W.: Proxy signature without random oracles. In: Cao, J., Stojmenovic, I., Jia, X., Das, S.K. (eds.) MSN 2006. LNCS, vol. 4325, pp. 473–484. Springer, Heidelberg (2006)
5. Hsu, C., Wu, T., He, W.: New proxy multi-signature scheme. Applied Mathematics and Computation 162, 1201–1206 (2005)
6. Hsu, C., Wu, T., Wu, T.: New nonrepudiable threshold proxy signature scheme with known signers. Journal of Systems and Software 58(2,1), 119–124 (2001)

7. Ji, J., Li, D., Wang, M.: New proxy multi-signature, multi-proxy signature and multi-proxy multi-signature schemes from bilinear pairings. Chinese Journal of Computers 27(10), 1429–1435 (2004)
8. Ji, J., Li, D.: A new proxy multi-signature scheme. Journal of Computer Research and Development 41(4), 715–719 (2004)
9. Lee, J., Cheon, J., Kim, S.: An analysis of proxy signatures: Is a secure channel necessary? RSA 2003. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 68–79. Springer, Heidelberg (2003)
10. Li, X., Chen, K., Sun, L.: Certificateless signature and proxy signature schemes from bilinear pairings. Lithuanian Mathematical Journal 45(1), 76–83 (2005)
11. Lee, B., Kim, H., Kim, K.: Strong proxy signature and its applications. In: SCIS 2001, pp. 603–608 (2001)
12. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures: delegation of the power to sign messages. IEICE Transactions on Fundamentals of Electronic Communications and Computer Science E79-A(9), 1338–1354 (1996)
13. Okamoto, T., Inomata, A., Okamoto, E.: A proposal of short proxy signature using pairing. In: ITCC 2005, pp. 631–635. IEEE Computer Society, Los Alamitos (2005)
14. Okamoto, T., Tada, M., Okamoto, E.: Extended proxy signatures for smart cards. In: Zheng, Y., Mambo, M. (eds.) ISW 1999. LNCS, vol. 1729, pp. 247–258. Springer, Heidelberg (1999)
15. Paterson, K.G., Schuldt, J.C.N.: Efficient identity-based signature secure in the standard model. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006)
16. Waters, B.: Efficient Identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
17. Wang, G., Bao, F., Zhou, J., Deng, R.H.: Security analysis of some proxy signatures. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 305–319. Springer, Heidelberg (2004)
18. Wang, Q., Cao, Z.: Formal model of proxy multi-signature and a construction. Chinese Journal of Computer 29(9), 1928–1935 (2006)
19. Wang, H., Pieprzyk, J.: Efficient one-time proxy signatures. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 507–522. Springer, Heidelberg (2003)
20. Yi, L., Bai, G., Xiao, G.: Proxy multi-signature scheme: a new type of proxy signature scheme. Electronics Letters 36(6), 527–528 (2000)
21. Zhang, F., Kim, K.: Efficient ID-based blind signature and proxy signature from bilinear pairings. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 218–219. Springer, Heidelberg (2003)
22. Zhang, F., Safavi-Naini, R., Lin, C.: New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairings, http://eprint.iacr.org/2003/104

Server-Aided Verification Signatures: Definitions and New Constructions

Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering
University of Wollongong, Australia
{ww986,ymu,wsusilo,xh068}@uow.edu.au

Abstract. A server-aided verification signature scheme consists of a digital signature scheme and a server-aided verification protocol. By executing the server-aided verification protocol with the server, one can perform the verification of signatures with less computational cost compared to the original verification algorithm. This mechanism is therefore indispensable for low-power devices such as smart cards. The contributions of this paper are manyfold. Firstly, we introduce and define the existential unforgeability of server-aided verification signatures. We prove that the new notion includes the existing security requirements in server-aided verification signatures. Then, we analyze the Girault-Lefranc scheme in Asiacrypt 2005 and show that their scheme can be made secure in our model, but the computational cost is more than the claimed in the original scheme. After that, we propose the first server-aided verification BLS, which is existentially unforgeable in the random oracle model under the Bilinear Diffie-Hellman assumption. Finally, we consider the collusion and adaptive chosen message attack in server-aided verification signatures. For the first time in the literature, the security of server-aided verification signatures against such attacks is defined. We provide a concrete construction of a server-aided verification BLS secure against the collusion and chosen message attack.

Keywords: Server-aided computation, server-aided verification, BLS, ZSS, untrusted server, random oracle.

1 Introduction

Cryptographic protocols introduce extra computational costs to computer systems. It would be desirable if such cost could be reduced to a level that does not affect the overall performance of a computer system. Although applying cryptographic protocols to normal computer systems is not a problem, low power devices such as smart cards and mobile terminals require an additional care when cryptographic protocols are applied.

Techniques such as pre-computation and off-line computation have been adopted in order to improve the efficiency of the cryptographic protocols. These

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 141–155, 2008.

techniques can indeed improve the performance of cryptographic protocols. However, the computational requirement of many cryptographic systems with excellent security features still remain too heavy to low-power devices. One example is the pairing on elliptic curves. Due to its nice properties, pairing has been widely used in the recent research of cryptography, in particular in the construction of identity-based encryption and short signatures. However, pairings are computationally expensive. Reducing computational cost in pairing cryptography is a challenging task.

A promising solution is Server-aided Computation, where the client fulfils cryptographic operations with the help of a powerful server. If the server is fully-trusted, it could be done easily by allowing a secure channel between the client and the server, where the server can do anything for the client. However, in the real word, the client could be facing an untrusted server that could try to extract the secret of the client (in case of server-aided signing) or respond with a false result (in case of server-aided verification).

Many schemes about server-aided computation [1,4,5,9,10,12,14,15,16,17,20] have been found in the literature. Amongst those, the server-aided verification signature SAV-Σ attracts our attention. SAV-Σ generally consists of a digital signature scheme and a server-aided verification protocol. One can perform the signature verification with less computational cost by executing the server-aided verification protocol with the server. The notion of server-aided verification was introduced by Quisquater and De Soete [18] in order to speed up RSA verification with a small exponent. In Eurocrypt 1995, Lim and Lee proposed efficient protocols for speeding up the verification of identity proofs and signatures in discrete-logarithm-based identification schemes, based on the "randomization" of the verification equation [13]. Girault and Quisquater [8] proposed another different approach, which does not require pre-computation or randomization. Their server-aided verification protocol is computationally secure based on the hardness of a sub-problem of the initial underlying problem of the signature scheme. Hohenberger and Lysyanskaya considered the sever-aided verification under the situation that the server is made of two untrusted software packages, which are assumed not to communicate with each other [13]. Under this assumption, it allows a very light public computation task (typically one modular multiplication in the Schnorr scheme). Girault and Lefranc [7] proposed a more generalized model of server-aided verification without the assumption in [13]. A generic server-aided verification protocol for digital signatures from bilinear maps was also proposed [7]. Their protocol can be applied to signature schemes with the similar construction as the BB signature [2] and the ZSS signature [21].

Motivations & Contributions

The motivation of this paper is to define the security of server-aided verification signatures and construct new schemes that satisfy our security model. Firstly, we introduce and define the existential unforgeability of server-aided verification signatures (or, EUF-SAV-Σ for short). We prove that EUF-SAV-Σ implies the

existential unforgeability of signature scheme and the soundness of server-aided verification protocol. Our definition of EUF-SAV-Σ follows the same assumption in [7]; that is, the server does not have any valid signature of that message when it tries to prove that an invalid signature of the message is valid. Under this assumption, an existential unforgeable server-aided verification signature scheme ensures that even the server is not able to create a signature of a new message which can be proved valid by using the server-aided verification protocol.

Secondly, we consider the security of the ZSS signature [21] with the server-aided protocol proposed in [7]. Our analysis shows that the server-aided verification ZSS in [7] can be secure in our model, at the cost of more computational cost than that required in [7]. This is because the server in our model is allowed to execute the server-aided verification protocol with the verifier before proving the verifier that an invalid signature is valid. Thus, the malicious server in our model have more advantages than that considered in [7].

Thirdly, we introduce the server-aided verification to the BLS signature [3] and provide the first construction of server-aided verification BLS. We prove that our protocol is existentially unforgeable in the random oracle model, under the Bilinear Diffie-Hellman assumption.

We finally define a new type of attack for server-aided verification signatures, namely, collusion and adaptive chosen message attacks. This is the first time such attacks are considered and defined in server-aided verification signatures. Previous definitions (including EUF-SAV-Σ) are all based on the assumption that the malicious server does not have any valid signature of the message when it tries to prove an invalid signature of that message is valid. Our second model removes this assumption and allows the server to collude with the signer. A concrete construction of server-aided verification BLS secure against this attack is also proposed in this paper.

Paper Organization

The rest of this paper is organized as follows. In Section 2, we define the notion of server-aided verification signature scheme (SAV-Σ) whose existential unforgeability is defined in Section 3. We then analyze the existential unforgeability of a previously proposed SAV-Σ in Section 3. A new construction of existentially unforgeable SAV-Σ and its security analysis are given in Section 4. After that, we define the collusion and adaptive chosen message attacks in Section 5. A concrete construction of SAV-Σ secure against the new attack is also given in Section 5. Finally, we conclude this paper in Section 6.

2 Server-Aided Verification Signatures

In this section, we will review the definition of signature schemes and define server-aided verification signatures.

2.1 Syntax of a Signature Scheme Σ

A signature scheme Σ consists of the following algorithms:

Parameter-Generation: $\mathbf{ParamGen}(1^k) \rightarrow param$.
This algorithm takes as input a security parameter k and returns a string $param$ which denotes the common scheme parameters, including the description of the message space $\mathcal{M}$ and the signature space Ω, etc. $param$ is shared among all the users in the system.

Key-Generation: $\mathbf{KeyGen}(param) \rightarrow (sk, pk)$.
This algorithm takes as input the system parameter $param$ and returns a secret/public key-pair (sk, pk) for a user in the system.

Signature-Generation: $\mathbf{Sign}(param, m, sk, pk) = \sigma$.
This algorithm takes as input the system parameter $param$, the message m and the key pair (sk, pk), and returns a signature σ.

Signature-Verification: $\mathbf{Verify}(param, m, \sigma, pk) \rightarrow \{\texttt{Valid}, \texttt{Invalid}\}$.
This algorithm takes as input the system parameter $param$, the message/signatrue pair (m, σ) and the public key pk, and returns `Valid` or `Invalid`. σ is said to be a valid signature of m under pk if **Verify** outputs `Valid`. Otherwise, σ is said to be invalid.

Completeness. Any signature properly generated by **Sign** can always pass through the verification in **Verify**. That is, $\mathbf{Verify}(param, m, \mathbf{Sign}(param, m, sk, pk), pk) = \texttt{Valid}$.

Existential unforgeability of Σ. The standard notion of security for a signature scheme is called existential unforgeability under adaptive chosen message attacks [11], which is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup. The challenger $\mathcal{C}$ runs the algorithm **ParamGen** and **KeyGen** to obtain system parameter $param$ and one key pair (sk, pk). The adversary $\mathcal{A}$ is given $param$ and pk.

Queries. Proceeding adaptively, the adversary $\mathcal{A}$ can request signatures of at most q_s messages. For each sign query $m_i \in \{m_1, \cdots, m_{q_s}\}$, the challenger $\mathcal{C}$ returns $\sigma_i = \mathbf{Sign}(param, m_i, sk, pk)$ as response.

Output. Eventually, the adversary $\mathcal{A}$ outputs a pair (m^*, σ^*) and wins the game if:

1. $m^* \notin \{m_1, \cdots, m_{q_s}\}$; and
2. $\mathbf{Verify}(param, m^*, \sigma^*, pk) = \texttt{Valid}$.

We define $\Sigma\text{-}\mathsf{Adv}_{\mathcal{A}}$ to be the probability that the adversary $\mathcal{A}$ wins the above game, taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 1. *A forger $\mathcal{A}$ is said to (t, q_s, ε)-break a signature scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_s signature queries, and $\Sigma\text{-}\mathsf{Adv}_{\mathcal{A}}$ is at least ε. A signature scheme is (t, q_s, ε)-existentially unforgeable against adaptive chosen message attacks if there exists no forger that (t, q_s, ε)-breaks it.*

2.2 Syntax of a Server-Aided Verification Signature Scheme SAV-Σ

A server-aided verification signature scheme SAV-Σ consists of six algorithms: **ParamGen**, **KeyGen**, **Sign**, **Verify**, **SA-Verifier-Setup**, and **SA-Verify**. The first four algorithms are the same as those in an ordinary signature scheme Σ defined in Section 2.1, and the last two are defined as follows:

Server-Aided-Verifier-Setup: **SA-Verifier-Setup**$(param) \rightarrow$ `VString`.
This algorithm takes as input the system parameter $param$ and returns the bit string `VString`, which contains the information that can be pre-computed by the verifier. Note that `VString` might be the same as $param$ if no pre-computation is required.

Server-Aided Verification: **SA-Verify**$(\texttt{Server}^{(param)}, \texttt{Verifier}^{(m,\sigma,pk,\texttt{VString})}) \rightarrow \{\texttt{Valid}, \texttt{Invalid}\}$.

SA-Verify is an interactive protocol between `Server` and `Verifier`, who only has a limited computational ability and is not able to perform all computations in **Verify** alone. Given the message/signatrue pair (m, σ), as well as the public key pk and the inner information `VString`, `Verifier` checks the validity of σ with the help of `Server` by running **SA-Verify**. **SA-Verify** returns `Valid` if `Server` can convince `Verifier` that σ is valid. Otherwise, σ is said to be invalid.

Completeness. There are two types of completeness in SAV-Σ:

1. **Completeness of** Σ. Any signature properly generated by **Sign** can always pass through the verification in **Verify**. That is,

$$\textbf{Verify}(param, m, \textbf{Sign}(param, m, sk, pk), pk) = \texttt{Valid}.$$

2. **Completeness of SA-Verify.** An honest server can correctly convince the verifier about the validness (or, invalidness) of a signature. That is,

$$\begin{aligned} &\textbf{SA-Verify}(\texttt{Server}^{(param)}, \texttt{Verifier}^{(m,\sigma,pk,\texttt{VString})}) \\ &= \textbf{Verify}(param, m, \sigma, pk). \end{aligned}$$

2.3 Computational-Saving in SAV-Σ

Computational-Saving, probably, is the most obvious property that can distinguish a server-aided verification signature scheme SAV-Σ from an ordinary signature scheme Σ. This property enables the verifier in SAV-Σ to check the validness of signatures in a more computational efficient way than in Σ. This property is formally defined below.

Definition 2 (Computational-Saving). *Let* Φ-**Verify** *and* Φ-**SA-Verify** *denote the verifier's computational cost in* **Verify** *and* **SA-Verify**, *respectively. A server-aided verification signature scheme* SAV-Σ *is said to be* **Computational-Saving** *if* Φ-**SA-Verify** *is strictly less than* Φ-**Verify**, *i.e.,* Φ-**SA-Verify** $<$ Φ-**Verify**.

3 Existentially Unforgeable SAV-Σ

It is clear that the security of SAV-Σ must include two security notions: existential unforgeability of Σ (EUF-Σ) and the soundness of **SA-Verify** (Soundness-**SA-Verify**). The former is the same as that in Definition 1, while the latter is a new notion and only appears in the scenario of SAV-Σ. As usual, the notion soundness requires that the server should not be able to use **SA-Verify** to convince the verifier that an invalid signature is valid. The formal definition of the soundness depends on the assumption about the server. Below we will give the first security model of SAV-Σ under the same assumption in [7]. We will define another model under different assumptions in Section 5.

3.1 Definition of Existential Unforgeability of SAV-Σ

Our first model is following the assumption in [7], namely, the server does not know the valid signature of the message when it tries to use **SA-Verify** to convince the verifier that an invalid signature of that message is valid. Under this assumption, it is not necessary to consider EUF-Σ and Soundness-**SA-Verify** separately. Instead, we will give a unified notion, called the *existential unforgeability of* SAV-Σ (or, EUF-SAV-Σ for short), which implies EUF-Σ and Soundness-**SA-Verify**.

Briefly speaking, EUF-SAV-Σ requires that the adversary should not be (computationally) capable of producing a signature of a new message which can be proved as `Valid` by **SA-Verify**, even the adversary acts as `Server`. A formal game-based definition is as the following.

Setup. The challenger $\mathcal{C}$ runs the algorithm **ParamGen**, **KeyGen** and **SA-Verifier-Setup** to obtain system parameter $param$, one key pair (sk, pk) and `VString`. The adversary $\mathcal{A}$ is given $param$ and pk.

Queries. The adversary $\mathcal{A}$ can make the following queries:

- **Signature Queries.** Proceeding adaptively, the adversary $\mathcal{A}$ can request signatures of at most q_s messages. For each sign query $m_i \in \{m_1, \cdots, m_{q_s}\}$, the challenger $\mathcal{C}$ returns $\sigma_i = \textbf{Sign}(param, m_i, sk, pk)$ as response.
- **Server-Aided Verification Queries.** Proceeding adaptively, the adversary $\mathcal{A}$ can make at most q_v server-aided verification queries. For each query (m, σ), the challenger $\mathcal{C}$ responds by executing **SA-Verify** with the adversary $\mathcal{A}$, where the adversary $\mathcal{A}$ acts as `Server` and the challenger $\mathcal{C}$ acts as `Verifier`. At the end of each execution, the challenger returns the output of **SA-Verify** to the adversary $\mathcal{A}$.

Output. Eventually, the adversary $\mathcal{A}$ outputs a pair (m^*, σ^*) and wins the game if:

1. $m^* \notin \{m_1, \cdots, m_{q_s}\}$; and
2. **SA-Verify**$(\mathcal{A}^{(param, \texttt{InnerInfo})}, \mathcal{C}^{(m^*, \sigma^*, pk, \texttt{VString})}) = \texttt{Valid}$, where `InnerInfo` refers to the inner information of $\mathcal{A}$ (e.g., the random element) in the generation of σ^*.

We define SAV-Σ-Adv$_{\mathcal{A}}$ to be the probability that the adversary $\mathcal{A}$ wins in the above game, taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 3. *A forger $\mathcal{A}$ is said to $(t, q_s, q_v, \varepsilon)$-break a SAV-$\Sigma$, if $\mathcal{A}$ runs in time at most t, makes at most q_s signature queries, q_v server-aided verification queries, and SAV-Σ-Adv$_{\mathcal{A}}$ is at least ε. A SAV-Σ is $(t, q_s, q_v, \varepsilon)$-existentially unforgeable under adaptive chosen message attacks if there exists no forger that $(t, q_s, q_v, \varepsilon)$-breaks it.*

When discussing security in the random oracle model, we add a fourth parameter q_h to denote an upper bound on the number of queries that the adversary makes to the random oracle.

Remarks on EUF-SAV-Σ. We note that in **Setup**, `VString` is not given to the adversary. This is due to the concern that `VString` might contain some private information of the verifier, which must be kept as secret in sever-aided verification signatures. We will show in Section 3.3 that the adversary defined in the above model is stronger than that in [7].

3.2 Further Observations on EUF-SAV-Σ

We will show the relationship among EUF-SAV-Σ, EUF-Σ and Soundness-**SA-Verify**.

It is self-evident that EUF-SAV-Σ guarantees Soundness-**SA-Verify**. Otherwise, if there is an adversary can prove an invalid signature is valid by **SA-Verify** with success probability ε, then it can also break the existential unforgeability of SAV-Σ with the same probability. We now prove that EUF-SAV-Σ also implies EUF-Σ.

Theorem 1. *If SAV-Σ is $(t, q_s, q_v, \varepsilon)$-existentially unforgeable, then Σ is (t, q_s, ε)-existentially unforgeable.*

Proof. Let the ordinary signature scheme Σ =(**ParamGen**, **KeyGen**, **Sign**, **Verify**), and its sever-aided verification counterpart SAV-Σ=(Σ, **SA-Verifier-Setup**, **SA-Verify**). We prove the correctness of this theorem by converting a (t, q_s, ε) forger $\Sigma_{\mathcal{A}}$ to a $(t, q_s, 0, \varepsilon)$ forger SAV-$\Sigma_{\mathcal{A}}$.

As defined in the game in Section 3.1, SAV-$\Sigma_{\mathcal{A}}$ will obtain $(param, pk)$ from its challenger of SAV-Σ. Then, SAV-$\Sigma_{\mathcal{A}}$ acts as the challenger of $\Sigma_{\mathcal{A}}$ as following.

Setup. $(param, pk)$ is given to $\Sigma_{\mathcal{A}}$.

Queries. For each signature query m_i from $\Sigma_{\mathcal{A}}$, SAV-$\Sigma_{\mathcal{A}}$ forwards m_i to its challenger as a signature query of SAV-Σ. As defined, σ_i = **Sign**$(param, m, sk, pk)$ will be returned as the answer. SAV-$\Sigma_{\mathcal{A}}$ then forwards σ_i to $\Sigma_{\mathcal{A}}$. It is clear that each signature query from $\Sigma_{\mathcal{A}}$ can be correctly answered.

Output. After making queries, $\Sigma_{\mathcal{A}}$ will output a pair (m^*, σ^*). SAV-$\Sigma_{\mathcal{A}}$ sets (m^*, σ^*) as its own output.

If $\Sigma_{\mathcal{A}}$ (t, q_s, ε)-breaks the signature scheme Σ, then m^* is not one of the signature queries and $\Pr[\mathbf{Verify}(param, m^*, \sigma^*, pk) = \texttt{Valid}] \geq \varepsilon$. Due to the completeness of SAV-Σ, if $\mathbf{Verify}(param, m^*, \sigma^*, pk) = \texttt{Valid}$, then **SA-Verify** will return `Valid` as well. Therefore, SAV-$\Sigma_{\mathcal{A}}$ wins the game with the same probability ε, without making any server-aided verification queries. This completes the proof.

3.3 Analysis of the SAV-Σ in Asiacrypt'05

In this section, we consider the existential unforgeability of the generic SAV-Σ proposed by Girault and Lefranc [7]. Their server-aided verification protocol applies to signature schemes whose verification algorithms are similar to those in ZSS [21] and BB [2] signatures. We first review some fundamental backgrounds which are related in the protocol.

Bilinear Mapping: Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two groups of prime order p and let g be a generator of $\mathbb{G}_1$. The map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear mapping if the following three conditions hold true:

- e is bilinear, i.e., $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- e is non-degenerate, i.e., $e(g, g) \neq 1_{\mathbb{G}_T}$.
- e is efficiently computable.

We say that $(\mathbb{G}_1, \mathbb{G}_T)$ are bilinear groups if there exists the bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ as above, and e, and the group action in $\mathbb{G}_1$ and $\mathbb{G}_T$ can be computed efficiently. Such groups can be built from Weil pairing or Tate pairing on elliptic curves.

The Description of SAV-ZSS [7]

1. **ParamGen:** Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, k be the system security number and g be the generator of $\mathbb{G}_1$. e denotes the bilinear map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. There is one cryptographic hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$. The system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, k, g, p, e, h)$.
2. **KeyGen:** The signer picks a random number $x \in \mathbb{Z}_p^*$ and keeps it as the secret key. The public key is set as $pk = g^x$.
3. **Sign:** For a message m to be signed, the signer uses its secret key to generate the signature $\sigma = g^{\frac{1}{h(m)+x}}$.
4. **Verify:** For a message/signature pair (m, σ), everyone can check whether $e(\sigma, g^{h(m)} \cdot pk) \stackrel{?}{=} e(g, g)$. If the equation holds, output `Valid`. Otherwise, output `Invalid`.
5. **SA-Verifier-Setup:** Given the system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, k, g, p, e, h)$, the verifier picks a random integer t in $\mathbb{Z}_p$ and computes $K_1 = e(g, g)^t$. The `VString` is (t, K_1).
6. **SA-Verify:** The verifier and the server interact with each other using the protocol described in Figure 1.

Verifier (`VString`: (t, K_1))		Server ($param$)
Input: $R = (g^{h(m)} \cdot pk)^t$	$\xrightarrow{\sigma, R}$ $\xleftarrow{K_2}$	$K_2 = e(\sigma, R)$
Output: `Valid`, if $K_1 = K_2$; `Invalid`, otherwise.		

Fig. 1. SA-Verify in SAV-ZSS [7]

Security of SAV-ZSS [7]. We now show that SAV-ZSS [7] is insecure in the model defined in Section 3.1, if the same (t, K_1) is used in each execution of **SA-Verify** described in Figure 1.

We first briefly review the security conclusion of SAV-ZSS proved in [7]:

1. A malicious server is not able to prove a verifier that an invalid signature of a message m is valid by using **SA-Verify** in Figure 1, if
2. the server does not know the ZSS signature of m and k-BCAA problem is hard (Please refer to [7] for the definition of k-BCAA).

However, the malicious server considered in [7] is not allowed to execute **SA-Verify** with the verifier, before it tries to prove the verifier that an invalid signature is valid. We believe this restriction is not reasonable as the verifier in the real world would execute **SA-Verify** with the server for several times. In the model defined in Section 3.1, we allow the adversary (acting as the server) to choose any message-signature pair, and execute **SA-Verify** with the challenger (acting as the verifier). This is analogous to the definition of existentially unforgeablity, where the forger is allowed to obtain valid signatures of messages chosen by itself. Under this model, SAV-ZSS [7] will be insecure[1] if the same (t, K_1) is used in **SA-Verify** in Figure 1. The following shows how the adversary in our model can break the existential unforgeability of SAV-ZSS [7]:

1. The adversary $\mathcal{A}$ first issues a signature query on a message m. Let the response from the challenger be σ.
2. $\mathcal{A}$ makes a sever-aided verification request (m, σ). As shown in **SA-Verify** in Figure 1, the challenger will send the adversary $R = (g^{h(m)} \cdot pk)^t$.
3. $\mathcal{A}$ computes $K_2 = e(\sigma, R)$. As σ is a valid ZSS signature of m, $K_1 = K_2 = e(g, g)^t$.
4. With the knowledge of K_1, $\mathcal{A}$ is able to prove that any invalid signature is valid if the same (t, K_1) is used in **SA-Verify**. To do that, $\mathcal{A}$ just sends K_1 to the challenger in every execution of **SA-Verify**. Thus, $\mathcal{A}$ can always win the game defined in Section 3.1.

[1] SAV-ZSS in [7] is still secure against the adversary defined in [7]. However, the adversary in [7] is weaker than the one defined in this paper.

It is clear that the above attack will not work if the verifier pre-computes $q_v + 1$ pairs (t, K_1) in SAV-ZSS [7] and the adversary is allowed to make at most q_v server-aided verification queries. This will require more storage space of the verifier. Alternatively, the verifier can choose different t, and compute $(g^{H(m)} \cdot pk)^t$ and $e(g,g)^t$ in each execution of **SA-Verify**. This however will lead to one more exponentiation in $\mathbb{G}_T$ than the computational cost of the verifier claimed in [7].

4 Existentially Unforgeable SAV-BLS

In this section we will propose a new server-aided verification signature scheme: SAV-BLS.

4.1 Complexity Assumptions

The bilinear mapping we used in our protocol is the same as that defined in Section 3.3.

Bilinear Diffie-Hellman Problem (BDH): Given (g, g^a, g^b, g^c) for some $a, b, c \in \mathbb{Z}_p^*$, compute $e(g,g)^{abc} \in \mathbb{G}_T$. An algorithm $\mathcal{A}$ has advantage ε in solving BDH on $(\mathbb{G}_1, \mathbb{G}_T)$ if

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g,g)^{abc}] \geq \varepsilon$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p^*$, the random choice of generator $g \in \mathbb{G}_1$, and the random bits of $\mathcal{A}$.

Bilinear Diffie-Hellman Assumption: The (t, ε)-BDH assumption holds on $(\mathbb{G}_1, \mathbb{G}_T)$ if no t-time adversary has advantage at least ε in solving BDH on $(\mathbb{G}_1, \mathbb{G}_T)$.

4.2 Description of Existentially Unforgeable SAV-BLS

Our scheme is based on the BLS signature [3]. The description of our protocol is as following.

1. **ParamGen:** Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, k be the system security number and g be the generator of $\mathbb{G}_1$. e denotes the bilinear map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. There is one cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. The system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, k, g, p, e, H)$.
2. **KeyGen:** The signer picks a random number $x \in \mathbb{Z}_p^*$ and keeps it as the secret key. The public key is set as $pk = g^x$.
3. **Sign:** For a message m to be signed, the signer uses its secret key to generate the signature $\sigma = H(m)^x$.
4. **Verify:** For a message/signature pair (m, σ), everyone can check whether $e(\sigma, g) \stackrel{?}{=} e(H(m), pk)$. If the equation holds, output `Valid`. Otherwise, output `Invalid`.

Verifier (`VString`: (r, R))		Server ($param$)
Input: $(m, \sigma), pk, param$	$\xrightarrow{\sigma, R}$	
	$\xleftarrow{K_1}$	$K_1 = e(\sigma, R)$
Compute: $K_2 = e(H(m), pk)^r$		
Output: `Valid`, if $K_1 = K_2$; `Invalid`, otherwise.		

Fig. 2. SA-Verify in SAV-BLS with EUF

5. **SA-Verifier-Setup:** Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g, p, e, H)$, the verifier V randomly chooses $r \in \mathbb{Z}_p$ and sets $R = g^r$. The `VString` is (r, R).
6. **SA-Verify:** The verifier V and the server S interact with each other using the protocol described in Figure 2.
   Note that R is precomputed and the verifier sends the same R to the server in server-aided verification of different message-signature pairs.

Computational-Saving. The verifier in SAV-BLS described above needs to compute one pairing, one exponentiation on $\mathbb{G}_T$, and one map-to-point hash. It is obvious that Φ- **SA-Verify**$<$ Φ-**Verify**.

Security Proof of Existentially Unforgeable SAV-BLS

Theorem 2. *The SAV-BLS signature scheme is* $(t, q_s, q_v, q_h, \varepsilon)$*-existentially unforgeable against adaptive chosen message attacks, if* $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_h + 2q_s + 2q_v + 1), \frac{\varepsilon}{\mathfrak{e} q_v (q_s + 1)})$*-BDH assumption holds on* $(\mathbb{G}_1, \mathbb{G}_T)$*. Here,* $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ *is a constant that depends on* $(\mathbb{G}_1, \mathbb{G}_T)$ *and* $\mathfrak{e}$ *is the base of natural logarithm.*

Proof. We omit the proof due to the page limitation.

Remark: One can use our method to construct a SAV-Waters' signature [19].

5 SAV-Σ Secure against Collusion and Adaptive Chosen Message Attacks

One assumption in previous definitions (including Definition 3) is that the server does not know any valid signature of the message m, when it tries to use **SA-Verify** to convince the verifier that an invalid signature of m is valid. In this section, we address our attention on the security of SAV-Σ against the collusion between the server and the signer, and propose a server-aided verification protocol secure against this attack.

5.1 Definition of the Security of SAV-Σ against Collusion and Adaptive Chosen Message Attacks

If we allow the server and the signer to collude, the server will have valid signatures of any messages. Thus, it is impossible to give a unified security notion to capture both EUF-Σ and Soundness-**SA-Verify** simultaneously. With this in mind, we now define the soundness of the server-aided verification protocol **SA-Verify** against collusion and adaptive chosen message attacks. In the game, the adversary is given the secret key of the signer.

Setup. The challenger $\mathcal{C}$ runs the algorithm **ParamGen**, **KeyGen** and **SA-Verifier-Setup** to obtain system parameter $param$, one key pair (sk, pk) and `VString`. The adversary $\mathcal{A}$ is given $param$ and (sk, pk).

Queries. The adversary $\mathcal{A}$ only needs to make **Server-Aided Verification Queries**. Proceeding adaptively, the adversary $\mathcal{A}$ can make at most q_v such queries. The challenger $\mathcal{C}$ responds each query in the same way as described in Definition 3.

Output. The adversary $\mathcal{A}$ will output a message m^*. We denote Ω_{m^*} as the set of valid signatures of m^*. The challenger $\mathcal{C}$ chooses a random element σ^* in $\Omega \setminus \Omega_{m^*}$ as the response to the adversary $\mathcal{A}$. That is, σ^* is a random invalid signature of m^*. We say $\mathcal{A}$ wins the game if **SA-Verify**$(\mathcal{A}, \mathcal{C}^{(m^*, \sigma^*, pk, \texttt{VString})}) = \texttt{Valid}$.

We define Soundness-**SA-Verify**-Adv$_{\mathcal{A}}$ to be the probability that the adversary $\mathcal{A}$ wins in the above game, taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 4. *An adversary $\mathcal{A}$ is said to (t, q_v, ε)-break the soundness of* **SA-Verify** *in a* SAV-Σ *if $\mathcal{A}$ runs in time at most t, makes at most q_v server-aided verification queries, and* Soundness-**SA-Verify**-Adv$_{\mathcal{A}}$ *is at least ε. The* **SA-Verify** *in a* SAV-Σ *is (t, q_v, ε)-sound against collusion and adaptive chosen message attacks if there exists no adversary that (t, q_v, ε)-breaks it.*

Definition 5. SAV-Σ *is $(t, q_s, q_v, \varepsilon)$-secure against collusion and adaptive chosen message attacks if Σ is (t, q_s, ε)-existentially unforgeable against adaptive chosen message attacks and its server-aided verification protocol* **SA-Verify** *is (t, q_v, ε)-sound against collusion and adaptive chosen message attacks.*

5.2 SAV-BLS Secure against Collusion and Adaptive Chosen Message Attacks

We now give a server-aided verification protocol for the BLS signature [3], which is secure against the collusion and adaptive chosen message attacks.

1. **ParamGen**, **KeyGen**, **Sign**, **Verify:** these algorithms are the same as defined in Section 4.2.
2. **SA-Verifier-Setup:** Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g, p, e, H)$, the verifier V computes $K_1 = e(g, g)$. The `VString` is K_1.
3. **SA-Verify:** The verifier V and the server S interact with each other using the protocol described in Figure 3.

Verifier (`VString`: K_1)		Server (param)
Input: $(m, \sigma), pk, param$		
Compute: $r \in_R \mathbb{Z}_p$, $\sigma' = \sigma \cdot g^r$	$\xrightarrow{m,\sigma',pk}$	
		$K_2 = e(\sigma', g)$
	$\xleftarrow{K_2,K_3}$	$K_3 = e(H(m), pk)$
Output: `Valid`, if $K_2 = K_3 \cdot K_1^r$ `Invalid`, otherwise.		

Fig. 3. SA-Verify in SAV-BLS with Soundness

Computational-Saving. The verifier in SAV-BLS described above only needs to compute one multiplication on $\mathbb{G}_1$, one (fixed-base) exponentiation on $\mathbb{G}_1$, one multiplication on $\mathbb{G}_T$ and one (fixed-based) exponentiation on $\mathbb{G}_T$. In particular, there is no pairing or map-to-point operation. Thus, Φ- **SA-Verify**$<\Phi$-**Verify**.

Security Proof of SAV-BLS Against Collusion and Chosen Message Attacks

We only need to show that the server-aided verification protocol in Figure 3 is sound against collusion and adaptive chosen message attacks.

Theorem 3. *The server-aided verification protocol described in Figure 3 is* $(t, q_v, \frac{1}{p-1})$*-sound against collusion and adaptive chosen message attacks.*

Proof. We omit the proof due to the page limitation.

Remark: One can use our method to construct a SAV-Waters' signature [19]. We can easily improve the model described above by allowing the adversary to choose the signature σ^* in $\Omega \setminus \Omega_{m^*}$ and the adversary wins the game if it can prove the invalid signature to be valid. Unfortunately, until now we cannot find a corresponding SAV signature scheme which is secure against such adversary.

6 Conclusion

This paper focused on server-aided verification signatures. We formally defined the existential unforgeability of server-aided verification signatures which is proved to include the existing security requirements in the server-aided verification signatures. Then, we analyzed the Girault-Lefranc scheme from Asiacrypt 2005 and proposed the first server-aided verification BLS, which was existentially unforgeable in the random oracle model under the Bilinear Diffie-Hellman assumption. Finally, for the first time, we introduced and defined the security

of server-aided verification signatures against the collusion and adaptive chosen message attack. A concrete construction of server-aided verification BLS secure against such attack was also presented.

References

1. Anderson, R.J.: An Attack on Server-Assisted Authentication Protocols. Electronic Letters 28(15), 1473 (1992)
2. Boneh, D., Boyen, X.: Short Signatures without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 382–400. Springer, Heidelberg (2004)
3. Boneh, D., Lynn, G., Shacham, H.: Short Signature from The Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
4. Burns, J., Mitchell, C.J.: Parameter Selection for Server-Aided RSA Computation Schemes. IEEE Transaction on Computers 43, 147–163 (1994)
5. Beguin, P., Quisquater, J.-J.: Fast Server-Aided RSA Signatures Secure Against Active Attacks. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 57–69. Springer, Heidelberg (1995)
6. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
7. Girault, M., Lefranc, D.: Server-Aided Verification: Theory and Practice. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 605–623. Springer, Heidelberg (2005)
8. Girault, M., Quisquater, J.J.: GQ + GPS = new ideas + new protocols. In: Eurocrypt 2002 - Rump Session (2002)
9. Girault, M., Paillès, J.C.: On-line/Off-line RSA-like. In: International Workshop on Coding and Cryptography (2003)
10. Gennaro, R., Rabin, T., Krawczyk, H.: RSA-Based Undeniable Signatrues. Journal of Cryptology 13(4), 397–416 (2000)
11. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 17(2), 281–308 (1988)
12. Kawamura, S., Shimbo, A.: Fast Server-Aided Secret Computation Protocols for Modular Exponentiation. IEEE Journal on selected areas communications 11 (1993)
13. Lim, C.H., Lee, P.J.: Security and Performance of Server-Aided RSA Computation Protocols. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 70–83. Springer, Heidelberg (1995)
14. Matsumoto, T., Imai, H., Laih, C.-S., Yen, S.-M.: On Verifiable Implicit Asking Protocols for RSA Computation. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 296–307. Springer, Heidelberg (1993)
15. Matsumoto, T., Kato, K., Imai, H.: Speeding Up Secret Computation with Insecure Auxiliary Devices. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 497–506. Springer, Heidelberg (1990)
16. Nguyen, P., Stern, J.: The Béguin-Quisquater Server-Aided RSA Protocol from Crypto95 is not Secure. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 372–379. Springer, Heidelberg (1998)

17. Pfitamann, B., Waidner, M.: Attacks on Protocols for Sever-Aided RSA Computation. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 153–162. Springer, Heidelberg (1993)
18. Quisquater, J.-J., De Soete, M.: Speeding Up Smart Card RSA Computation with Insecure Coprosessors. In: Proceedings of Smart Cards 2000 , pp. 191–197 (1989)
19. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
20. Yen, S.-M., Laih, C.-S.: More about the Active Attack on the Server-Aided Secret Computation Protocol. Electronic Letters, 2250 (1992)
21. Zhang, F., Safavi-Naini, R., Susilo, W.: An Efficient Signature Scheme from Bilinear Pairing and its Applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)

On Proofs of Security for DAA Schemes*

Liqun Chen[1], Paul Morrissey[2], and Nigel P. Smart[2]

[1] Hewlett-Packard Laboratories,
Filton Road,
Stoke Gifford,
Bristol, BS34 8QZ,
United Kingdom
liqun.chen@hp.com
[2] Computer Science Department,
Woodland Road,
University of Bristol,
Bristol, BS8 1UB,
United Kingdom
{paulm,nigel}@cs.bris.ac.uk

Abstract. Direct anonymous attestation (DAA) is a mechanism for a remote user to provide a verifier with some assurance it is using software and/or hardware from trusted sets of software and/or hardware respectively. In addition, the user is able to control if and when a verifier is able to link two signatures: to determine whether or not they were produced by the same platform. The verifier is never able to tell which which particular platform produced a given signature or pair of signatures.

We first address a problem with the proof of security for the original DAA scheme of Brickell, Camenisch and Chen. In particular, we construct an adversary that can tell if its in a simulation or not. We then provide the necessary changes to the simulator such that the adversary can no longer do this and prove this fact, hence repairing the proof.

Our main contribution is a security analysis of the Chen, Morrissey and Smart (CMS) DAA scheme. This scheme uses asymmetric bilinear pairings and was proposed without a proof of security. We use the well established simulation based security model of Brickell, Camenisch and Chen and also use a similar proof technique to theirs. We prove the CMS scheme is secure in the random oracle model relative to the bilinear Lysyanskaya, Rivest, Sahai and Wolf (LRSW) assumption, the hardness of discrete logarithms in the groups used and collision resistance of the hash functions used in the scheme.

1 Introduction

The huge growth in the number and type of services available on the internet, such as online shopping, social networking and online secure backup/storage,

* The second and third author would like to thank EPSRC, eCrypt and HP Labs for partially supporting the work in this paper.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 156–175, 2008.

has led to a growth in the number and type of security threats posed to any given service provider. As a result, service providers may require a given user to provide some assurance that they can be trusted. One way of achieving this is to use trusted computing.

The particular use of trusted computing we look at in this paper is direct anonymous attestation (DAA). This provides a mechanism for a remote user to anonymously assure some verifier that it is using software and hardware from specified sets of trusted software and hardware respectively. In addition DAA allows for user controlled linkability of signatures. This means the user controls when a verifier can tell if a given pair of signatures were produced by the same platform. Under no circumstances can a verifier tell which platform produced a given signature.

Related Work. Brickell, Camenisch and Chen proposed the first DAA scheme [3] which uses the Camenisch – Lysyanskaya (CL) signature scheme [7]. In [3] this scheme and a security model for DAA schemes were proposed and the scheme proved secure under the decisional Diffie–Hellman and strong RSA assumptions. The DAA security model proposed in [3] was a simulation based one using real and ideal system executions, and requiring that no computationally bounded environment can distinguish if it is in the real system with a real adversary or an ideal system with a simulator. Camenisch and Groth [6] later improved the performance of the scheme of [3] by introducing randomisation into the CL signature scheme. Backes, Maffei and Unruh [1] used a mechanised technique to analyse the scheme of [3]. Other works have looked at different security aspects of DAA schemes [4,12], efficiency improvements [4,5,10] and different uses [2,11].

The most efficient DAA scheme to date is that of Chen, Morrissey and Smart [9]. This is based on the scheme of [4,5] but uses asymmetric bilinear maps as opposed to symmetric ones. This move to the asymmetric setting allows for a number of efficiency improvements and simplifications. In particular the scheme of [9] requires extremely little work from the trusted platform module (TPM). The TPM only has to perform operations in one standard sized elliptic curve group and does not have to perform any expensive operations such as pairings. The scheme of [9] was presented without a proof of security.

Our Contribution. There are two main contributions in this paper. The first concerns a problem with the proof of security of the original DAA scheme [3]. The second is a proof of security of the CMS DAA scheme [9]. We describe the details of each separately.

Fixing the Proof of [3]. The first DAA scheme of Brickell, Camenisch and Chen [3] uses an RSA based method to compute signatures of knowledge. We first take a look at the proof of security for this scheme. Proofs such as this are prone to error; they require a high level of detail and present many opportunities for potential problems. In particular we find this proof contains a minor flaw which allows for the construction of an adversary that can tell it is run in a simulation with high probability. Following this we provide a different way of constructing the simulator that avoids this problem. Finally we provide a proof that this

adversary cannot tell if it is in a simulation or not. In short we repair the proof security of the original DAA scheme of [3].

A Proof of Security of the CMS DAA Scheme. The CMS DAA scheme of [9] is an optimised DAA scheme developed from the scheme of [4,5] and obtained by moving to the asymmetric pairings setting. The main contribution of this paper is the first proof of security for the CMS scheme.

Our proof uses the real/ideal system model and the simulation paradigm as in [3]. The reason we choose this model is to ensure we completely capture the full behaviour of the system. Other security models for DAA schemes do exist such as that of [4] but it is not yet clear how this relates to the more established security model of [3]. Our proof is in some ways similar to that of [3] but with a few important differences. Most notably, in the simulation of the Sign protocol the simulator can only partially forge signatures, and hence embed the hard problem, when a Join protocol has been run with an honest issuer. A similar case arises in the Verify algorithm simulation. One important consequence of the simplicity of the CMS scheme is that our proof of security is also much simpler than that of the original scheme [3]. This makes our proof less likely to contain any minor errors in the details. Specifically, we prove the CMS scheme is secure, in the random oracle model, relative to the bilinear LRSW assumption, the hardness of discrete logarithms in one of the groups chosen for the scheme and the collision resistance of the hash functions used in the scheme.

Paper Overview. Some notational conventions and preliminaries are given in Section 2. We give an overview of the execution and security model for DAA schemes from [3] in Section 3. In Section 4 we look at the proof of security for the DAA scheme of [3]. Finally, we give a security analysis of the CMS scheme [9] in Section 5.

2 Notation and Preliminaries

In this section we describe any notational conventions we use throughout the paper. We then briefly review the bilinear LRSW assumption and the definition of a DAA scheme.

Notation. If S is any set then we denote the action of sampling an element from S uniformly at random and assigning the result to the variable x as $x \xleftarrow{\$} S$. If A is any algorithm then we denote the action of obtaining x by running A on inputs $y_1, \ldots, y_n$, with access to oracles $\mathcal{O}_1(\cdot), \mathcal{O}_2(\cdot), \ldots, \mathcal{O}_m(\cdot)$, as $x \leftarrow A^{\mathcal{O}_1(\cdot),\mathcal{O}_2(\cdot),\ldots,\mathcal{O}_m(\cdot)}(y_1, \ldots, y_n)$. We write $\{0,1\}^t$ for the set of binary strings of length t and $\{0,1\}^*$ for the set of binary strings of arbitrary length and use the abbreviations d.p.t. for deterministic polynomial time and p.p.t. for probabilistic polynomial time.

Throughout we let $\mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle$ and $\mathbb{G}_T$ be groups of large prime exponent $q \approx 2^t$ for security parameter t. We further assume that the discrete logarithm in these groups is believed to be hard. The groups $\mathbb{G}_1, \mathbb{G}_2$ will be

written additively and the group $\mathbb{G}_T$ multiplicatively. If $\mathbb{G}$ is some group then we use the notation $\mathbb{G}^\times$ to mean the non-identity elements of $\mathbb{G}$. If $\mathbb{G}$ is some ring or field we take $\mathbb{G}^\times$ to mean the non-zero elements of $\mathbb{G}$ (non-identity elements for the addition operation) and $\mathbb{G}^*$ to be the multiplicative subgroup of $\mathbb{G}$.

Definition 1 (Pairing). *A* pairing (or bilinear map) *is a map* $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ *such that:*

(1) *The map* $\hat{t}$ *is* bilinear. *This means that* $\forall\ P, P' \in \mathbb{G}_1$ *and* $\forall\ Q, Q' \in \mathbb{G}_2$ *that*
- $\hat{t}(P + P', Q) = \hat{t}(P, Q) \cdot \hat{t}(P', Q) \in \mathbb{G}_T$.
- $\hat{t}(P, Q + Q') = \hat{t}(P, Q) \cdot \hat{t}(P, Q') \in \mathbb{G}_T$.

(2) *The map* $\hat{t}$ *is* non-degenerate. *This means that*
- $\forall\ P \in \mathbb{G}_1^\times\ \exists\ Q \in \mathbb{G}_2$ *such that* $\hat{t}(P, Q) \neq 1_{\mathbb{G}_T} \in \mathbb{G}_T$.
- $\forall\ Q \in \mathbb{G}_2^\times\ \exists\ P \in \mathbb{G}_1$ *such that* $\hat{t}(P, Q) \neq 1_{\mathbb{G}_T} \in \mathbb{G}_T$.

(3) *The map* $\hat{e}$ *is* computable *i.e. there exist some polynomial time algorithm to compute* $\hat{t}(P, Q) \in \mathbb{G}_T$ *for all* $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$.

Bilinear LRSW Assumption. Here we briefly review the bilinear LRSW assumption first proposed as a variant of the LRSW assumption in [8]. For this we define the oracle $\mathcal{O}_{X,Y}(\cdot)$ which on input $f \in \mathbb{Z}_q$ outputs a triple $(A, y \cdot A, (x + fxy)A)$ where $A \xleftarrow{e} \mathbb{G}_1, X = x \cdot P_1$ and $Y = y \cdot P_2$. We then have the following definition.

Definition 2 (bilinear LRSW Advantage). *We define the bilinear LRSW advantage* $\mathbf{Adv}_{\mathcal{A}}^{bLRSW}(t)$ *of an adversary* $\mathcal{A}$ *against* $(\mathbb{G}_1, \mathbb{G}_2, P_1, P_2, q, \hat{t})$ *as*

$$\Pr\left[\begin{array}{c} x, y \xleftarrow{e} \mathbb{Z}_q;\ X \leftarrow xP_1, Y \leftarrow yP_2;\ (f, A, B, C) \leftarrow \mathcal{A}^{\mathcal{O}_{X,Y}(\cdot)}(\mathbb{G}_1, \mathbb{G}_2, P_1, P_2, q, \hat{t}) \\ \wedge \left(f \notin \mathcal{Q},\ f \in \mathbb{Z}_q^\times,\ A \in \mathbb{G}_1,\ B = y \cdot A,\ C = (x + fxy) \cdot A\right) \end{array}\right]$$

where $\mathcal{Q}$ *is the set of queries that* $\mathcal{A}$ *made to* $\mathcal{O}_{X,Y}(\cdot)$ *and* $q \approx 2^t$.

We then say a tuple $(\mathbb{G}_1, \mathbb{G}_2, P_1, P_2, q, \hat{t})$ satisfies the bilinear LRSW assumption if for any p.p.t. adversary $\mathcal{A}$ its advantage $\mathbf{Adv}_{\mathcal{A}}^{bLRSW}(t)$ is negligible in t.

General DAA Schemes. We refer to the entities in a DAA scheme as players. We first review the description of each type of player as given in [5]. We then review the formal definition of a DAA scheme from [9].

The DAA Players. We only give a brief description of each type of DAA player here. For a more detailed description the reader is referred to [5]. A general DAA scheme has a set of players that consists of:

- A set of users $\mathcal{U} = \mathcal{M} \times \mathcal{H}$ where each $u_i = (m_i, h_i) \in \mathcal{U}$ consists of
  - A TPM module $m_i \in \mathcal{M}$ with endorsement key ek_i and seed $\mathsf{DaaSeed}_i$ where $\mathcal{M}$ is the set of TPM modules in the scheme.
  - A host $h_i \in \mathcal{H}$ which has a counter cnt_i, a set of commitments $\{\mathsf{comm}\}_i$ and a set of credentials $\{\mathsf{cre}\}_i$ and where $\mathcal{H}$ is the set of hosts in the scheme.

- A set of issuers $\mathcal{I}$. Each $I_k \in \mathcal{I}$ has a public and private key pair $(\mathsf{ipk}_k, \mathsf{isk}_k)$ and long term fixed public parameter K_k (for example the long term public key of this issuer). Each $I_k \in \mathcal{I}$ also maintains a local list of rogue TPM internal secret values $\mathsf{RogueList}(I_k)$.
- A set of verifiers $\mathcal{V}$. Each $V_j \in \mathcal{V}$ maintains a set of basenames $\{\mathsf{bsn}\}_j$ and a local list of rogue TPM internal secret values $\mathsf{RogueList}(V_j)$. Each $V_j \in \mathcal{V}$ may also optionally maintain a list of received signature and message pairs (this may be used to to trade memory for computational efficiency when linking signatures).

We assume that initially the sets $\{\mathsf{comm}\}_i$ and $\{\mathsf{cre}\}_i$ are empty for all $U_i \in \mathcal{U}$, the lists $\mathsf{RogueList}(I_k)$ are empty for all $I_k \in \mathcal{I}$, and the sets $\{\mathsf{bsn}\}_j$ and lists $\mathsf{RogueList}(V_j)$ are empty for all $V_j \in \mathcal{V}$.

The set $\{\mathsf{bsn}\}_j$ is used to achieve user controlled linkability of signatures. A given player may be corrupted or honest. We take corruption to mean the adversary knows the full internal state of that player and as a result, once a player is corrupted, the adversary takes over the role of that player. We say a user $u_i = (m_i, h_i)$ is fully corrupted if both h_i and m_i are corrupted and partially corrupted if only h_i is corrupted. We assume it is easier to break into a host than a TPM module and as a result do not model the case of an honest host with a corrupt TPM.

Formal Definition of a DAA Scheme. We now recall the formal definition of a DAA scheme from [9].

Definition 3 (Daa Scheme). Formally, we define a Daa scheme to be a tuple of protocols and algorithms $\mathsf{Daa} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link}, \mathsf{RogueTag})$ where:

- $\mathsf{Setup}(1^t)$ is a p.p.t. system setup algorithm. For security parameter t this outputs a set of system parameters par containing all of the issuer public keys ipk_k and the various parameter spaces. This algorithm also sets up and securely distributes each of the issuer secret keys isk_k.
- $\mathsf{Join}(u_i, I_k)$ is a 3 party protocol between a TPM, a host and an issuer. In a correct initial run with honest players the host should obtain an additional valid commitment and an additional valid credential. In correct subsequent runs an existing commitment and an existing credential should be replaced by newer ones.
- $\mathsf{Sign}(u_i, \mathsf{msg})$ is a 2 party protocol between a TPM and a host used to generate a signature of knowledge on msg. In a correct run with honest players the signature of knowledge will be constructed according to some basename for some specified verifier that may or may not allow the signature to be linked to other signatures with this same verifier.
- $\mathsf{Verify}(\sigma, \mathsf{msg})$ is a d.p.t. or p.p.t. verification algorithm that allows a given verifier to verify the signature of knowledge σ of a credential on msg intended for that verifier with a specific basename. The verification process will involve checking the signature against $\mathsf{RogueList}(V_j)$. This algorithm returns either *accept* or *reject*. If the verifier is maintaining a list of signatures received then it will add any correctly verified signatures to this list.

- $\mathsf{Link}(\sigma_0, \sigma_1)$ is a d.p.t. linking algorithm that returns either **linked**, **unlinked** or $\perp$. The algorithm should return $\perp$ if either signature was produced with a rogue key, return **linked** if both are valid signatures and the user who produced them wanted these to be linkable to each other, and return **unlinked** otherwise.
- $\mathsf{RogueTag}(f, \sigma)$ is a d.p.t. rogue tagging algorithm that returns **true** if σ is a valid signature that was produced using the TPM secret value f and **false** otherwise.

For correctness we require that if

- A user $u_i \in \mathcal{U}$ engages in a run of Join with I_k resulting in u_i obtaining a commitment comm on a TPM secret value f and a credential cre corresponding to f.
- The user u_i then creates two signatures σ_b on two messages msg_b for $b \in \{0, 1\}$ intended for verifier $V_j \in \mathcal{V}$ with basename bsn.
- The secret TPM value f used to compute these is such that $f \notin \mathsf{RogueList}(V_j)$.

Then $\mathsf{Verify}(\sigma_0, \mathsf{msg}_0) = \mathsf{Verify}(\sigma_1, \mathsf{msg}_1) = \mathit{accept}$ and if $\mathsf{bsn} \neq \perp$ then $\mathsf{Link}(\sigma_0, \sigma_1)$ = **linked**.

3 DAA Execution and Security Model

In this section we give a detailed description of the real/ideal system model [3] for DAA schemes.

Real System Execution for a DAA Scheme. For the real system we model a set of players consisting of $n_{\mathcal{U}}$ users $u_i \in \mathcal{U}$ each with a host h_i and corresponding TPM module m_i, a set of $n_{\mathcal{V}}$ verifiers $V_j \in \mathcal{V}$ and a set of $n_{\mathcal{I}}$ issuers $I_k \in \mathcal{I}$. The honest players in the system receive inputs from and send outputs to the environment Env. Honest players also run cryptographic protocols with each other and perform cryptographic computations themselves according to the description of the DAA scheme. We model an adversary $\mathcal{A}$ as a p.p.t. algorithm that controls a number of corrupt players. Since the adversary controls the set of corrupt players then it will arbitrarily interact with other players and Env.

Ideal System Execution for a DAA Scheme. In the ideal system we have the same set of players as in the real system. In addition there exists some trusted third party $\mathcal{T}$. The main difference to the real system is players engage in protocol runs and perform cryptographic computations by passing inputs to $\mathcal{T}$ and receiving outputs from $\mathcal{T}$ rather than perform these themselves according to the scheme.

The trusted third party $\mathcal{T}$ provides the functionality we want from a secure DAA scheme by maintaining a number of lists and making decisions based on these lists. These lists include a list $\mathsf{CorruptTPM}$ of endorsement key and counter pairs, a list of signatures issued $\mathsf{Signatures}$, a list of members $\mathsf{Members}$ and a rogue list $\mathsf{RogueList}$.

We assume whenever a TPM is corrupted it tells $\mathcal{T}$ by sending its index i (which we use as an identifier) to $\mathcal{T}$ who adds this to CorruptTPM. The entries of Signatures have the form $(\sigma, \mathsf{msg}, i, \mathsf{cnt}, V_j, \mathsf{bsn})$ and each is intended to mean $\mathcal{T}$ computed a signature σ on a message msg on behalf of the user with identifier i and using internal secret value corresponding to cnt intended for the verifier V_j and that is linkable to all other signatures with V_j that have the same basename $\mathsf{bsn} \in \{0,1\}^* \cup \ \perp$ (providing $\mathsf{bsn} \neq \perp$). The entries of RogueList will contain TPM identifier and counter pairs. The list Members also contains identifier and counter pairs and is essentially a list of those TPM's that $\mathcal{T}$ has issued credentials to on behalf of issuers. Intuitively, the list CorruptTPM is a list of TPM's for which the adversary has complete control and hence knows all values of internal secrets for each value of counter. On the other hand, the list RogueList will be a list of TPM and internal secret pairs that have been compromised. The adversary may only know a single value of internal secret for a rogue TPM. To this end, $\mathcal{T}$ uses CorruptTPM to decide if a given identifier and counter pair should be added to RogueList or not. The trusted third party then performs the following functionality on behalf of players:

Setup. Any corrupted TPM modules inform $\mathcal{T}$ of this by sending their identifier i to $\mathcal{T}$ who adds this to CorruptTPM.

Join. The host h_i contacts $\mathcal{T}$ with the identifier i and counter cnt and requests to become a member with respect to cnt and issuer $I_k \in \mathcal{I}$. Next $\mathcal{T}$ sends cnt to m_i and asks if it wants to join with respect to cnt. The module m_i informs $\mathcal{T}$ of its decision and if it wants to join $\mathcal{T}$ consults the list CorruptTPM and if $i \in$ CorruptTPM adds an entry (i, cnt) to RogueList, sends to I_k the pair (i, cnt), and informs I_k if $(i, \mathsf{cnt}) \in$ RogueList or not. The issuer I_k then makes a decision as to whether (i, cnt) can become a member or not and informs $\mathcal{T}$ of this. If I_k decides that (i, cnt) can become a member then $\mathcal{T}$ adds (i, cnt) to Members. Finally $\mathcal{T}$ informs h_i of the decision.

Sign/Verify. A given host requests to sign a message msg for a verifier V_j with basename bsn using a given pair (i, cnt) by sending to $\mathcal{T}$ a tuple $(\mathsf{msg}, i, \mathsf{cnt}, V_j)$.

- If $(i, \mathsf{cnt}) \notin$ Members then $\mathcal{T}$ denies the request and replies to h_i with $\perp$.
- Else if $(i, \mathsf{cnt}) \in$ Members then $\mathcal{T}$ forwards msg and cnt to the relevant TPM and asks if it wants to sign with respect to cnt. If so then $\mathcal{T}$ asks h_i for a basename with which to produce the signature.
- If $(i, \mathsf{cnt}) \in$ RogueList then $\mathcal{T}$ informs V_j that msg has been signed by a rogue TPM.
- If $\mathsf{bsn} = \perp$ then $\mathcal{T}$ first generates a random σ and next adds a new entry $(\sigma, \mathsf{msg}, i, \mathsf{cnt}, V_j, \perp)$ to Signatures. Next $\mathcal{T}$ informs V_j that msg was signed with respect to bsn.
- Else if $\mathsf{bsn} \neq \perp$ then $\mathcal{T}$ generates a random σ, adds $(\sigma, \mathsf{msg}, i, \mathsf{cnt}, V_j, \mathsf{bsn})$ to Signatures then informs V_j that msg was signed with respect to bsn.

Link. A given verifier V_j requests a linkage decision from $\mathcal{T}$ by submitting two signatures (σ_0, σ_1) to $\mathcal{T}$. If there exists two entries on Signatures of the form

$(\sigma_b, \mathsf{msg}_b, i, \mathsf{cnt}, V_j, \mathsf{bsn}_b)$ for some i, cnt_i and $b \in \{0,1\}$ and $\mathsf{bsn}_0 = \mathsf{bsn}_1 \neq \perp$ then $\mathcal{T}$ returns **linked** and otherwise returns **unlinked** to V_j.

RogueTag. When the rogue tagging oracle $\mathcal{O}_{\mathcal{R}}$ wishes to add an entry to RogueList it submits a pair (i, cnt) to $\mathcal{T}$. Then $\mathcal{T}$ informs $\mathcal{O}_{\mathcal{R}}$ as to whether $i \in$ CorruptTPM or not. If there is such an entry and $(i, \mathsf{cnt}) \in$ Members then $\mathcal{O}_{\mathcal{R}}$ adds (i, cnt) to RogueList and otherwise does not.

Note that the signature and verification functionality provided by $\mathcal{T}$ is given together. The reason why is we want that each signature is only presented for verification once, to the intended verifier, and that no signatures are produced and then not presented for verification. Also note that during the Join and Sign operations the trusted third party asks a module if it wants to join or sign a message respectively. We assume that if a module asked does not belong to the host that is asking then the module refuses and otherwise agrees.

Securely Implementing Functionality. Intuitively, we say a system is secure if its behaviour is computationally indistinguishable from a an ideal DAA system. The formal definition of a secure implementation of a DAA scheme is then as follows.

Definition 4 (Secure Implementation). *A given* DAA *scheme* Daa *is a secure implementation if for every computationally bounded environment* Env *and every* p.p.t. *adversary* $\mathcal{A}$ *there exists some simulator* $\mathcal{S}$*, that controls the same set of players in an ideal–system as* $\mathcal{A}$ *does in a real–system, such that* Env *cannot distinguish whether it is run in the real–system and interacts with* $\mathcal{A}$ *or whether it is run in an ideal–system and interacts with* $\mathcal{S}$*.*

We assume the trusted third party $\mathcal{T}$ in an ideal system is in some sense "invisible" to the environment since otherwise the environment can trivially distinguish real from ideal systems. To this end any protocols for which message are passed through $\mathcal{T}$ appear as if they are passed directly between players and any computations performed by $\mathcal{T}$ on behalf of players appear as if they are performed by the players themselves.

4 A Note on the Proof of the Scheme of [3]

In this section we take a detailed look at the proof of security for the original DAA scheme as given in [3]. The notation we use to describe parts of this proof and of the scheme are taken directly from [3]. Due to space constraints we do not repeat the notational conventions or describe the scheme but refer the reader to [3] for a full description.

In [3] a proof of security is presented for the proposed DAA scheme. In this proof the simulator $\mathcal{S}$, acting in the ideal system, is given black box access to the real system adversary $\mathcal{A}$. The simulator attempts to forge all signatures constructed using an honest TPM for which any interaction with $\mathcal{A}$ is required. There are two situations where a signature is partially forged by $\mathcal{S}$: one case of the Sign protocol and one case of the Verify protocol. In this section we first describe

the necessary details of how these signatures are constructed in a real protocol run. We then construct an adversary that can distinguish if it is in a simulation or not with overwhelming probability based on the way these signature forgeries are constructed by the simulator. Finally we present an alternative method of forging such partial signatures that avoids this problem and prove that the adversary can no longer distinguish if it is in a simulation or not when this method is used.

Overview of Signature Construction in [3]. We only describe the details of how the values $\tilde{T}_{1t}, \tilde{T}_1$ and $\hat{T}_1$ are computed during signing and verification with a TPM m_i and a host h_i. This is because these parameters are the only important ones in our arguments. Informally, during a correct run of the Sign protocol of [3] m_i first computes a value $\tilde{T}_{1t}$ and passes this to h_i. Next h_i computes the value of $\tilde{T}_1$ from $\tilde{T}_{1t}$ and uses this as part of the signature. Upon verification the verifier will compute a value $\hat{T}_1$ and, if the hash function H used is collision resistant, with overwhelming probability the signature will verify if and only if $\tilde{T}_1 = \hat{T}_1$. We now describe the details. The values $\tilde{T}_{1t}, \tilde{T}_1$ and $\hat{T}_1$ are computed as

$$\begin{aligned}
\tilde{T}_{1t} &= R_0^{r_{f_0}} R_1^{r_{f_1}} S^{r_v} \pmod n \\
\tilde{T}_1 &= \tilde{T}_{1t} T_1^{r_e} h^{-r_{ew}} \pmod n \\
\hat{T}_1 &= Z^{-c} T_1^{s_e + c2^{l_e-1}} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} h^{-s_{ew}} \pmod n.
\end{aligned}$$

We use the following relations

$$\begin{aligned}
&T_1 = Ah^w \pmod n && s_e = r_e + c(e - 2^{le-1}) \\
&s_i = r_i + ci && \text{for } i \in \{ew, f_0, f_1, s_v\} \\
&T_1^{s_e + c2^{l_e-1}} = T_1^{r_e} T_1^{ce} \pmod n && Z = A^e R_0^{f_0} R_1^{f_1} S^{s_v} \pmod n.
\end{aligned}$$

Then, if the signature is computed correctly, $\tilde{T}_1 = \hat{T}_1$ follows from

$$\begin{aligned}
\hat{T}_1 &= Z^{-c} T_1^{s_e + c2^{l_e-1}} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} h^{-s_{ew}} \pmod n \\
&= (A^e R_0^{f_0} R_1^{f_1} S^v)^{-c} T_1^{r_e} T_1^{ce} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} h^{-s_{ew}} \pmod n \\
&= A^{-ec} T_1^{r_e} (Ah^w)^{ce} R_0^{s_{f_0} - cf_0} R_1^{s_{f_1} - cf_1} s^{s_v - cv} h^{-s_{ew}} \pmod n \\
&= T_1^{r_e} h^{cew} R_0^{r_{f_0}} R_1^{r_{f_1}} S^{r_v} h^{-s_{ew}} \pmod n \\
&= T_1^{r_e} \tilde{T}_{1t} h^{-r_{ew}} = \tilde{T}_1 \pmod n.
\end{aligned}$$

Constructing an Adversary that can Distinguish. We first recall the necessary details of the description of $\mathcal{S}$ in the proof of [3] during the simulation of a Sign protocol run. In the case we are concerned with $\mathcal{S}$ is simulating the behaviour of an honest TPM to $\mathcal{A}$ who is performing computations on behalf of a corrupt host. The simulator and adversary are aiming to work together to produce a signature of knowledge as part of the signing protocol. We are particularly concerned with the computation of the value $\tilde{T}_1$. In step 3 (c) of the description of $\mathcal{S}$ it computes $\tilde{T}_{1t} = Z^{-c} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} \pmod n$ and passes this to $\mathcal{A}$ along with some other parameters.

At this point $\mathcal{A}$ may either compute what it thinks is a correct signature for this particular run or not. If $\mathcal{A}$ computes what it thinks is a correct signature then it will compute $\tilde{T}_1 = \tilde{T}_{1t} T_1^{r_e} h^{-r_{ew}} \pmod n$. During verification an honest

verifier will compute $\hat{T}_1 = Z^{-c} T_1^{s_e + c2^{l_e - 1}} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} h^{-s_{ew}} \pmod{n}$. The signature provided only verifies correctly if $\hat{T}_1 = \tilde{T}_1$. However, due to the way $\mathcal{S}$ constructs $\tilde{T}_{1t}$, if $\mathcal{A}$ computes what it thinks is a correct signature from $\tilde{T}_{1t}$, then $\tilde{T}_1 = \hat{T}_1$ with only negligible probability in the size of n and hence the signature will not verify with overwhelming probability. The following computations, all modulo n, demonstrate why this is the case

$$\begin{aligned} \hat{T}_1 &= Z^{-c} T_1^{r_e} T_1^{ce} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} h^{-s_{ew}} & &= \tilde{T}_{1t} T_1^{r_e} T_1^{ce} h^{-s_{ew}} \\ &= \tilde{T}_{1t} T_1^{r_e} (Ah^w)^{ce} h^{-s_{ew}} & &= \tilde{T}_{1t} T_1^{r_e} A^{ce} h^{-r_e w} \\ &= \tilde{T}_1 A^{ce} & &= \tilde{T}_1 \Leftrightarrow A^{ce} = 1 \end{aligned}$$

To construct an adversary that can tell it is run in a simulation we simply have it compute $\hat{T}_1$ and compare this to the value $\tilde{T}_1$. If $\hat{T}_1 \neq \tilde{T}_1 \pmod{n}$ and $A^{ce} \neq 1 \pmod{n}$ then $\mathcal{A}$ knows it is in a simulation. The adversary can compute A^{ce} and $\hat{T}_1$ since all values required to compute these are known to both host and TPM. Furthermore, if $\hat{T}_1 = \tilde{T}_1 A^{ce} \pmod{n}$ then the adversary knows the exact details of how the simulator is trying to forge its part of the signature. This check does not give the adversary any information if $A^{ce} = 1 \pmod{n}$ which will happen with very small probability for sufficiently large n.

Repairing the Proof. To correct the proof of [3] we alter the way $\mathcal{S}$ computes $\tilde{T}_{1t}$ such that, when $\mathcal{A}$ is computing a valid signature using this, $\tilde{T}_1 = \hat{T}_1$. This can be done by $\mathcal{S}$ computing $\tilde{T}_{1t} = R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} Z^{-c} A^{ec} \pmod{n}$. The simulator will always know A and e since the credential sent to the host will be encrypted under ek_i and will be passed to the simulator for decryption. We then have the following Lemma.

Lemma 1. *If the value $\tilde{T}_{1t}$ is computed by $\mathcal{S}$ as $\tilde{T}_{1t} = R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} Z^{-c} A^{ec} \pmod{n}$ in the proof of security of the scheme of [3] then no polynomially bounded adversary can tell it is run in a simulation.*

Proof. Firstly, when $\mathcal{S}$ computes $\tilde{T}_{1t}$ as $\tilde{T}_{1t} = R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_v} Z^{-c} A^{ec} \pmod{n}$ its easy to see that the value $\tilde{T}_1$ that the adversary computes from this, when it is computing what it thinks is a correct signature, is such that $\tilde{T}_1 = \hat{T}_1 \pmod{n}$. Secondly, since all of the values chosen by $\mathcal{S}$ that are different from those in a real run of the protocol are chosen in the same way as in [3], the same argument from [3] that the adversary cannot distinguish based on these applies. □

5 Security Analysis of the CMS Scheme

In this section we give a security analysis of the CMS DAA scheme [9]. The notation we use to describe the scheme is the same as that of [9].

We briefly Recall some of the main points of the CMS scheme. The Setup algorithm sets up all system parameters. In particular, this selects the groups for the scheme $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, the pairing to be used $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$ and the the hash functions in the scheme H_1, H_2, H_3 and H_4. During the Join protocol the TPM computes a Schorr signature of knowledge on the secret value f to

produce the commitment and the issuer computes a bilinear CL signature on this commitment to produce the credential. This credential is then checked by the host. For the Sign protocol the host and TPM work together to produce a signature of knowledge of the discrete logarithm f to some base in $\mathbb{G}_1$. This is done using only a single mask value as opposed to two mask values in [5]. The Verify algorithm checks correctness of a signature of knowledge and checks this against the list of rogue TPM values. The Link algorithm verifies correctness of a pair of signatures and checks if they were computed using the same basename or not and the RogueTag checks if a given signature was produced with a given value of TPM secret f. For a full description of the CMS scheme we refer to the reader to [9].

We are now able to state the main result of this paper regarding the security of the CMS DAA scheme.

Theorem 1. *The CMS* DAA *scheme is secure in the random oracle model under the bilinear LRSW assumption in* $(\mathbb{G}_1, \mathbb{G}_2, P_1, P_2, \hat{t})$*, the hardness of discrete logarithms in* $\mathbb{G}_1$ *and if the hash functions* H_3 *and* H_4 *are collision resistant.*

Proof. We first give an overview of the proof. To prove the theorem we need to show that for every adversary $\mathcal{A}$ in the real system there exists a simulator $\mathcal{S}$ in the ideal system such that the environment cannot tell which system it is in and such that the adversary cannot tell it is in a simulation. We assume a given adversary $\mathcal{A}$ exists and $\mathcal{S}$ has access to $\mathcal{A}$ as a black box. The simulator then has to simulate the behaviour of the honest players to $\mathcal{A}$ such that $\mathcal{A}$ cannot tell it is run in a simulation. The simulator allows $\mathcal{A}$ to perform computations and run protocols for the corrupted players and controls the random oracles. The simulator also interacts with $\mathcal{T}$. In this case the simulator has to perfectly simulate the behaviour of the corrupted players to $\mathcal{T}$ in the ideal system such that the environment cannot tell if it is run in a real or ideal system.

To construct the proof we first describe the operation of the simulator $\mathcal{S}$ and its interaction with the environment Env and the adversary $\mathcal{A}$ by describing how $\mathcal{S}$ handles the communications according to combinations of player corruptions. We use the notation (ihM) to describe the communication and computations between a corrupted initiator (lower case i), a corrupt host (lower case h) and an honest TPM (upper case M). This corresponds to a partially corrupted user communicating with a corrupt issuer. We then prove that the simulator will not abort outputting "failure X" for $X \in \{1, 2, 3, 4, 5\}$. Each such failure event corresponds to the adversary being able to solve some hard problem that is embedded into the interaction between the adversary and simulator. Finally we prove $\mathcal{A}$ cannot tell it is run in a simulation and the environment cannot distinguish if it is run in a real system with $\mathcal{A}$ or in an ideal system with $\mathcal{S}$ constructed from $\mathcal{A}$.

Answering Random Oracle Queries. To handle random oracle queries $\mathcal{S}$ maintains a number of, initially empty, lists HList_i for $i \in \{1, 2, 3, 4\}$. Each corresponds to a list of input and output pairs used in the simulation of one of the 4 random oracles.

Algorithm 1. GetRand Algorithm.

Input: A tuple (f, bsn)
Output: A value $r' \in \mathbb{Z}_q$

if $\exists\ (r', f, F, \mathsf{bsn}) \in \mathsf{HList}_2$ **then**
  return r';
else if $\exists\ (r', \perp, F, \mathsf{bsn}) \in \mathsf{HList}_2$ and $F = f \cdot P_1$ **then**
  replace $(r', \perp, F, \mathsf{bsn})$ with (r', f, F, bsn);
  return r';
else if $\exists\ (\perp, \perp, F, \perp) \in \mathsf{HList}_2$ and $F = f \cdot P_1$ **then**
  choose $r' \xleftarrow{e} \mathbb{Z}_q$ then replace $(\perp, \perp, F, \perp)$ with (r', f, F, bsn);
  return r';
else
  $r' \xleftarrow{e} \mathbb{Z}_q$; $F \leftarrow f \cdot P_1$;
  add (r', f, F, bsn) to HList_2;
  return r';

Queries to the oracles H_1, H_3 and H_4 are answered in the obvious manner. Particular care must be taken with queries to H_2 since in some cases $\mathcal{S}$ will simulate protocols with an unknown value of f yet still need to be able to answer queries in a consistent manner.

The entries of HList_2 are tuples of the form (r', f, F, bsn) where $r' \in \mathbb{Z}_q, f \in \mathbb{Z}_q, F \in \mathbb{G}_1$ and $\mathsf{bsn} \in \{0,1\}^*$. We define algorithm 1 to aid in the answering H_2 queries.

When the adversary makes a query $H_2(f \| \mathsf{bsn})$ the simulator makes a query **GetRand**(f, bsn) and replies with the returned value of r'.

Simulation of the Setup Algorithm. During the Setup the behaviour of $\mathcal{S}$ is according to the corruption state of the issuer I_k running the algorithm. The simulator performs an ideal system setup to $\mathcal{T}$ and a real system setup to $\mathcal{A}$. For the ideal system setup $\mathcal{S}$ informs $\mathcal{T}$ which modules are corrupted by sending ek_i to $\mathcal{T}$ for each one controlled by $\mathcal{S}$. For the real system simulation to $\mathcal{A}$ the simulator performs the following:

(1) First $\mathcal{S}$ simulates the generation of the system wide parameters by executing steps 1 to 3 of the Setup algorithm and storing/passing the parameters to $\mathcal{A}$.
(2) For each corrupted issuer I_k (case (i)) $\mathcal{A}$ will pass the values ipk_k to $\mathcal{S}$ after performing step 4 of Setup. For each honest m_i the setup is simulated to $\mathcal{A}$ by $\mathcal{S}$ selecting values for each $\mathsf{DaaSeed}_i$ and setting $\mathsf{cnt}_i = 0$. For each corrupted user $u_i = (m_i, h_i)$ the simulator runs the ideal system Join protocol with I_k using $\mathsf{cnt}_i = 0$. This is to ensure that these users actually run the Join protocol at least once: the simulator would not be aware of whether they did or did not otherwise. None of the honest parties in the ideal system will note any of this so this does not give the environment a method of distinguishing the system.
(3) For each honest issuer I_k (case (I)) the simulator also has to simulate the generation of public and private key pairs to $\mathcal{A}$. This is done as in step 4 of

the Setup algorithm and the relevant values stored and passed to $\mathcal{A}$. For each corrupted m_i the simulator has to simulate the ideal system setup to $\mathcal{T}$ by setting $\mathsf{cnt}_i = 0$ in the ideal system.

Simulation of the Join Protocol. We are able to distinguish 4 cases of corruption states for players for which we need to describe the behaviour of $\mathcal{S}$.

Case (iHM). The issuer is corrupted but the user is not. The simulator has to play the role of the host and module to $\mathcal{A}$ and has to play the role of the issuer to $\mathcal{T}$. The simulator first receives a pair (i, cnt) and information as to whether the module is tagged as a rogue for that pair from $\mathcal{T}$. The simulator then does the following:

- Begins a simulated run of the real Join protocol by playing the role of $u_i = (m_i, h_i)$ with $\mathcal{A}$ playing the role of the corrupt initiator. Since the user is honest $\mathcal{S}$ will know $\mathsf{DaaSeed}_i$ and will correctly compute and store a value for $\mathsf{comm} = (F, c, s)$.
- Eventually $\mathcal{S}$ should receive a credential (if $\mathcal{A}$ decides to continue) from $\mathcal{A}$ and inform $\mathcal{T}$ that it allows (i, cnt) to be added to Members. Otherwise $\mathcal{S}$ informs $\mathcal{T}$ that (i, cnt_i) cannot become a member.

Case (Ihm). Here $\mathcal{S}$ has to play the part of the honest issuer I_k to $\mathcal{A}$ who controls a fully corrupted user $u_i = (m_i, h_i)$ and the part of u_i user to $\mathcal{T}$. The simulator will receive a value for a commitment comm from $\mathcal{A}$. The simulator then does the following:

- Engages in a real system Join protocol with $\mathcal{A}$ up until the point where $\mathcal{S}$ has to send a credential to $\mathcal{A}$. If this run of the real system Join protocol aborts before this point then $\mathcal{S}$ needs to do nothing more.
- If comm has been used previously for user u_i then $\mathcal{S}$ makes a request to $\mathcal{T}$ to become a member with the identifier i for u_i and the counter value cnt for the run where comm was previously used.
- If comm has not been previously used then $\mathcal{S}$ makes a request to $\mathcal{T}$ to become a member for i and $\mathsf{cnt} + 1$.
- $\mathcal{T}$ will then interact with the ideal system issuer I_k that $\mathcal{S}$ is simulating to $\mathcal{A}$ and eventually returns a decision to $\mathcal{S}$ as to whether i can become a member with respect to the counter value submitted.
- If the pair submitted to $\mathcal{T}$ is allowed to become a member then $\mathcal{S}$ simulates the generation of a credential (it can do this since I_k is honest and hence $\mathcal{S}$ knows the value of isk_k) and passes this to $\mathcal{A}$. Otherwise $\mathcal{S}$ informs $\mathcal{A}$ it cannot become a member.

Case (IhM). In this case the simulator forges the production of a commitment for an unknown value of f using its power over the random oracles then correctly simulates a credential and forges a value of E for unknown f using the issuer secret key values. The simulator plays the role of m_i and I_k to $\mathcal{A}$ and the role of h_i to $\mathcal{T}$.

- The simulator first receives a request to become a member from $\mathcal{A}$ then, acting as the honest issuer, generates $n_I \xleftarrow{\$} \{0,1\}^t$, assigns $\mathsf{comm}_{\mathrm{req}} \leftarrow (n_I, \mathsf{K}_k, X, Y)$ and returns this to $\mathcal{A}$.
- The adversary will then pass a value of $\mathsf{comm}'_{\mathrm{req}}$ back to $\mathcal{S}$ as the response from the issuer (note this may be different to the $\mathsf{comm}_{\mathrm{req}}$ chosen by $\mathcal{S}$ in the previous step). The simulator then performs any necessary checks on this and if it fails then $\mathcal{S}$ informs $\mathcal{A}$ it does not want to join in this case. Then, acting as h_i in the ideal system, makes a query to $\mathcal{T}$ to join with some counter value and upon receiving a request to join from $\mathcal{T}$, acting as m_i, informs $\mathcal{T}$ it does not want to join in this case.
- The simulator then has to choose a value of F for which it does not know the corresponding value f. It does this by choosing $F \xleftarrow{\$} \mathbb{G}_1$ until $\nexists\, (*, f, F, *) \in \mathsf{HList}_2$ then adds an entry $(\perp, \perp, F, \perp)$ to HList_2.
- The simulator then forges the production of a commitment by choosing $s, c \xleftarrow{\$} \mathbb{Z}_q$, computing $U = sP_1 - cF$ and $\mathsf{str} = 1 \| X \| Y \| n_I$ then patching the random oracle for H_1 such that $c = H_1(\mathsf{str} \| F \| U)$. Finally $\mathsf{comm} \leftarrow (F, c, s)$ and comm is passed to $\mathcal{A}$.
- Next $\mathcal{S}$, acting as I_k, should receive some commitment comm' from $\mathcal{A}$ (again $\mathcal{A}$ may have modified the one passed to it). If comm' fails any of the checks the issuer performs on it then $\mathcal{S}$ replies to inform $\mathcal{A}$ that it cannot become a member. Otherwise $\mathcal{S}$ increments cnt_i and makes a request to $\mathcal{T}$ to become a member with respect to cnt_i and i (acting as the corrupt host in the ideal system).
- If $\mathcal{S}$ receives a decision that it cannot become a member with respect to i and cnt_i then $\mathcal{S}$, acting as I_k, informs $\mathcal{A}$ that it cannot become a member. Otherwise $\mathcal{S}$ computes the credential by selecting $r \xleftarrow{\$} \mathbb{Z}_q$, assigning $A \leftarrow rP_1$, $B \leftarrow yA$ and $C \leftarrow (xA + rxyF)$ and $\mathsf{cre} = (A, B, C)$. Then computes $\mathcal{E} \leftarrow E_{\mathsf{ek}_i}(\mathsf{cre})$ and passes this to $\mathcal{A}$.
- The simulator, acting as m_i, should then receive a value $\mathcal{E}'$ from $\mathcal{A}$ (which may be modified from $\mathcal{E}$). It then performs any checks on this and if these fail aborts the run and otherwise forges E using $E \leftarrow ryF$ and returns this to $\mathcal{A}$.

We note that we have to forge a commitment that validates correctly since the adversary may decide to check the validity of this before passing it to the issuer and if we don't correctly forge it then the adversary will be able to notice it is not run in the real system.

Case (ihM). In this case $\mathcal{S}$ does not forge the production of a commitment or credential: it is unable to since the issuer is controlled by $\mathcal{A}$. Instead $\mathcal{S}$ simulates a correct run of the Join protocol with $\mathcal{A}$. The simulator, acting as m_i, will first receive a value of n_I from $\mathcal{A}$, acting as h_i, as part of a commitment request. The simulator then does the following:

- Correctly simulates the TPM side of the join protocol to produce a commitment. During this a value (r', f, F, bsn) will be added to HList_2.

- Sends a value for identifier and counter to $\mathcal{T}$ as a joining request. The response from $\mathcal{T}$ will be sent back to $\mathcal{S}$.
- The simulator then responds to $\mathcal{A}$ with the value of commitment computed and continues to simulate the protocol as m_i. If the adversary sends a value of $\mathcal{E}$ that correctly decrypts to a valid credential to the simulator (where $\mathcal{A}$ is playing the role of h_i) then $\mathcal{S}$ responds to $\mathcal{T}$ to inform it that the endorsement key and counter pair can join and if not then informs $\mathcal{T}$ it cannot.
- The simulator then computes the value for E and responds to the adversary with this.

Simulation of Sign. For the simulation of the Sign protocol we only need to consider the case (hM). This is because when both h_i and m_i are corrupted the adversary will control all communication and computations without the aid of $\mathcal{S}$ and in the case where neither is corrupted then h_i and m_i will communicate directly with $\mathcal{T}$. Also, as mentioned earlier, we do not consider a corrupt TPM with an honest host. The behaviour of $\mathcal{S}$ in this case depends upon the corruption state of the issuer that the Join protocol (that the current value of f corresponds to) was run with.

The simulator behaves as follows:

- The simulator will receive as input a value bsn from $\mathcal{A}$ (acting as the host). If no Join protocol has been successfully finished by m_i with the issuer I_k (i.e. with the counter value corresponding to I_k in the ideal system) then the simulator rejects the request.
- Otherwise the behaviour of $\mathcal{S}$ depends upon whether the Join protocol was run with an honest or corrupt issuer. If the issuer was honest (case (IhM) in the Join protocol) then $\mathcal{S}$ forges the signature using the (forged) value of E computed during this Join protocol as follows:
  (a) The simulator has to assign a value of r' to the value of bsn received. If $\mathsf{bsn} = \perp$ then $\mathcal{S}$ assigns $r' \xleftarrow{c} \mathbb{Z}_q$. Else if $\mathsf{bsn} \neq \perp$ then $\mathcal{S}$ first retrieves the value of F for any of the Join protocols run with u_i and I_k (these will all be the same). The simulator then checks HList_2 and if there is an entry $(*, f, F, *)$ then $\mathcal{A}$ is able to compute discrete logs in $\mathbb{G}_1$ so $\mathcal{S}$ aborts outputting "failure 1". Otherwise there will be at least one entry $(r^*, \perp, F, \mathsf{bsn}^*)$ on HList_2. If $\mathsf{bsn} = \mathsf{bsn}^*$ then $\mathcal{S}$ uses $r' = r^*$ and otherwise selects $r' \xleftarrow{c} \mathbb{Z}_q$ and adds $(r', \perp, F, \mathsf{bsn})$ to HList_2.
  (b) Next $\mathcal{S}$ selects $c, s \xleftarrow{c} \mathbb{Z}_q$, computes $D = sB - cE$ (where B corresponds to the credential for E) and sends r' and $D' = r' \cdot D$ to $\mathcal{A}$.
  (c) The simulator will then receive a value of c' back from $\mathcal{A}$ and selects $n_T \xleftarrow{c} \{0,1\}^t$ then patches the random oracle for H_4 such that $c = H_4(c' \| n_T \| \mathsf{msg})$.
  (d) Next $\mathcal{S}$ has to simulate the part of h_i to $\mathcal{T}$. It does this by making a request to $\mathcal{T}$ to sign msg with respect to cnt_i. If $\mathcal{T}$ informs $\mathcal{S}$ that m_i is ready to sign and requests a basename then $\mathcal{S}$ postpones the answer to $\mathcal{T}$ ($\mathcal{S}$ will answer this call only when it has seen a signature from $\mathcal{A}$ during the verification protocol) and replies to $\mathcal{A}$, as m_i, with (c, s, n_T).

Otherwise if the Join protocol was run with a corrupt issuer then the $\mathcal{S}$ simulates the production of a valid signature as follows:

(a) In this case the run of the Join protocol would have required $\mathcal{S}$ to choose a value of f and the simulator simulates the TPM part of the signing protocol correctly right up to the point where the value of c is computed using H_4.

(b) Next $\mathcal{S}$ has to simulate the corrupted host part of the signing protocol with $\mathcal{T}$. It does this by making a request to $\mathcal{T}$, as h_i, to sign msg with respect to cnt. If $\mathcal{T}$ informs $\mathcal{S}$ that m_i is ready to sign and requests a basename then $\mathcal{S}$ postpones the answer to $\mathcal{T}$ ($\mathcal{S}$ will answer this call only when it has seen a signature from $\mathcal{A}$ during the verification protocol) and replies to $\mathcal{A}$, as m_i, with (c, s, n_T).

Simulation of Verify. For this we have two main cases. The first is an honest verifier, controlled by $\mathcal{S}$, that receives a signature from a corrupted host which we denote case (V). The second is an adversarially controlled verifier where the simulator wants to verify a signature for an honest host which we denote case (v).

Case (v). Since V_j is corrupt u_i must be honest. The simulator will receive a notification from $\mathcal{T}$ that a user signed msg with respect to basename bsn. This will be towards the end of a run of the signing protocol of the completely honest user with $\mathcal{T}$. The simulator $\mathcal{S}$ then has to simulate the production of a signature in the real system to $\mathcal{A}$ on behalf of u_i that correctly verifies. To do this the value of credential that the signature is on must correctly verify. This can only be forged for the case where the user obtains this credential from an honest issuer (under the control of $\mathcal{S}$) since otherwise, the adversary can simply check if $\rho'_c = \hat{t}(A' + E', X)$ or not. Hence, once again, in the case where we have an honest issuer we are able to forge the signature and in the case of an adversarially controlled issuer we have to correctly simulate the signature.

For the case where the signature was produced with an honest issuer the simulator forges the signature as follows.

(1) First $\mathcal{S}$, acting as h_i, has to select or compute a validly produced, but forged, credential $\mathsf{cre} = (A, B, C)$ and obtain the corresponding value E. If no credential has been obtained by the simulator for the correct issuer then the simulator first simulates a Join protocol for the case of (IHM) by using its powers over the random oracles and selecting a value of F for unknown f in a similar way to the Join forgery of the case (IhM). Otherwise $\mathcal{S}$ selects a random credential issued by the specific issuer $\mathsf{cre} = (A, B, C)$ and obtains the corresponding E.

(2) Next $\mathcal{S}$ chooses a value of r'. If $\mathsf{bsn} = \perp$ then $\mathcal{S}$ sets $r' \xleftarrow{c} \mathbb{Z}_q$. If $\mathsf{bsn} \neq \perp$ then $\mathcal{S}$ first retrieves the value of F for any of the Join protocols run with m_i, h_i and I_k (these will all be the same). The simulator then checks HList_2 and if there is an entry $(*, f, F, *)$ then the adversary is able to compute discrete logs in $\mathbb{G}_1$ so $\mathcal{S}$ aborts outputting "failure 1". Otherwise there will be an (at least one) entry $(r^*, \perp, F, \mathsf{bsn}^*)$. If $\mathsf{bsn} = \mathsf{bsn}^*$ then $\mathcal{S}$ uses $r' = r^*$ and otherwise

selects $r' \xleftarrow{\$} \mathbb{Z}_q$ and adds an entry $(r', \perp, F, \mathsf{bsn})$. The value of $r', \mathsf{bsn}, \mathsf{cre}$ and F should now all correctly relate to each other for an unknown value of f.

(3) The simulator then sets $D = sB - cE$, computes $A' = r'A, B' = r'B, C' = r'C, E' = rE, D' = r'D, \rho'_a = \hat{t}(A', X), \rho'_b = \hat{t}(B', X)$ and $\rho'_c = \hat{t}(C', P_2)$. The simulator selects $s, c, c' \xleftarrow{\$} \mathbb{Z}_q$ and computes $\tau = (\rho'_b)^s \cdot (\rho'_c/\rho'_a)^{-c}$.

(4) Next $\mathcal{S}$ fixes the relevant random oracles by choosing $n_V \xleftarrow{\$} \{0,1\}^t$ then fixing H_3 such that $c' = H_3(\mathsf{ipk}_k \| A' \| B' \| C' \| D' \| E' \| \rho'_a \| \rho'_b \| \rho'_c \| \tau \| n_V)$. Then $\mathcal{S}$ selects $n_T \xleftarrow{\$} \{0,1\}^t$ and sets $c = H_4(c' \| n_T \| \mathsf{msg})$.

(5) Finally $\mathcal{S}$ outputs the signature $\sigma = (A', B', C', E', c, s, n_V, n_T)$.

For the case of an adversarially controlled issuer the simulator has to simulate a correct signature production. Since this corresponds to a Join protocol of the case (iHM) the simulator will either already have a validly produced credential or will simply engage in a run of the Join protocol for this case to obtain one. The simulator then engages in a correct run of the Sign protocol to produce a signature σ for a known value of f.

Case (V). Here $\mathcal{S}$ will receive a new signature σ from $\mathcal{A}$ on a message msg with respect to a basename bsn. We bear in mind that this means the host must be corrupted but not necessarily the TPM and throughout will distinguish cases based on this. The simulator first performs a verification check except for the checking for rogue values. If this check fails then $\mathcal{S}$ can just ignore the signature. Otherwise $\mathcal{S}$ performs the rogue check. There are two cases for the behaviour of $\mathcal{S}$ depending on whether the signature was produced by a value of f on the rogue list for this verifier or not.

- If the signature fails the rogue check then this could be either due to m_i being under the control of $\mathcal{A}$ as well as h_i or that the adversary was able to obtain f by some other means. However, during the rogue tagging simulation, if the adversary is able to find f for an honest TPM then $\mathcal{S}$ would abort outputting a failure and hence we need only consider the case where both m_i and h_i are corrupt. When the signature fails the rogue check the simulator is able to obtain the value of f used. The simulator has to then find which m_i to associate the signature to and ensure that in the ideal system any required signature calls are made and that the simulated rogue list, SimRogueList, agrees with RogueList maintained by $\mathcal{T}$. The simulator does this based on the corruption state of the issuer I_k that issued the credential for the signature.

  - If I_k was honest (controlled by $\mathcal{S}$ in the production of the credential signed) then there will be some entry on HList_1 with a value F such that $F = f \cdot P_1$ where f is the value for which σ failed the rogue check. Once this value of F is found $\mathcal{S}$ can identify which m_i produced σ in its simulation and obtain i and cnt. Once it has these it makes a call to $\mathcal{T}$ as h_i to sign msg with respect to cnt and bsn and adds an entry (f, i, cnt_i) to SimRogueList.

    Note the same value of f must be used for the credential issued and for σ since σ correctly verified. If one value of f was used for the credential and

another for the signature on this credential then we would have $\tau' \neq \tau^{\dagger}$ in the verification algorithm.

- If I_k was corrupt there would have been no communication with $\mathcal{S}$, up to this point, from the user for the particular value of credential on which the signature is made. In this case $\mathcal{S}$ is free to choose any corrupt TPM value i and counter value cnt to associate the signature to for a user that is not already a member with respect to cnt. The simulator then makes a call to $\mathcal{T}$ to make this user a member and then tags the user as rogue. The user then makes a call to $\mathcal{T}$ as the host to sign msg with respect to cnt and bsn.

- If σ passes the rogue check then $\mathcal{S}$ checks if it has seen the tuple (c, s, n_T) used in the signature. If so this will be in a signature forgery/simulation with an honest m_i or that this signature has been re-submitted.
  - If σ was simply re-submitted then $\mathcal{S}$ need do nothing (the signature should already be assigned to a user).
  - If (c, s, n_T) was used by $\mathcal{S}$ as part of an honest m_i part of a Sign protocol and $\mathcal{S}$ still owes $\mathcal{T}$ a reply for a basename for this signature then $\mathcal{S}$ replies to $\mathcal{T}$ as the corresponding h_i with the value of bsn.
  - If (c, s, n_T) was used by $\mathcal{S}$ as part of an honest m_i part of a Sign protocol but $\mathcal{S}$ does *not* owe $\mathcal{T}$ a reply for this signature then the adversary must be able to produce multiple distinct signatures for the same tuple (c, s, n_T) and hence must be able to forge signatures by producing new signatures from old signatures so $\mathcal{S}$ terminates outputting "failure 2".

  If (c, s, n_T) are new then σ was completely produced by $\mathcal{A}$ for a corrupted m_i that has correctly verified. The simulator has to assign the pair to a "free" corrupted m_i and counter value and proceeds as follows.
  - If bsn $=\perp$ then $\mathcal{S}$ can select any corrupted m_i and value of cnt such that m_i is corrupted but not tagged as rogue for cnt. Then, acting as the corresponding corrupted host h_i, $\mathcal{S}$ initiates the signing of msg with respect to cnt and bsn with $\mathcal{T}$.
  - If bsn $\neq\perp$ then $\mathcal{S}$ can select any corrupted m_i and counter value cnt such that m_i is not tagged as rogue with respect to cnt and $\mathcal{S}$ has not yet triggered a signature call to $\mathcal{T}$ as h_i with respect to cnt and bsn. If it finds such a "free" module and counter value then it initiates the signing of msg with respect to bsn and cnt with $\mathcal{T}$ as the host. If there is no such module and cnt pair and the issuer, for the credential used in the signature, is honest (controlled by $\mathcal{S}$), then $\mathcal{A}$ must be able to forge signatures and $\mathcal{S}$ terminates outputting "failure 3". If this is the case and the issuer is corrupt, then $\mathcal{S}$ can simply generate such a pair by letting some corrupt m_i and h_i join with respect to some free counter value cnt.

Simulation of RogueTag. We assume that RogueTag will only be run by $\mathcal{S}$ when $\mathcal{S}$ is simulating an honest V_j and is given a tuple (A, B, C, E, f) from $\mathcal{A}$. The simulator has to first decide if this is a validly issued credential on f. It does this by checking that $\hat{t}(A, Y) = \hat{t}(B, P_2)$ and $\hat{t}(A + E, X) = \hat{t}(C, P_2)$ for

that issuer. If any of these do not hold then $\mathcal{S}$ can ignore this. Otherwise we distinguish if the tuple was issued by a corrupt or honest issuer I_k or not issued at all.

Case (I). If, in addition to the issuer being honest, there is some honest m_i with a value of F such that $F = f \cdot P_1$ then $\mathcal{S}$ will have simulated the Join protocol for an unknown value of f and hence the adversary must be able to compute discrete logs so $\mathcal{S}$ aborts outputting "failure 4".

Otherwise f must correspond to a corrupted m_i. Since I_k is honest $\mathcal{S}$ will have a list of all values of F used in issued credentials. The simulator uses this to find which u_i this rogue value of f corresponds to. The simulator then makes a query to $\mathcal{T}$ to advise that this user should be tagged as rogue with respect to the relevant counter and encapsulation key. If no such value of F exists then the adversary must have been able to forge a valid credential without the aid of the issuer. In this case $\mathcal{S}$ aborts outputting "failure 5".

Case (i). Here $\mathcal{S}$ checks to see if it has seen values of B' and E' in a signature such that $E' = f \cdot B'$. There may be several such pairs for the same host (by using the same credential for multiple signatures and different hosts since the value of f is a rogue value). If so $\mathcal{S}$ selects one of these hosts and tags the corresponding m_i as rogue with $\mathcal{T}$. Otherwise $\mathcal{S}$ stores f without assigning it to a user and waits until this credential is used for a signature verification to identify the user who it belongs to.

The proof the the theorem then follows from the description of the simulator $\mathcal{S}$ and Lemmas 2 and 3. □

We then have the following two Lemmas.

Lemma 2. *If the bilinear LRSW assumption holds for groups $\mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle, \mathbb{G}_T$ of large prime order q and the pairing $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$ and the discrete logarithm assumption holds in $\mathbb{G}_1$ then $\mathcal{S}$ will abort with negligible probability.*

Proof. For the proof of this we argue that $\mathcal{S}$ will abort outputting "failure X" for $X \in \{1, 2, 3, 4, 5\}$. We do this by analysing the conditions under which each failure event happens and concluding that these will each occurs with negligible probability. The details are given in the full (ePrint) version of this paper. □

Lemma 3. *In an execution of the above system with simulator $\mathcal{S}$ no computationally bounded environment can distinguish if it's run in the real system with a real adversary or in an ideal system with a simulator provided the adversary used by $\mathcal{S}$ does not abort. Furthermore, no* p.p.t. *adversary can distinguish if it is run in a simulation of the scheme or in an actual execution of the scheme.*

Proof. The proof that the environment cannot distinguish the system it is run in follows from $\mathcal{S}$ copying the exact equivalent behaviour of $\mathcal{A}$ in the ideal system providing $\mathcal{A}$ does not abort. For the proof that $\mathcal{A}$ cannot tell it is run in a simulation we argue that each of the inputs given to $\mathcal{A}$ that are different from those in an actual run of the scheme are computationally indistinguishable from those given in the simulation. The details are given in the full (ePrint) version of this paper. □

Acknowledgements. The second and third authors would like to thank the EPSRC, eCrypt and HP Labs for partially supporting the work in this paper. Most of the work in this paper was done whilst the second author was on placement with HP Labs.

References

1. Backes, M., Maffei, M., Unruh, D.: Zero Knowledge in the Applied Pi–Calculus and Automated Verification of the Direct Anonymous Attestation Protocol. Cryptology ePrint Archive. Report 2007/289 (2007), http://eprint.iacr.org/2007/289
2. Balfe, S., Lakhani, A.D., Paterson, K.G.: Securing Peer-to-Peer Networks using Trusted Computing. In: Mitchell, C. (ed.) Trusted Computing, ch. 10, pp. 271–298. IEEE Computer Society Press, Los Alamitos (2005)
3. Brickell, E., Camenisch, J., Chen, L.: Direct Anonymous Attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 132–145. ACM Press, New York (2004)
4. Brickell, E., Chen, L., Li, J.: Simplified Security Notions for Direct Anonymous Attestation and a Concrete Scheme from Pairings. Cryptology ePrint Archive. Report 2008/104 (2008), http://eprint.iacr.org/2008/104
5. Brickell, E., Chen, L., Li, J.: A New Direct Anonymous Attestation Scheme from Bilinear Maps. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 166–178. Springer, Heidelberg (2008)
6. Camenisch, J., Groth, J.: Group Signatures: Better efficiency and new Theoretical Aspects. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 122–135. Springer, Heidelberg (2005)
7. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
8. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
9. Chen, L., Morrissey, P., Smart, N.P.: Pairings in Trusted Computing. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 1–17. Springer, Heidelberg (2008)
10. Ge, H., Tate, S.R.: A Direct Anonymous Attestation Scheme for Embedded Devices. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, Springer, Heidelberg (2007)
11. Leung, A., Mitchell, C.J.: Ninja: Non-Identity Based, Privacy Preserving Authentication for Ubiquitous Environments. In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 73–90. Springer, Heidelberg (2007)
12. Smyth, B., Chen, L., Ryan, M.: Direct Anonymous Attestation (DAA): Ensuring Privacy with Corrupt Administrators. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 218–231. Springer, Heidelberg (2007)

Cryptanalysis of Vo-Kim Forward Secure Signature in ICISC 2005

Jia Yu[1], Fanyu Kong[2], Xiangguo Cheng[1], Rong Hao[1], and Guowen Li[3]

[1] College of Information Engineering,
Qingdao University, Qingdao 266071, China
{yujia,chengxg,hr}@qdu.edu.cn
[2] Institute of Network Security,
Shandong University, Jinan 250100, China
fanyukong@sdu.edu.cn
[3] School of Computer Science and Technology,
Shandong Jianzhu University, Jinan 250101, China
guowenli@gmail.com

Abstract. D. L. Vo and K. Kim proposed a forward secure signature scheme from bilinear pairings in annual International Conference on Information Security and Cryptology 2005. They claimed that their scheme satisfies several merits including requiring the general security parameters only independent to the total number of time periods and performing key evolving for unlimited time periods while maintaining sizes of keys and signature fixed. They also claimed this scheme is forward secure under the assumption of computational Diffie-Hellman problem. In this paper, we analyze the security of this scheme and point out this scheme doesn't satisfy the forward security.

Keywords: forward security; digital signature; provable security; key exposure.

1 Introduction

Digital signatures are playing increasingly important role in many electronic applications such as e-commerce, digital checks and digital cash. Unfortunately, for regular digital signatures, if the secret key is exposed, not only are all future signatures pertaining to the compromised key invalid, but also all signatures previously signed as well because verifier cannot know a signature is produced before or after the compromise. Exposure of secret keys threatens the security of digital signatures greatly. However, key exposure seems unavoidable in reality especially with ever-increasingly use of portable and unprotected devices. It is much easier for an adversary to break into storage system to get the secret key than to attack the real cryptographic system. Forward secure signatures are proposed to reduce the damage of key exposure. The whole lifetime of signature is divided into multiple time periods. The current secret key is only used for signing messages in this period. At the end of this period, a new secret key is

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 176–184, 2008.

produced from the old one by a key update algorithm, while the public key is fixed during the whole lifetime. Therefore, the exposure of the current secret key doesn't affect the security of the signatures signed in previous periods.

Forward secure signatures can provide potential benefits to many applications such as Certificate Authority (CA). The CA uses its secret key to sign public key certificates of users and certificate revocation lists. The secret key of the CA is usually used for a long time because it is very difficult for the CA to use the new secret key to re-sign all public key certificates of the users. The secret key exposure of ordinary signature would bright devastating consequence in the CA. Fortunately, forward secure signature can greatly reduce the damage of secret key exposure of the CA because all signatures of previous time periods needn't be reissued.

Forward secure signature, firstly proposed by Anderson in an invited lecture [1] and formalized by Bellare and Miner [2] has received much attention. There are two categories of methods to construct the forward secure signature: one is based on modifying special signature scheme; while the other employs ordinary signature scheme as a black box. Schemes [2,3,4,5,6,7] belong to the first category. These schemes based on special signatures such as Ong-Schnoor signature [8] or Guilou-Quisquater signature [9] make use of different techniques to improve the efficiency of key update time or signing and verifying time. In the second category, some method is used to certify (such as by a chain of certificates) the current public key for a particular time period. Schemes [10,11,12,13] belong to this category. These schemes usually need large storage space for storing the current certificates and the keys for issuing future certificates. The verifying algorithms in these schemes need long time because the entire certificate chain has to be verifying in this procedure.

In ICISC2005, Vo and Kim [14] proposed yet another forward secure signature scheme from bilinear pairings. They claimed that their signature scheme had many good merits and was proved to be forward secure assuming CDH problem is hard. In this work, we point out their scheme doesn't satisfy the forward security of signatures.

In the forthcoming section we introduce forward secure signature scheme and its security. A description of Vo-Kim scheme is given in Section 3. In section 4, we give two attacking algorithms for this scheme. In section 5, we give the further analysis of security proof in [14]. Finally, section 6 concludes this paper.

2 Forward Secure Signature Scheme and Its Security

Forward secure signatures can evolve the secret key periodically while keeping the public key unchanged throughout the whole lifetime. The whole lifetime of the signature is divided into many time periods. In each time period, it performs an additional key update algorithm to update the secret key, and then deletes the old secret key. Forward security arises from the fact that the secret key update is one-way and it is computationally infeasible for an adversary to compute the secret keys previous key-exposure period from the exposed secret key. Below

we will give some informal descriptions of the forward secure signature and it security. The formal descriptions can be referred to [2,3].

Forward secure signature scheme is a key-evolving signature scheme that consists of four algorithms. The first algorithm is key generation algorithm in which some security parameters are taken as input and output the secret key in the first period. In the second (key update) algorithm, input the secret key in the current period and output the secret key in the next period. In the third (signing) algorithm, input a time period, a secret key and a message, and output a signature of the message for this time period. In the last (verifying) algorithm, input the public key, a message and a candidate signature, and outputs 1 when the signature is a valid signature or 0, otherwise.

Forward secure signature scheme satisfies that an adaptive chosen-message adversary is computationally infeasible to forge a signature of some previous period even if she discovers the secret key for the current time period.

In the experiment of security analysis, assume that the adversary runs in the following phases: In the chosen message attack phase, the adversary can query a signature oracle to obtain the signature of any message corresponding to the current period. At the end of each time period, the adversary can select to stay in this phase or go to the next period. In the break-in phase, the adversary is given the secret key for the break-in period. In the forgery phase, if the adversary outputs a forgery of a new message for a time period prior to break-in period, then we say the adversary succeeds. If the adversary can't forge a new message for a time period prior to break-in period with a non-negligible probability, then it means the signature scheme satisfies forward security.

3 Review of Vo-Kim Scheme

Following the definition of key-evolving signature scheme, Vo and Kim's forward secure signature scheme is a quadruple of algorithms (*FSIG.key*, *FSIG.update*, *FSIG.sign*, *FSIG.verify*), where

(1)*FSIG.key*: key generation algorithm. Input a security parameter k, and output initial public key and secret key pair (SK_0, PK).

Run a parameter generator IG to generate groups G_1, G_2 of some prime order q and an admissible pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$.

Select a generator $P \in G_1$ and three random values $s, t, r_0 \in_R Z_q^*$. Compute $Q = sP$, $T = tP$.

Set $PK = \{G_1, G_2, \hat{e}, P, Q, T\}$. Choose two hash functions $H_1 : \{0,1\}^* \rightarrow Z_q^*$, $H_2 : \{0,1\}^* \times \{0,1\}^* \times G_1 \rightarrow G_1$.

Compute $s_0 = s + r_0 H_1(0)$; $t_0 = t - r_0 H_1(0)$; $V_0 = t_0 Q_0$.

Erase s, t, r_0, t_0.

Set $SK_0 \leftarrow (s_0, V_0, Q_0)$.

Output initial public key and secret key pair (SK_0, PK).

(2) *FSIG.update*: Input the public key PK, the current time period i and the secret key SK_{i-1} for the previous period. Output the current secret key SK_i.

Parse SK_{i-1} as $(s_{i-1}, V_{i-1}, Q_{i-1})$.
Select $r_i \in_R Z_q^*$ and compute

$$s_i = s_{i-1} + r_i H_1(i);$$
$$Q_i = Q_{i-1} + r_i H_1(i) P;$$
$$V_i = V_{i-1} + r_i H_1(i)(T - Q_{i-1} - Q_i).$$

Erase $s_{i-1}, r_i, Q_{i-1}, V_{i-1}$.
Output the current secret key $SK_i = (s_i, V_i, Q_i)$.

(3)*FSIG.sign*: Input a message M, the current time period i and the current secret key SK_i. Output the signature $< i, \sigma >$ for message M in the period i.
Parse SK_i as (s_i, V_i, Q_i).
Set $U = Q_i$.
Compute

$$\alpha = s_i Q_i + V_i, \text{ and } \beta = s_i H_2(i, M, U).$$

The signer outputs a signature $< i, \sigma = (U, \alpha, \beta) >$.

(4)*FSIG.verify*: Input a signature $< i, \sigma >$ in period i for a message M.
Parse σ as (U, α, β) and verify

$$\hat{e}(\alpha, P) = \hat{e}(U, T + Q);$$
$$\hat{e}(\beta, P) = \hat{e}(H_2(i, M, U), U + Q).$$

If these equations work then return 1, else return 0.

4 The Attacking Algorithms

We construct an adversary F who randomly selects a time period b to break in. The adversary can get the secret key in period b according to the security experiment in Section 2. We assume the secret key is $SK_b = (s_b, V_b, Q_b)$.Two attacking algorithms for the forward security of Vo and Kim's scheme are given below. The first can forge the signature for any message in time period $b - 1$ and the second can forge the signature for any message in any time period i *s.t.* $0 \leq i < b$. In fact, the first algorithm can be viewed as a special case of the second algorithm.

4.1 The First Algorithm

We firstly give an algorithm to forge any message M in time period $b - 1$. The adversary F does as follows:
(1)Randomly selects $r'_b \in_R Z_q^*$, and computes

$$s'_{b-1} = s_b - r'_b H_1(b);$$
$$Q'_{b-1} = Q_b - r'_b H_1(b) P;$$
$$V'_{b-1} = V_b + s_b Q_b - s'_{b-1} Q'_{b-1} - r'_b H_1(b)(T + Q).$$

(2) She selects $(s'_{b-1}, V'_{b-1}, Q'_{b-1})$ as the secret key in time period $b-1$ even if she doesn't know the real secret key of this period.

(3)Let $U = Q'_{b-1}$. She computes

$$\alpha = s'_{b-1}Q'_{b-1} + V'_{b-1}, \text{ and } \beta = s'_{b-1}H_2(b-1, M, U).$$

(4) The adversary F outputs a signature $< b-1, \sigma = (U, \alpha, \beta) >$ as a forgery of message M in time period $b-1$. The forgery can pass the verification because

$$\begin{aligned}
&\hat{e}(\alpha, P)\\
&= \hat{e}(s'_{b-1}Q'_{b-1} + V'_{b-1}, P)\\
&= \hat{e}(s'_{b-1}Q'_{b-1} + V_b + s_bQ_b - s'_{b-1}Q'_{b-1} - r'_bH_1(b)(T+Q), P)\\
&= \hat{e}(V_b + s_bQ_b - r'_bH_1(b)(T+Q), P)\\
&= \hat{e}((s+t)Q_b - r'_bH_1(b)(s+t)P, P)\\
&= \hat{e}(Q_b - r'_bH_1(b)P, (s+t)P)\\
&= \hat{e}(Q'_{b-1}, Q+T)\\
&= \hat{e}(U, Q+T)
\end{aligned}$$

and

$$\begin{aligned}
&\hat{e}(\beta, P)\\
&= \hat{e}(s'_{b-1}H_2(b-1, M, U), P)\\
&= \hat{e}(H_2(b-1, M, U), (s_b - r'_bH_1(b))P)\\
&= \hat{e}(H_2(b-1, M, U), s_bP - r'_bH_1(b)P)\\
&= \hat{e}(H_2(b-1, M, U), Q_b + Q - r'_bH_1(b)P)\\
&= \hat{e}(H_2(b-1, M, U), Q'_{b-1} + Q)\\
&= \hat{e}(H_2(b-1, M, U), U + Q)
\end{aligned}$$

Here equations $V_b + s_bQ = (s+t)Q_b$, $s_bP = Q_b + Q$ hold because

$$\begin{aligned}
&V_b + s_bQ\\
&= V_b + (s_{b-1} + r_bH_1(b))Q\\
&= V_{b-1} + r_bH_1(b)(T - Q_{b-1} - Q_b) + (s + \sum\nolimits_{i=0}^{b} r_iH_1(i))Q_b\\
&= V_{b-1} + r_bH_1(b)(T - Q_{b-1}) + sQ_b + \sum\nolimits_{i=0}^{b-1} r_iH_1(i)Q_b\\
&= V_{b-1} + r_bH_1(b)(T - Q_{b-1}) + sQ_b + \sum\nolimits_{i=0}^{b-1} r_iH_1(i)(Q_{b-1} + r_bH_1(b)P)\\
&= V_{b-1} + r_bH_1(b)(T - Q_{b-1}) + sQ_b + \sum\nolimits_{i=0}^{b-1} r_iH_1(i)Q_{b-1}\\
&\quad + r_bH_1(b)\sum\nolimits_{i=0}^{b-1} r_iH_1(i)P\\
&= V_{b-1} + r_bH_1(b)T + \sum\nolimits_{i=0}^{b-1} r_iH_1(i)Q_{b-1} + s(Q_{b-1} + r_bH_1(b))
\end{aligned}$$

$$
\begin{aligned}
&= V_{b-1} + s_{b-1}Q_{b-1} + r_b H_1(b)T + sr_b H_1(b) \\
&\vdots \\
&= V_0 + s_0 Q_0 + \sum_{i=1}^{b} r_i H_1(i)T + s\sum_{i=1}^{b} r_i H_1(i) \\
&= V_0 + \sum_{i=1}^{b} r_i H_1(i)T + (s + r_0 H_1(0))Q_0 + s\sum_{i=1}^{b} r_i H_1(i) \\
&= V_0 + \sum_{i=1}^{b} r_i H_1(i)T + r_0 H_1(0)Q_0 + s\sum_{i=0}^{b} r_i H_1(i) \\
&= V_0 + \sum_{i=1}^{b} r_i H_1(i)T + r_0 H_1(0)Q_0 + sQ_b \\
&= (t_0 + r_0 H_1(0))Q_0 + \sum_{i=1}^{b} r_i H_1(i)T + sQ_b \\
&= tr_0 H_1(0)P + \sum_{i=1}^{b} r_i H_1(i)T + sQ_b \\
&= \sum_{i=0}^{b} r_i H_1(i)T + sQ_b \\
&= t\sum_{i=0}^{b} r_i H_1(i)P + sQ_b \\
&= (s+t)Q_b
\end{aligned}
$$

$$
\begin{aligned}
&s_b P \\
&= (s_{b-1} + r_b H_1(b))P \\
&= (s_{b-2} + r_{b-1}H_1(b-1) + r_b H_1(b))P \\
&\vdots \\
&= (s + \sum_{i=0}^{b} r_i H_1(i))P \\
&= \sum_{i=0}^{b} r_i H_1(i)P + sP \\
&= Q_b + Q
\end{aligned}
$$

Above verifying procedure can be got from [14].

4.2 The Second Algorithm

In addition, we give another more general algorithm to forge any message M in any time period i *s.t.* $0 \le i < b$. The adversary F does as follows:

(1)Randomly selects $r'_i \in_R Z_q^*$, and computes

$$
\begin{aligned}
s'_i &= s_b - r'_i H_1(i) \\
Q'_i &= Q_b - r'_i H_1(i)P \\
V'_i &= V_b + s_b Q_b - s'_i Q'_i - r'_i H_1(i)(T+Q).
\end{aligned}
$$

(2) She selects (s'_i, V'_i, Q'_i) as the secret key in time period i even if she doesn't know the real secret key in time period i.

(3)Let $U = Q'_i$. She computes

$$\alpha = s'_i Q'_i + V'_i, \text{ and } \beta = s'_i H_2(i, M, U).$$

(4) The adversary F outputs a signature $< i, \sigma = (U, \alpha, \beta) >$ as a forgery of message M in time period i. The forgery can pass the verification because

$$\begin{aligned}
&\hat{e}(\alpha, P) \\
&= \hat{e}(s'_i Q'_i + V'_i, P) \\
&= \hat{e}(s'_i Q'_i + V_b + s_b Q_b - s'_i Q'_i - r'_i H_1(i)(T + Q), P) \\
&= \hat{e}(V_b + s_b Q_b - r'_i H_1(i)(T + Q), P) \\
&= \hat{e}((s + t)Q_b - r'_i H_1(i)(s + t)P, P) \\
&= \hat{e}(Q_b - r'_i H_1(i)P, (s + t)P) \\
&= \hat{e}(Q'_i, Q + T) \\
&= \hat{e}(U, Q + T)
\end{aligned}$$

and

$$\begin{aligned}
&\hat{e}(\beta, P) \\
&= \hat{e}(s'_i H_2(i, M, U), P) \\
&= \hat{e}(H_2(i, M, U), (s_b - r'_i H_1(i))P) \\
&= \hat{e}(H_2(i, M, U), s_b P - r'_i H_1(i)P) \\
&= \hat{e}(H_2(i, M, U), Q_b + Q - r'_i H_1(i)P) \\
&= \hat{e}(H_2(i, M, U), Q'_i + Q) \\
&= \hat{e}(H_2(i, M, U), U + Q)
\end{aligned}$$

5 The Further Analysis of Security Proof in [14]

In their security analysis, to break CDH problem in the additive group G_1, an adversary A is given P, $P' = aP$, $Q' = bP$, where a,b randomly chosen and remain unknown to A. The task of A is to drive $S' = abP$ with the help of the forger F. A provides the public key to F and answer its hash queries, signing queries, and breakin query. First, A guesses a random $i > 0$ at which F will ask for breakin query. Then A sets the public key $PK = (G_1, G_2, \hat{e}, P, Q, T)$, where $Q = Q'$. A provides PK to F and runs it. A can answer the hash queries and the signing queries since it controls the hash will. In order to answer a signature query number n on a message M'_n during the time period $j'_n < n$. A selects randomly: $x_{j'_n}, y_{j'_n}, s_{j'_n} \in_R Z_q^*$, and computes $U'_n = y_{j'_n} P$, $\alpha'_n = y_{j'_n}(Q + T)$, $\beta'_n = x_{j'_n}(U'_n + Q)$, $V'_n = \alpha'_n - s_{j'_n} P$, $h'_n = x_{j'_n} P$ for answer the signature query. A guesses a random index g', and hopes the forgery is based on g'-th hash query. A makes this hash value special, i.e., P'. In forgery phase, F outputs a forgery $< i', \sigma' >$ on a message $M_{g'}$, where $\sigma' = (U', \alpha', \beta')$ and $i' < i$. They assume U' equals to the one in time period $i' < i$, in which A has queried for signatures

and A has value $U' = y'P$ in that time period, otherwise A fails. Then A can compute $abP = \beta' - y^{-1}P'$.

Analysis: However, the probability that A succeeds to compute $abP = \beta' - y^{-1}P'$ is negligible because the probability that U' equals to the one in time period $i' < i$ in which A has queried for signatures is negligible. The reason is that A has no ability to control F to perform choosing message query. Therefore, F may not query any signature in time period i' at all. It is the main mistake in their proof. Therefore, their security theorem doesn't hold.

6 Conclusions

In this paper, we firstly introduce forward secure signature and its security. And then descript Vo-Kim forward secure signature in ICISC 2005. At last we analyze the security of this scheme. By giving two attacking algorithms, we can draw a conclusion that this scheme is not forward secure. Therefore, how to construct a forward secure signature whose full performance parameters including key generation time, key update time, signing time, verifying time, the signature size, the public key size, and the secret key size are all independent of the total number of time periods T and don't change with the current time period increasing is still an open problem [16].

Acknowledgments. We would like to thank anonymous referees of the second international conference on provable security (ProvSec 2008) for the suggestions to improve this paper. This research is supported by National Natural Science Foundation of China (60703089) and the National High-Tech R & D Program (863 Program) of China (2006AA012110).

References

1. Anderson, R.: Two remarks on public key cryptology. Invited Lecture. In: The 4th ACM Conference on Computer and Communications Security (1997)
2. Bellare, M., Miner, S.: A forward-secure digital signature scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
3. Abdalla, M., Reyzin, L.: A new forward-secure digital signature scheme. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 116–129. Springer, Heidelberg (2000)
4. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 499–514. Springer, Heidelberg (2001)
5. Kozlov, A., Reyzin, L.: Forward-secure signatures with fast key update. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 247–262. Springer, Heidelberg (2003)
6. Kang, B.G., Park, J.H., Halm, S.G.: A new forward secure signature scheme. Cryptology ePrint Archive, Report 2004/183 (2004)
7. Camenisch, J., Koprowski, M.: Fine-grained forward-secure signature schemes without ran-dom oracles. Discrete Applied Mathematics 154(2), 175–188 (2006)

8. Ong, H., Schnorr, C.P.: Fast signature generation with a fiat Shamir-like scheme. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 432–440. Springer, Heidelberg (1991)
9. Guillou, L.C., Quisquatr, J.J.: A paradoxical identity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
10. Krawczyk, H.: Simple forward-secure signatures for any signature scheme. In: the 7th ACM conference on Computer and Communications Security, pp. 108–115. ACM Press, New York (2000)
11. Maklin, T., Micciancio, D., Miner, S.: Efficient generic forward-secure signatures with an unbounded number of time periods. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 400–417. Springer, Heidelberg (2002)
12. Boyen, X., Shacham, H., Shen, E., Waters, B.: Forward Secure Signatures with Untrusted Update. In: The 13th ACM conference on Computer and communications security, pp. 191–200. ACM Press, New York (2006)
13. Libert, B., Jacques, J., Yung, M.: Forward-Secure Signatures in Untrusted Update Envi-ronments: Efficient and Generic Constructions. In: The 14th ACM conference on Computer and communications security, pp. 266–275. ACM Press, New York (2007)
14. Vo, D.L., Kim, K.: Yet another forward secure signature from bilinear pairings. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 441–455. Springer, Heidelberg (2006)
15. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing effi-cient protocols. In: The First ACM Conference on Computer and Communications Secu-rity, pp. 62–73. ACM Press, New York (1993)
16. Itkis, G.: Forward Security: Adaptive Cryptography-Time Evolution. Invited chapter for the Handbook of Information Security (2005), http://www.cs.bu.edu/faculty/itkis/pap/forward-secure-survey.pdf

Computationally Sound Symbolic Analysis of Probabilistic Protocols with Ideal Setups*

Zhengqin Luo

INRIA Sophia-Antipolis
Zhengqin.Luo@sophia.inria.fr

Abstract. Recently, many approaches have been proposed for building simple symbolic proofs of cryptographic protocols with computational soundness. However, most of them support only bare-bone execution model without any ideal setup, such as the existence of authenticated channel, and only deterministic protocols. Thus many protocols are not expressible in those models. Following the work of Canetti and Herzog [1], we propose a probabilistic symbolic model for analyzing cryptographic protocols and a general way of incorporating ideal setups by using a probabilistic process calculus. Each ideal setup in the symbolic model will correspond to an ideal functionality in the computational model. Furthermore, we show the computational faithfulness of this symbolic model with respect to a hybrid computational model in which ideal functionalities are employed.

1 Introduction

Proving whether cryptographic protocols are secure is one of the central problems in modern cryptography. Over the last two decades, there are mainly two views on analyzing these protocols. On the one hand, although the computational approach provides a rigorous framework for defining and proving security properties, proofs often involve tedious reductions from probabilistic algorithms to underlying cryptographic schemes, which are generally hard to be automated. On the other hand, the formal method approach proposes a much simpler model for describing and analyzing protocols by using abstract term algebra for modeling perfect cryptography, opening doors to numerous automated tools (or partially automated) in this area. However, the adversary's behavior is restricted by a pre-defined set of operations, which might not capture all possible attacks in the computational model.

Recently, researchers have been making efforts on linking these two approaches, enjoying the simplicity of the formal approach and the rigor in the computational approach at the same time. We confine us now on the method of obtaining soundness results of Dolev-Yao [2] style model with respect to the computational model. The seminal work is proposed by Abadi and Rogaway in [3]. They propose a simple language of encryption terms, and show that equivalence between

* This work was partially done when the author was at Shanghai Jiao Tong University.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 185–199, 2008.

symbolic terms implies computational indistinguishability between ensembles generated by replacing formal encryptions with concrete schemes. Their work only addresses the case of symmetric encryption in the presence of a passive adversary. Subsequently, it is significantly extended to deal with public-key encryption scheme, signature scheme even in the presence of an active adversary [4,5,6,7,8,9,10].

Previous works focus on soundness results of symbolic abstraction of encryption or signature in deterministic protocol models. However, a large number of protocols utilize randomization at their behavioral level, or employ basic cryptographic protocols as building blocks, such as coin-tossing protocol, or even assume the existence of authenticated channel or anonymous channel.

In this work, we consider to introduce *probability* into the symbolic model to address the expressiveness of internal probabilistic behaviors in protocols. And then we consider to introduce *ideal setups* for modeling network assumptions and simple two-party protocols as cryptographic building blocks. The notion of ideal setup is inspired by the ideal functionality in the Universally Composable (UC) security framework [11], a computational framework for designing and analyzing protocols. In that framework, ideal functionality can be used to characterize network assumptions as well as specify security requirements of protocols. Similarly, the ideal setups in our framework play two important roles:

- Modeling network assumptions, such as authenticated channel, anonymous channel, and etc.
- Modeling cryptographic protocols as basic building blocks, such as coin-tossing protocol, oblivious transfer protocol, and zero-knowledge proof protocol, for constructing large and complex protocols.

Ideal setups should capture appropriate intuitions for each tasks. It provides a modular approach for analyzing symbolic protocols: when analyzing a protocol which invokes basic cryptographic protocols mentioned above, we can analyze only an "abstract" protocol using corresponding ideal setups, and then claim that the original protocol satisfies same properties as the "abstract" protocol. We follow the approach of Canetti and Herzog [1], using the UC framework as the underlying computational framework.

The contributions of our paper are:

- We define a simple language for describing protocol programs which can be both interpreted in the symbolic model and the computational model. By the language, we are able to express multi-party protocols in which parties can make internal probabilistic choice, employ public-key encryption and signature, or use ideal setups. The language supports two ideal setups: authenticated channel and coin-tossing protocol (Section 3).
- We show how to interpret protocol programs in our symbolic model by using a subset of the Probabilistic Applied Pi (PAPi) calculus [12]. Each ideal setup is implemented by an auxiliary process (Section 4).
- We also explain how to translate protocol programs into a certain hybrid model of the UC framework, in which ideal functionalities corresponding to ideal setups in the symbolic model (Section 5).

– We finally demonstrate the faithfulness of our symbolic model with respect to the hybrid model. That is, almost all attacks in the hybrid model can be interpreted in the symbolic model, except for negligible probability (Section 6).

Related works. The efforts on linking the symbolic approach and the computational one, especially showing soundness results of Dolev-Yao model with respect to certain computational model, were initiated by Abadi and Rogaway [3], and were extended by [4,5,6,7,8,9,10] in several aspects. Backes, Pfitzmann and Waidner proposed an abstract cryptographic library based on the reactive simutability setting [13,14,15]. Their symbolic model supports nested operation of cryptographic primitives, such as symmetric and public-key encryption, signature, and message authenticated code under arbitrary active adversary. They also demonstrated for several protocols that Dolev-Yao style proof implies the computational security [16,17,18]. Based on the UC framework, Canetti and Herzog established the soundness of symbolic analysis with respect to the UC framework [1]. However, they only considered a restricted class of protocols and only support certified public-key encryption. Patil extended it to handle also standard signature [19].

2 Background

This section provides necessary background information on the topic. We first introduce a subset of the PAPi calculus. We then briefly review the UC framework and the Universally Composable Symbolic Analysis (UCSA) framework.

2.1 A Subset of PAPi Calculus

The applied pi calculus is an extension of the pure pi calculus. It introduces terms and equations over terms for modeling cryptographic primitives, and uses techniques in process algebra to reason about distributed systems and protocols. The PAPi extends it into a probabilistic framework, allowing analysis of probabilistic processes.

The basics elements in the PAPi calculus are a set of *names*, a set *variables*, and a *signature* Σ which consist of a finite set of function symbols with arities. Terms can be defined by applying function symbols in Σ on names, variables and terms. We can equip a given signature Σ with an *equational theory* E to relate two terms which are syntactically different but semantically equivalent.

The grammar of plain processes and extended processes are given below:

$$P, Q ::= \mathbf{0} \mid \bar{u}\langle M\rangle.P \mid u(x).P \mid P \oplus_{\frac{1}{2}} Q \mid !P \mid P|Q \mid \nu n.P \mid \text{if } M = N \text{ then } P \text{ else } Q$$

$$A, B ::= P \mid \nu n.A \mid \nu x.A \mid A|B \mid \{M/x\}$$

Comparing to the original grammar of plain process in the PAPi calculus, we omit the non-deterministic choice operator, and confine the probabilistic choice operator $\oplus_p$ to only one half probabilistic operator $\oplus_{\frac{1}{2}}$. Extending the plain processes with active substitution $\{M/x\}$, we are able to reason about the static knowledge exposed by a certain process. An evaluation context $C[\_]$ is a process with a hole under restriction or parallel composition. For space reason, we do not present the semantics of PAPi here, it can be found in [12,20,21]

2.2 The UC Framework and UCSA Framework

The UC framework provides a general methodology for designing and analyzing cryptographic protocols, especially asserting whether a given protocol securely realizes its security specification. The most salient feature of this framework is the strong composable property, by which one can ensure that a protocol still maintains its security properties when being executed in an arbitrary unpredictable environment, or being composed in a modular way. An comprehensive overview of the UC framework can be found in [11].

The UCSA framework facilitates the *universal composition theorem* to simplify their framework of sound symbolic analysis. The symbolic Dolev-Yao model is a simplified model for analyzing two-party deterministic protocol which uses only public-key encryption. Each protocol peer is defined by a mapping from current state and incoming messages to outgoing messages. The adversary is only limited to a set of symbolic operations according to the rules representing its limitation with respect to perfect cryptography.

Instead of establishing the computational faithfulness of the symbolic model with respect to a concrete computational model using concrete public-key encryption scheme, the UCSA shows that the symbolic model is faithful for a hybrid model in the UC framework. The hybrid model uses $\mathcal{F}_{\text{CPKE}}$ for idealized encryption service which is secure unconditionally, even in the presence of a computational unbounded adversary. Since $\mathcal{F}_{\text{CPKE}}$ can be UC-realized by any CCA-secure[1] encryption scheme, it serves as a "bridge" between the symbolic model and the concrete model: when we obtain a sound symbolic proof of protocol in the hybrid model, we can facilitate the UC theorem to replace each instance of $\mathcal{F}_{\text{CPKE}}$ to an instance of CCA-secure encryption scheme, while maintaining the security in a computational sense at the same time.

3 A Simple Language for Probabilistic Protocols

We first present a simple language for describing behavior of probabilistic protocols. The language can be used to describe high-level codes of protocols without any implementation detail of network communication and underlying cryptography. We could compile a protocol program either into symbolic model by using abstract term algebra, or into computational model by using concrete cryptographic schemes.

[1] Chosen Ciphertext Attack.

Definition 1 (Protocol Program). *Fixed a finite set of identifier of parties* $\mathcal{C} = \{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{M}\}$*, a protocol program is defined by a function* $\mathcal{P} = \{(\mathtt{A}, \mathtt{P_A}), (\mathtt{B}, \mathtt{P_B}), \ldots, (\mathtt{M}, \mathtt{P_M})\}$*, mapping each identifer to a program defined by the grammar below, where* x, m, c, s, b *represent variables for different types of values.*

$$
\begin{aligned}
R ::=\ & \mathtt{A} \mid \mathtt{B} \mid \ldots \mid \mathtt{M} \\
B ::=\ & \mathtt{true} \mid \mathtt{false} \\
I ::=\ & x := \mathtt{newnonce}() \mid x := \mathtt{encrypt}(m, R) \mid x := \mathtt{decrypt}(c) \mid \\
& x := \mathtt{sign}(m) \mid x := \mathtt{verify}(m, s, R) \mid x := \mathtt{pair}(m_1, m_2) \mid \\
& x := \mathtt{fst}(m) \mid x := \mathtt{snd}(m) \mid x := R \mid x := B \\
E ::=\ & \mathtt{input}(x) \mid \mathtt{output}(m) \mid \mathtt{send}(m) \mid \mathtt{recv}(x) \mid \\
& \mathtt{send}^{\mathtt{a}}(m, R) \mid \mathtt{recv}^{\mathtt{a}}(x, R) \mid \mathtt{coin}^{\mathtt{i}}(b, R) \mid \mathtt{coin}^{\mathtt{r}}(b, R) \\
P ::=\ & I \mid E \mid P; P \mid \mathtt{if}\ x = y\ \mathtt{then}\ P\ \mathtt{else}\ P \mid \mathtt{prob}(P; P)
\end{aligned}
$$

Internal Computations. The grammar structure I defines atomic internal computation for protocol parties. A party could generate a fresh random nonce by command `newnonce`. It could also encrypt a message with someone's public-key by `encrypt`, or decrypt a ciphertext with its own private-key by `decrypt`. We assume that each party's identity is bound to its public-key, which cannot be revealed to others. Thus, we use an identity of a party instead of a public-key when encrypting. The commands for generating and verifying signatures, `sign` and `verify`, are similar to the case of public-key encryption. `pair`, `fst` and `snd` are standard paring operation.

External Interactions. `input` and `output` are used to obtain input from the environment and return output to it. `send` and `recv` model the sending and receiving over an adversary-controlled network. Intuitively, it models an asynchronous network, and the adversary could learn, intercept, modify, and re-schedule messages over this network.

Ideal Setups. In addition to common external interactions, we introduce two ideal setups. The first one is the ideal setup of authenticated channel. A party could send an authenticated message m to party `B` by command $\mathtt{send}^{\mathtt{a}}(m, \mathtt{B})$, and could wait to receive an authenticated message from party `B` by $\mathtt{recv}^{\mathtt{a}}(x, \mathtt{B})$ and store it in x. By our assumption, the adversary should not be able to modify or reproduce authenticated messages, but it can learn and intercept them. The second one is the ideal setup of coin-tossing protocol. Commands $\mathtt{coin}^{\mathtt{i}}$ and $\mathtt{coin}^{\mathtt{r}}$ allow two parties to initiate an ideal coin-tossing protocol as initiator and responder, respectively. By assumptions, the adversary should not be able to influence the fairness of the common coin by any means.

Control Flows. We provide three types of control flows here. The sequential execution ($\mathtt{P_1}; \mathtt{P_2}$) and conditional execution ($\mathtt{if}\ M = N\ \mathtt{then}\ \mathtt{P_1}\ \mathtt{else}\ \mathtt{P_2}$) are usual. Command $\mathtt{prob}(\mathtt{P_1}, \mathtt{P_2})$ means executing $\mathtt{P_1}$ with probability 0.5 and executing $\mathtt{P_2}$ with the rest. The reason why we only model one half choice here is that choices with arbitrary probability could always be approximated by multiple one half choices.

For example, a simple challenge-response protocol can be described as follows:

A's program :	B's program :
$N_a = \texttt{newnonce}();$	$\texttt{recv}(m_1');$
$x_1 = \texttt{pair}(N_a, \texttt{B});$	$x_1' = \texttt{decrypt}(m_1');$
$m_1 = \texttt{encrypt}(x_1, \texttt{B});$	$N = \texttt{fst}(x_1');$
$\texttt{send}(m_1);$	$\texttt{send}(N);$
$\texttt{recv}(m_2);$	
$\texttt{if } m_2 = N_a \texttt{ then } output(\texttt{true});$	

Fig. 1. Protocol program for a simple challenge-response protocol

4 Symbolic Interpretation

In this section, we show how to translate a protocol program into a symbolic protocol in our symbolic model which is described by the subset of PAPi calculus introduced in Section 2.1. The abstract term algebra is modeled by an equational theory. Each single party is modeled by a process in the calculus. Also, we demonstrate how to characterize ideal setups by auxiliary processes.

Equational theory. We use two types of function symbol for different purposes, as showed below.

- Constant: `true`/0, `false`/0, `garb`/0.
- Cryptographic operator: `enc`/2, `dec`/2, `pk`/1, `sign`/2, `ver`/3, `vk`/1, `pair`/2, `fst`/1, `snd`/1.

The number followed by each function symbol indicates its arity. The `true` and `false` are boolean constants. `grab` refers to ill-formed terms such as an inappropriate decrypted term. The operations of encryption, signature and paring are defined in a usual way. The cryptography is modeled in a Dolev-Yao style as being perfect. The equations are given in Figure 2. These equations are fairly

$$\begin{aligned}
&\texttt{fst}(\texttt{pair}(x, y)) = x \\
&\texttt{snd}(\texttt{pair}(x, y)) = y \\
&\texttt{dec}(\texttt{enc}(x, \texttt{pk}(y)), y) = x \\
&\texttt{ver}(x, \texttt{sign}(x, y), \texttt{vk}(y)) = \texttt{true}
\end{aligned}$$

Fig. 2. Equational theory E for symbolic protocols

standard for those primitives. The function symbol `pk` maps user's private key to its public key. One can use `dec` and its private key to decrypt a message encrypted under corresponding public key. Without the private key, the adversary cannot learn anything about the message. The case for signature is similar. In addition, we require that improperly operated terms equate to `garb`, such as decryption of a non-encrypted term.

Translating protocol program. Given a protocol program $\mathcal{P}$ for identity set $\mathcal{C}$. For every $\mathtt{R} \in \mathcal{C}$, we translate $\mathcal{P}(\mathtt{R})$ to a process $\mathcal{S}_{\mathtt{R}}$ which uses the following channel names to interact with other processes.

- $input_{\mathtt{R}}$, $output_{\mathtt{R}}$: Obtaining inputs and returning outputs;
- $send_{\mathtt{R}}$, $recv_{\mathtt{R}}$: Sending and receiving over the adversary-controlled network;
- $send^a_{\mathtt{RA}}$, $recv^a_{\mathtt{AR}}$: Sending and receiving authenticated messages with $\mathtt{A}$, for $\mathtt{A} \in \mathcal{C}$;
- $toss^i_{\mathtt{RA}}$, $toss^r_{\mathtt{AR}}$: Initiating and responding coin-tossing with $\mathtt{A}$, for $\mathtt{A} \in \mathcal{C}$.

First, internal computations in $\mathcal{P}(\mathtt{R})$ will be translated to operations of function symbols according to the equational theory E. Then, external interactions will be translated to interactions on channels defined above. Finally, control flows in $\mathcal{P}(\mathtt{R})$ are expressible in PAPi calculus.

Definition 2 (Symbolic interpretation of protocol program). *Given an identity $\mathtt{R} \in \mathcal{C}$, a single protocol program $\mathcal{P}(\mathtt{R})$ can be inductively translated to a process $S_{\mathtt{R}}$ by rules defined in Appendix A.*

In Appendix A, We also give the translation of the simple challenge-response protocol in Figure 1.

Modeling ideal setups. We construct auxiliary processes to capture the intuition of our ideal setups of authenticated channel and coin-tossing protocols.

When channels are authenticated, if $\mathtt{B}$ receives a message m from $\mathtt{A}$, $\mathtt{A}$ must have sent m before. Furthermore, if $\mathtt{A}$ sent m to $\mathtt{B}$ only k times, then $\mathtt{B}$ will not receive m more than k times. Notice that the secrecy of those transmitted messages is not guaranteed, and messages are transmitted asynchronously. We use the following process to help $\mathtt{A}$ and $\mathtt{B}$ to exchange a message over channel $send^a_{\mathtt{AB}}$ and $recv^a_{\mathtt{AB}}$.

$$\begin{aligned}\mathcal{A}^a_s(\mathtt{A},\mathtt{B}) ::= \; & send^a_{\mathtt{AB}}(x).\\ & \nu c.(\overline{adv^a_{\mathtt{AB}}}\langle(x,c)\rangle.\\ & c(y).\\ & if\ y = x\ then\ \overline{recv^a_{\mathtt{AB}}}\langle x\rangle)\end{aligned}$$

The channels $send^a_{\mathtt{AB}}$ and $recv^a_{\mathtt{AB}}$ will be restricted between parties and the auxiliary process only, which are invisible to the adversary. Thus, the adversary cannot modify or reproduce a message. However, it can learn the message through channel $adv^a_{\mathtt{AB}}$. The fresh channel name c is used to distinguish different instances of the auxiliary processes between the same two participants. We use replication operator to allow each participant in set $\mathcal{C}$ being able to send and receive unlimited number of messages.

$$\mathcal{A}^a ::= \prod_{\mathtt{A},\mathtt{B}\in\mathcal{C},\mathtt{A}\neq\mathtt{B}} !\mathcal{A}^a_s(\mathtt{A},\mathtt{B})$$

For the ideal setup of coin-tossing protocol, it requires that two parties should be able to generate an unbiased random bit value. The adversary should not be

able to influence the result by any means. The auxiliary processes for generating a single bit can be defined as follows.

$$\begin{aligned}\mathcal{A}^t_{half}(\mathtt{A},\mathtt{B},x) ::= \nu c.\, \Big(toss^i_{\mathtt{AB}}().\overline{adv^i_{\mathtt{AB}}}\langle c\rangle.c().\overline{toss^i_{\mathtt{AB}}}\langle x\rangle\Big) \mid \\ \nu c.\, \big(toss^r_{\mathtt{AB}}().\overline{adv^r_{\mathtt{AB}}}\langle c\rangle.c().\overline{toss^r_{\mathtt{AB}}}\langle x\rangle\big)\end{aligned}$$

$$\mathcal{A}^t_s(\mathtt{A},\mathtt{B}) ::= \mathcal{A}^t_{half}(\mathtt{A},\mathtt{B},\mathtt{true}) \oplus_{0.5} \mathcal{A}^t_{half}(\mathtt{A},\mathtt{B},\mathtt{false})$$

The participant A initiates an instance of the ideal protocol via the channel name $toss^i_{\mathtt{AB}}$. Upon the request of the adversary, A receives an unbiased random bit. Then, the participant B as a responder gets the same random bit via channel $toss^r_{\mathtt{AB}}$ upon the request of the adversary. It can be seen that the choice of the random bit is independent to the behavior of the adversary. By using replication, we obtain an ideal process enabling each pair of parties to generate arbitrary long random strings.

$$\mathcal{A}^t ::= \prod_{\mathtt{A},\mathtt{B}\in\mathcal{C},\mathtt{A}\neq\mathtt{B}} !\mathcal{A}^t_s(\mathtt{A},\mathtt{B})$$

Adversary and executions. Given all the processes of the participants in the protocol, the process denoting whole protocol can be defined as follows, where $\mathcal{V}$ contains channel name between parties and auxiliary processes, which should not be observed by the adversary:

$$\mathcal{S} = \nu\mathcal{V}.\,\big(\,\mathcal{S}_{\mathtt{A}} \mid \mathcal{S}_{\mathtt{B}} \mid \ \ldots\ \mid \mathcal{S}_{\mathtt{M}} \mid \mathcal{A}^a \mid \mathcal{A}^i\big)$$

The adversary could freely interact with $\mathcal{S}$ by the following free channel names:

- $send_{\mathtt{A}}$, $recv_{\mathtt{A}}$: Controlling the insecure network;
- $adv^a_{\mathtt{AB}}$: Influence the auxiliary process of an authenticated channel;
- $adv^i_{\mathtt{AB}}$, $adv^r_{\mathtt{BA}}$: Influence the auxiliary process of an instance of coin-tossing protocol;

Naturally, the adversary can be modeled as a process $\mathcal{P}_{adv}$, which only contains free channel name described above. Also, we could construct an environment process $\mathcal{P}_{env}$ that provides input to each user process $\mathcal{S}_{\mathtt{R}}$ and receives outputs from them. We require that $\mathcal{P}_{env}$ contains only channel name $input_{\mathtt{R}}$ and $output_{\mathtt{R}}$, for $\mathtt{R}\in\mathcal{C}$. Given the adversary process $\mathcal{P}_{adv}$, an environment process $\mathcal{P}_{env}$, we can construct a context $C^{env}_{adv}[\_] = (\mathcal{P}_{adv}|\mathcal{P}_{env}|\_)$. Thus the execution of the protocol in the presence of the adversary can be seen as the interaction between $\mathcal{S}$ and $C^{env}_{adv}[\_]$.

5 Computational (Hybrid) Interpretation

Next, we define the computational model in our framework. It is a hybrid model in the UC framework which supports ideal functionalities. We elaborate ideal

functionalities for our ideal setups, and then present how to interpret protocol programs in this model. Notice that we do not directly translate symbolic protocols to concrete protocols, avoiding implementation details in the symbolic models.

Ideal functionalities for encryption and signature. Recall that public-key encryption scheme and signature scheme denote cryptographic primitives which can be implemented by local algorithms. Our formulation of these operations is the same as the original UCSA framework [1,19]. The public-key encryption is modeled by the certified public-key encryption functionality $\mathcal{F}_{\text{CPKE}}$, and the digital signature is modeled by the certification functionality $\mathcal{F}_{\text{CERT}}$. As we have assumed, these functionalities do not deal with key issues, and provide unconditional security, in which ciphertext (or signature) bears no computational relation to plaintext. Each party uses identities during encrypting/decrypting and signing/verifying. Note that $\mathcal{F}_{\text{CPKE}}$ can be securely realized by a CCA-secure encryption scheme with a trusted key-registration service. Similarly, $\mathcal{F}_{\text{CERT}}$ can be securely realized by a CMA-secure[2]. We do not give the detail definitions here, interested readers are refer to [1,19].

Ideal functionalities for ideal setups. For the ideal setup of authenticated channel, we present the authenticated communication ideal functionality $\mathcal{F}_{\text{AUTH}}$, as shown in Figure 3. It is originally formulated by Canetti in [11]. We omit the case

Functionality $\mathcal{F}_{\text{AUTH}}$

1. Upon receiving an input (Send, sid, m) from party P, do: If $sid = (P, R, sid')$ for some R, then generate a public delayed output (Send, sid, m) to R and halt. Otherwise ignore the input.

Fig. 3. Authenticated Communication Ideal functionality

of corrupted parties, since we do not consider party corruption in our symbolic model. One could see that the behavior of an instance of $\mathcal{F}_{\text{AUTH}}$ between $\mathtt{A}$ and $\mathtt{B}$ is essentially identical to the ideal process $\mathcal{A}_s^a(\mathtt{A}, \mathtt{B})$.

For the ideal setup of coin-tossing protocol. We present the coin-tossing ideal functionality $\mathcal{F}_{\text{CT}}$, as shown in Figure 4. This functionality can be thought as a "bridge" between a concrete cryptographic protocol of coin-tossing and the ideal process of coin tossing in our symbolic model. We can see that the coin is generated independently by $\mathcal{F}_{\text{CT}}$, which is certainly uncorrelated to the adversary's behavior.

Executing protocol program. Given a protocol program $\mathcal{P}$ with a set of identity $\mathcal{C}$, we specify an Interactive Turing Machine to interpret for a single protocol program $\mathcal{P}(\mathtt{R})$ for each $\mathtt{R} \in \mathcal{C}$.

[2] Chosen Message Attack.

Functionality $\mathcal{F}_{\mathrm{CT}}$

$\mathcal{F}_{\mathrm{CT}}$ proceeds as follows, running with parties P and R.

1. Upon receiving an input $(\mathsf{Toss}, sid, P, R)$ from party P, randomly choose a bit value r, and send $(\mathsf{Tossed}, sid, P)$ to the adversary S.
2. Upon receiving $(\mathsf{Result}, sid, P)$, send $(\mathsf{Tossed}, sid, r)$ to P.
3. Upon receiving an input $(\mathsf{Toss}, sid, P, R)$ from party R, send $(\mathsf{Tossed}, sid, R)$ to the adversary S.
4. Upon receiving $(\mathsf{Result}, sid, R)$, send $(\mathsf{Tossed}, sid, r)$ to R.

Fig. 4. Coin-tossing Ideal Functionality

We briefly sketch how this machine performs. Given a single protocol program $\mathcal{P}(\mathtt{R})$, ITM $M_{\mathtt{R}}$ maintains a memory state Σ which maps each variable occurred in $\mathcal{P}(\mathtt{R})$ into a concrete bit-string, with $\mathtt{PID}_{\mathtt{R}}$ as its concrete party identifier. Each bit-string is tagged with its type in order to maintain a structure of each message as in the abstract term algebra. For example, a bit-string c of ciphertext will be recorded as $\langle$"ciphertext", $c\rangle$, and bit-string r of random nonce will be recorded as $\langle$"nonce", $r\rangle$. At the beginning, each variable is mapped to an uninitiated value. Then $M_{\mathtt{R}}$ starts to interpret commands $\mathcal{P}(\mathtt{R})$ one by one, updating the memory state. Commands related to cryptographic operations and ideal setups will be executed as calls to appropriate instances of ideal functionalities.

- $x = \mathtt{encrypt}(m, \mathtt{A})$: Send $(\mathtt{Encrypt}, \mathtt{PID}_{\mathtt{A}}, m)$ to $\mathcal{F}_{\mathrm{CPKE}}$, receive c, and update x with $\langle$"ciphertext", $c\rangle$.
- $x = \mathtt{decrypt}(c)$: If the value of c is $\langle$"ciphertext", $c'\rangle$, then send $(\mathtt{Decrypt}, \mathtt{PID}_{\mathtt{R}}, c')$ to $\mathcal{F}_{\mathrm{CPKE}}$, receive m, and update x with m.
- $x = \mathtt{sign}(m)$: Send $(\mathtt{Sign}, \mathtt{PID}_{\mathtt{R}}, m)$ to $\mathcal{F}_{\mathrm{CERT}}$, receive s, and update x with $\langle$"signature", $s\rangle$.
- $x = \mathtt{verify}(m, s, \mathtt{A})$: If the value of s is $\langle$"signature", $s'\rangle$, then send $(\mathtt{Verify}, \mathtt{PID}_{\mathtt{A}}, m, s')$ to $\mathcal{F}_{\mathrm{CERT}}$, receive b, and update x with $\langle$"boolean", $b\rangle$.
- $\mathtt{send}^{\mathtt{R}}(m, \mathtt{A})$: Send $(\mathtt{Send}, \langle \mathtt{R}, \mathtt{A}\rangle, m)$ to $\mathcal{F}_{\mathrm{AUTH}}$.
- $\mathtt{recv}^{\mathtt{R}}(x, \mathtt{A})$: Receive m from $\mathcal{F}_{\mathrm{AUTH}}$, and update x with m.
- $\mathtt{coin}^{\mathtt{i}}(b, \mathtt{A})$: Send $(\mathtt{Toss}, \langle \mathtt{R}, \mathtt{A}\rangle)$ to $\mathcal{F}_{\mathrm{CT}}$, receive b, and update x with $\langle$"boolean", $b\rangle$
- $\mathtt{coin}^{\mathtt{r}}(b, \mathtt{A})$: Send $(\mathtt{Toss}, \langle \mathtt{A}, \mathtt{R}\rangle)$ to $\mathcal{F}_{\mathrm{CT}}$, receive b, and update x with $\langle$"boolean", $b\rangle$

Particularly, when it encounters a $\mathtt{prob}(\mathtt{P}_1, \mathtt{P}_2)$ command, it flips a coin and decides which branch to follow. Rest of these commands are executed in a standard way. Thus, we could obtain a set of ITMs $\mathcal{P}_h = \{M_{\mathtt{A}}, M_{\mathtt{B}}, \ldots, M_{\mathtt{M}}\}$ for executing the protocol program in the hybrid model.

6 Faithfulness of the Symbolic Model

In this section, we first define execution traces of symbolic protocol and hybrid protocol, and then we show our symbolic model is faithful with respect to our hybrid model.

Definition 3 (Trace of symbolic protocol). *Let $\mathcal{P}_s = \{\mathcal{S}_{\mathtt{A}}, \mathcal{S}_{\mathtt{B}}, \ldots, \mathcal{S}_{\mathtt{M}}\}$ be a symbolic protocol. Given the adversary process P_{adv} and the environment process P_{env}, the execution of the protocol can be viewed as interaction between the process $\mathcal{S} = \nu\mathcal{V}.\left(\mathcal{S}_{\mathtt{A}} \mid \mathcal{S}_{\mathtt{B}} \mid \ldots \mid \mathcal{S}_{\mathtt{M}} \mid \mathcal{A}^a \mid \mathcal{A}^i\right)$ and the context $C^{env}_{adv}[\_] = (\mathcal{P}_{adv}|\mathcal{P}_{env}|\_)$ under a scheduler F for resolving non-determinism. Given a terminating execution $e = \mathcal{S} \xrightarrow{\alpha_1}_{\mu_1} \mathcal{S}_2 \xrightarrow{\alpha_2}_{\mu_2} \cdots \xrightarrow{\alpha_k}_{\mu_k} \mathcal{S}_k$, the symbolic trace $\mathtt{tr}(e)$ is defined as: $\mathtt{tr}(e) = \mathtt{tr}(\alpha_1);\mathtt{tr}(\alpha_2);\ldots;\mathtt{tr}(\alpha_k)$. The definition of $\mathtt{tr}$ is given in Appendix B*[3].
Let $\mathrm{STRACE}^{F,\mathcal{P}_{adv}}_{\mathcal{P}_s,\mathcal{P}_{env}}$ be the random variable of symbolic traces, such that for every terminated execution e:

$$\Pr[\mathrm{STRACE}^{F,\mathcal{P}_{adv}}_{\mathcal{P}_s,\mathcal{P}_{env}} = \mathtt{tr}(e)] = Prob^F_{\mathcal{S}}(e)$$

We show that the adversary's power is limited by the equational theory in our symbolic model.

Proposition 1 (Closure property). *Given a symbolic trace t, such that*

$$\Pr[\mathrm{STRACE}^{F,\mathcal{P}_{adv}}_{\mathcal{P}_s,\mathcal{P}_{env}} = t] > 0$$

Let t_i in t be an adversary event with term M. Let ϕ be the frame recovered from $t_1, \ldots, t_{i-1}$. Then we have

$$\phi \vdash M$$

Proof. It is straightforward since the adversary $\mathcal{P}_{adv}$ is also a legal PAPi process. □

For the execution trace of hybrid protocols, we record activations of each entity in a sequential manner.

Definition 4 (Traces of hybrid protocols). *Let $\mathcal{P} = \{M_{\mathtt{A}}, M_{\mathtt{B}}, \ldots, M_{\mathtt{M}}\}$ be a hybrid protocol. Given the environment $\mathcal{Z}$, the input z, a security parameter k. Let r_c be random inputs for $\mathcal{Z}$, $\mathcal{F}_{\mathrm{CPKE}}$, and $\mathcal{F}_{\mathrm{CERT}}$, r_b be random inputs for each party and $\mathcal{F}_{CT}$. The execution trace $\mathrm{TRACE}_{\mathcal{P},\mathcal{Z}}(z,k,r_c,r_b)$ can be inductively defined by rules given in Appendix C.*
Let $\mathrm{TRACE}_{\mathcal{P}_h,\mathcal{Z}}(k,z)^*$ be the random variable describing $\mathrm{TRACE}_{\mathcal{P}_h,\mathcal{Z}}(k,z,r_c,r_b)$ when r_c and r_b are uniformly chosen.

Then, we define a mapping from hybrid traces to symbolic traces, translating each concrete event in hybrid traces into a symbolic event. Furthermore, we also define the validity of translated trace.

[3] The execution trace and its probability are defined in [12].

Definition 5 (Mapping from hybrid trace to symbolic trace). *Let t be a hybrid trace of an execution of hybrid protocol $\mathcal{P}_h$. We inductively define the mapping of t to a symbolic trace $symb(t)$ in two steps:*

1. *First, we translate each concrete string in each hybrid event t_i to corresponding symbolic terms, inductively by defining a partial mapping f from bit-strings to symbolic terms. Recall that we use a type tag for different types of values, this could help us to reconstruct symbolic terms using function symbols. We map all ill-formed bit-string to garbage term* `garb`.
2. *Secondly, we convert each hybrid event $\mathtt{E}_i$ to a symbolic interaction t_i, replacing concrete string in $\mathtt{E}_i$ into corresponding symbolic terms using f. If t_i is an adversary event with term M which does not satisfy closure property, then set $t_i = [\text{“}fail\text{”}, M]$*

If t is a random variable of hybrid traces, then $symb(t)$ would be random variable of symbolic traces.

Definition 6 (Valid symbolic traces). *Given a symbolic protocol $\mathcal{P}_s$ and a symbolic traces $symb(t)$ generated from hybrid trace t. We say that t is valid for $\mathcal{P}_s$ if and only if there exist an adversary process $\mathcal{P}_{adv}$, an environment processes $\mathcal{P}_{enc}$, and a scheduler F, such that*

$$\Pr[\mathrm{STRACE}^{F,\mathcal{P}_{adv}}_{\mathcal{P}_s,\mathcal{P}_{env}} = symb(t)] > 0$$

Otherwise, we say that t is not valid for $\mathcal{P}_s$.

Now, we are ready to state the faithfulness theorem of our symbolic model. Intuitively, the theorem says that all but a negligible fraction of hybrid traces can be interpreted in our symbolic model. Therefore, the limited symbolic adversary precisely captures the ability of a more powerful adversary in the hybrid model. Relying on this, it is suffice to analyze protocols in the symbolic model, and then claim that its hybrid counterpart satisfies same properties as the symbolic protocol.

Theorem 1 (Faithfulness of the symbolic model). *For all protocol program $\mathcal{P}$, environments $\mathcal{Z}$, and inputs z of length polynomial in the security parameter k,*

$$\Pr[t \leftarrow \mathrm{TRACE}_{\mathcal{P}_h,\mathcal{Z}}(k,z)^* : symb(t) \text{ is not valid for } \mathcal{P}_s] \leq \epsilon(k)$$

where ϵ is negligible[4].

7 Conclusions

In this paper, we have presented a probabilistic framework for computationally sound symbolic analysis of security protocols. We introduced ideal setups for modeling network assumptions as well as two-party cryptographic primitives, and we then showed how to interpret them in both symbolic model and hybrid model. Finally we demonstrated that our symbolic model is faithful with respect to the hybrid model.

[4] ϵ is *negligible* if $\forall c \geq 0. \exists k_c$ s.t. $\forall k \geq k_c. \epsilon(k) \leq k^{-c}$.

For future research directions, one promising direction is to introduce more ideal setups to the symbolic model, such as anonymous channel, blind signature scheme, commitment protocol, or even zero-knowledge proof protocol. Another interesting direction is to develop automated verification technique for the probabilistic symbolic model to obtain sound automated proof of cryptographic protocols.

Acknowledgment. The author would like to thank Xiaojuan Cai for helpful discussion while evaluating this work. The author also thank Jun Pang and Tamara Rezk for insightful comments. Finally, many thanks to the anonymous referees for helpful feedback.

References

1. Canetti, R., Herzog, J.: Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 380–403. Springer, Heidelberg (2006)
2. Dolev, D., Yao, A.C.-C.: On the security of public key protocols (extended abstract). In: FOCS, pp. 350–357 (1981)
3. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). In: Watanabe, O., Hagiya, M., Ito, T., van Leeuwen, J., Mosses, P.D. (eds.) TCS 2000. LNCS, vol. 1872, pp. 3–22. Springer, Heidelberg (2000)
4. Abadi, M., Jürjens, J.: Formal eavesdropping and its computational interpretation. In: Kobayashi, N., Pierce, B.C. (eds.) TACS 2001. LNCS, vol. 2215, pp. 82–94. Springer, Heidelberg (2001)
5. Horvitz, O., Gligor, V.D.: Weak key authenticity and the computational completeness of formal encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 530–547. Springer, Heidelberg (2003)
6. Micciancio, D., Warinschi, B.: Completeness theorems for the abadi-rogaway language of encrypted expressions. Journal of Computer Security 12(1), 99–130 (2004)
7. Micciancio, D., Warinschi, B.: Soundness of formal encryption in the presence of active adversaries. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 133–151. Springer, Heidelberg (2004)
8. Janvier, R., Lakhnech, Y., Mazaré, L.: Completing the picture: Soundness of formal encryption in the presence of active adversaries. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 172–185. Springer, Heidelberg (2005)
9. Kremer, S., Mazaré, L.: Adaptive soundness of static equivalence. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 610–625. Springer, Heidelberg (2007)
10. Comon-Lundh, H., Cortier, V.: Computational soundness of observational equivalence. Technical report, INRIA (2008)
11. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
12. Goubault-Larrecq, J., Palamidessi, C., Troina, A.: A probabilistic applied picalculus. In: Shao, Z. (ed.) APLAS 2007. LNCS, vol. 4807, pp. 175–190. Springer, Heidelberg (2007)
13. Backes, M., Pfitzmann, B., Waidner, M.: A composable cryptographic library with nested operations. In: CCS, pp. 220–230 (2003)
14. Backes, M., Pfitzmann, B.: Symmetric encryption in a simulatable dolev-yao style cryptographic library. In: CSFW, pp. 204–218 (2004)

15. Backes, M., Pfitzmann, B., Waidner, M.: Symmetric authentication in a simulatable dolev-yao-style cryptographic library. Int. J. Inf. Sec. 4, 135–154 (2005)
16. Backes, M., Pfitzmann, B.: A cryptographically sound security proof of the needham-schroeder-lowe public-key protocol. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 1–12. Springer, Heidelberg (2003)
17. Backes, M., Dürmuth, M.: A cryptographically sound dolev-yao style security proof of an electronic payment system. In: CSFW, pp. 78–93 (2005)
18. Backes, M.: A cryptographically sound dolev-yao style security proof of the otway-rees protocol. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 89–108. Springer, Heidelberg (2004)
19. Patil, A.: On symbolic analysis of cryptographic protocols. Master's thesis, Massachusetts Institute of Technology (2006)
20. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: POPL, pp. 104–115 (2001)
21. Abadi, M., Cortier, V.: Deciding knowledge in security protocols under equational theories. Theor. Comput. Sci. 367(1–2), 2–32 (2006)

A Translation Rules for Symbolic Interpretation

- Translating internal computations:
  - $symb(\texttt{newnonce}()) = \nu N.\nu x.(\{N/x\}|\_)$
  - $symb(\texttt{encrypt}(m, \texttt{A})) = \nu x.(\{\texttt{enc}(m, PK_A)/x\}|\_)$
  - $symb(\texttt{decrypt}(c)) = \nu x.(\{\texttt{dec}(m, SK_A)/x\}|\_)$
  - $symb(\texttt{sign}(m)) = \nu x.(\{\texttt{sign}(m, SigK_A)/x\}|\_)$
  - $symb(\texttt{verify}(m, s, \texttt{A})) = \nu x.(\{\texttt{ver}(m, s, VerK_A)/x\}|\_)$
  - $symb(\texttt{pair}(m_1, m_2)) = \nu x.(\{\texttt{pair}(m_1, m_2)/x\}|\_)$
  - $symb(\texttt{fst}(m)) = \nu x.(\{\texttt{fst}(m)/x\}|\_)$
  - $symb(\texttt{snd}(m)) = \nu x.(\{\texttt{snd}(m)/x\}|\_)$
- Translating external interactions:
  - $symb(\texttt{input}(x)) = input_{\texttt{R}}(x).(\_)$
  - $symb(\texttt{output}(m)) = \overline{output_{\texttt{R}}}\langle m\rangle.(\_)$
  - $symb(\texttt{send}(m)) = \overline{send_{\texttt{R}}}\langle m\rangle.(\_)$
  - $symb(\texttt{recv}(x)) = recv_{\texttt{R}}(x).(\_)$
  - $symb(\texttt{send}^{\texttt{a}}(m, \texttt{A})) = \overline{send_{\texttt{RA}}}\langle m\rangle.(\_)$
  - $symb(\texttt{recv}^{\texttt{a}}(x, \texttt{A})) = recv_{\texttt{AR}}(x).(\_)$
  - $symb(\texttt{coin}^{\texttt{i}}(b, \texttt{A})) = \overline{toss^i_{\texttt{RA}}}\langle\rangle.toss^i_{\texttt{RA}}(b).(\_)$
  - $symb(\texttt{coin}^{\texttt{r}}(b, \texttt{A})) = \overline{toss^r_{\texttt{RA}}}\langle\rangle.toss^r_{\texttt{RA}}(b).(\_)$
- Translating control flow:
  - $symb(\texttt{P}_1;\texttt{P}_2) = symb(\texttt{P}_1)[symb(\texttt{P}_2)]$
  - $symb(\texttt{if } M = N \texttt{ then P}_1 \texttt{ else P}_2)$
    $= \texttt{if } M = N \texttt{ then } symb(\texttt{P}_1) \texttt{ else } symb(\texttt{P}_2)$
  - $symb(\texttt{prob}(\texttt{P}_1;\texttt{P}_2)) = symb(\texttt{P}_1) \oplus_{0.5} symb(\texttt{P}_2)$
- Key distribution:
  - $PK_R[\_] ::= \nu SK_R.\,(\{\texttt{pk}(SK_R)/PK_R\} \mid \_)$
  - $SigK_R[\_] ::= \nu SigK_R.\,(\{\texttt{vk}(SigK_R)/VerK_R\} \mid \_)$
- Finally, $S_{\texttt{R}} ::= SigK_R\,[PK_R\,[symb(\mathcal{P}(\texttt{R}))[\texttt{0}]]]$

Examples For protocol in Figure 1, the corresponding processes are:

$$\begin{aligned}
\mathcal{S}_{\mathtt{A}} ::= {} & \nu N.\nu N_a.(\{N/N_a\}| \\
& \nu x_1.(\{\mathtt{pair}(Na,\mathtt{B})/x_1\}| \\
& \nu m_1.(\{\mathtt{enc}(x_1, PK_B)/m_1\}| \\
& \overline{send_{\mathtt{A}}}\langle m_1\rangle. \\
& recv_{\mathtt{R}}(x). \\
& \mathtt{if}\ x = N_a\ \mathtt{then}\ \overline{send_{\mathtt{A}}}\langle \mathtt{true}\rangle))))
\end{aligned}$$

$$\begin{aligned}
\mathcal{S}_{\mathtt{B}} ::= {} & recv_{\mathtt{B}}(m_1'). \\
& \nu x_1'.(\{\mathtt{dec}(m, SK_B)/x_1'\}| \\
& \nu N.(\{\mathtt{fst}(m_1')/x_1'\}| \\
& \overline{send_{\mathtt{A}}}\langle N\rangle)))
\end{aligned}$$

B Definition of $\mathtt{tr}(\cdot)$ in Section 6

- $\mathtt{tr}\left(\overline{send_{\mathtt{A}}}\langle M\rangle\right) = [\text{“}send\text{”}, \mathtt{A}, M]$
- $\mathtt{tr}\left(\overline{recv_{\mathtt{A}}}(M)\right) = [\text{“}adv\text{-}deliver\text{”}, \mathtt{A}, M]$
- $\mathtt{tr}\left(\overline{adv^{a}_{\mathtt{AB}}}\langle M, c\rangle\right) = [\text{“}auth\text{-}send\text{”}, \mathtt{A}, \mathtt{B}, M]$
- $\mathtt{tr}\left(input_{\mathtt{A}}(M)\right) = [\text{“}input\text{”}, \mathtt{A}, M]$; $\mathtt{tr}\left(\overline{output_{\mathtt{A}}}\langle M\rangle\right) = [\text{“}output\text{”}, \mathtt{A}, M]$
- $\mathtt{tr}\left(\overline{adv^{i}_{\mathtt{AB}}}\langle c, b\rangle\right) = [\text{“}toss\text{-}i\text{”}, \mathtt{A}, \mathtt{B}, b]$; $\mathtt{tr}\left(\overline{adv^{r}_{\mathtt{AB}}}\langle c, b\rangle\right) = [\text{“}toss\text{-}r\text{”}, \mathtt{A}, \mathtt{B}, b]$
- $\mathtt{tr}\left(c(M)\right) = [\text{“}adv\text{-}auth\text{”}, \mathtt{A}, \mathtt{B}, M]$, if c occurs in $\overline{adv^{a}_{\mathtt{AB}}}\langle M, c\rangle$ previously.
- $\mathtt{tr}\left(c()\right) = [\text{“}adv\text{-}tossed\text{-}i\text{”}, \mathtt{A}, \mathtt{B}, b]$, if c occurs in $\overline{adv^{i}_{\mathtt{AB}}}\langle c, b\rangle$ previously.
- $\mathtt{tr}\left(c()\right) = [\text{“}adv\text{-}tossed\text{-}r\text{”}, \mathtt{A}, \mathtt{B}, b]$, if c occurs in $\overline{adv^{r}_{\mathtt{AB}}}\langle c, b\rangle$ previously.

C Rules for Defining Hybrid Traces

- At the beginning, the trace $\mathtt{t}$ is empty.
- If the environment provides m as input to party $\mathtt{R}$, then append trace $\mathtt{t}$ with event $\mathtt{E} = \langle \text{“}input\text{”}, \mathtt{R}, m\rangle$.
- If the adversary is activated:
  - If it delivers a message m to party $\mathtt{R}$, then let $\mathtt{E} = \langle \text{“}adv\text{-}deliver\text{”}, \mathtt{R}, m\rangle$.
  - If it delivers a authenticated message m from party $\mathtt{S}$ to $\mathtt{R}$, then let $\mathtt{E} = \langle \text{“}adv\text{-}auth\text{”}, \mathtt{S}, \mathtt{R}, m\rangle$.
  - If it delivers coin-tossing result b to initiator $\mathtt{S}$ with responder $\mathtt{R}$, then let $\mathtt{E} = \langle \text{“}adv\text{-}tossed\text{-}i\text{”}, \mathtt{S}, \mathtt{R}, b\rangle$
  - If it delivers coin-tossing result b to responder $\mathtt{R}$ with initiator $\mathtt{S}$, then let $\mathtt{E} = \langle \text{“}adv\text{-}tossed\text{-}r\text{”}, \mathtt{S}, \mathtt{R}, b\rangle$
- If party $\mathtt{R}$ is activated:
  - If it outputs to the environment message m, then let $\mathtt{E} = \langle \text{“}output\text{”}, \mathtt{R}, m\rangle$.
  - If its send message m over insecure network, then let $\mathtt{E} = \langle \text{“}send\text{”}, \mathtt{R}, m\rangle$.
- If $\mathcal{F}_{\text{AUTH}}$ is activated, and $\mathtt{S}$ wants to send m to $\mathtt{R}$, then let $\mathtt{E} = \langle \text{“}send\text{-}auth\text{”}, \mathtt{S}, \mathtt{R}, m\rangle$.
- If $\mathcal{F}_{\text{CT}}$ is activated,
  - $\mathtt{S}$ wants to initiate coin-tossing with $\mathtt{R}$, then let $\mathtt{E} = \langle \text{“}toss\text{-}i\text{”}, \mathtt{S}, \mathtt{R}\rangle$.
  - $\mathtt{R}$ acts as the responder with $\mathtt{S}$, then let $\mathtt{E} = \langle \text{“}toss\text{-}r\text{”}, \mathtt{S}, \mathtt{R}\rangle$.
- If $\mathcal{F}_{\text{CPKE}}$ is activated,
  - If $\mathtt{S}$ encrypts with $(\mathtt{Encrypt}, \mathtt{PID}_{\mathtt{R}}, m)$, and $\mathcal{F}_{\text{CPKE}}$ returns c, then let $\mathtt{E} = \langle \text{“}ciphertext\text{”}, \mathtt{PID}_{\mathtt{R}}, m, c\rangle$.
  - If $\mathtt{S}$ decrypts with $(\mathtt{Decrypt}, \mathtt{PID}_{\mathtt{S}}, c)$, and $\mathcal{F}_{\text{CPKE}}$ returns m, then let $\mathtt{E} = \langle \text{“}decrypt\text{”}, \mathtt{PID}_{\mathtt{S}}, c, m\rangle$.
- The case for $\mathcal{F}_{\text{CERT}}$ is similar.

On the Equivalence of Generic Group Models

Tibor Jager and Jörg Schwenk

Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany

Abstract. The *generic group model* (GGM) is a commonly used tool in cryptography, especially in the analysis of fundamental cryptographic problems, such as the complexity of the discrete logarithm problem [1,2,3] or the relationship between breaking RSA and factoring integers [4,5,6]. Moreover, the GGM is frequently used to gain confidence in the security of newly introduced computational problems or cryptosystems [7,8,9,10,11]. The GGM serves basically as an idealization of an abstract algebraic group: An algorithm is restricted to basic group operations, such as computing the group law, checking for equality of elements, and possibly additional operations, without being able to exploit any specific property of a given group representation.

Different models formalizing the notion of generic groups have been proposed in the literature. Although all models aim to capture the same notion, it is not obvious that a security proof in one model implies security in the other model. Thus the validity of a proven statement may depend on the choice of the model. In this paper we prove the equivalence of the models proposed by Shoup [2] and Maurer [3].

1 Introduction

The security of asymmetric cryptographic systems depends on the intractability of certain computational problems. Many widely used cryptographic assumptions, such as the Diffie-Hellman assumption [12] and the RSA assumption [13], are unproven in a general model of computation (e.g. the Turing machine model). Therefore more restricted models have been considered. One such restricted model is the *generic group model* (GGM). In this model a class of algorithms is considered that run on an algebraic group, without exploiting specific properties of a given group representation. This implies that the algorithm works for all representation of the group in a similar way. Well-known examples of generic group algorithms for computing discrete logarithms are Shank's Baby-Step Giant-Step algorithm [14] and the Pollard's Rho algorithm [15]. In contrast, index calculus algorithms [16] for the multiplicative group $\mathbb{Z}_p^*$ of integers modulo p, for instance, use the fact that group elements are represented as integers that have a prime factorization, and that integer factorization is compatible with the group operation. There are groups, such as certain elliptic curve groups over finite fields, where essentially no better algorithms that generic algorithms are known. Thus, in this sense the GGM can be seen as an adequate idealization of this class of groups.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 200–209, 2008.

The GGM has been used frequently to gain confidence in newly introduced cryptographic assumptions [10,11] or to prove the security of cryptosystems [7,8,9]. On the one hand, one may argue that a proof in the generic group model yields no gain for practical security, since in practise always a concrete representation of group elements is given. There are even examples of (contrived) schemes which are provably secure in the GGM, but insecure when the generic group is instantiated by a concrete group (with given representation of elements) [17,18]. However, when used in a right manner the GGM can be seen as a valuable tool. Very useful applications with high relevance for cryptography are, for instance, the analysis of the *intrinsic* hardness of computational problems such as the discrete logarithm or the Diffie-Hellman problem [2,3] or root extraction in groups and rings [4,5,6]. The GGM serves also as a method to gain confidence in newly introduced problems, such as [10,11], thus providing evidence that the new problem may indeed be of cryptographic interest. Moreover, the GGM is especially interesting when the relationship between problems is studied, e.g. see [19,20,21,22]. For instance, a reduction from computing discrete logarithms to solving the Diffie-Hellman problem in the GGM would imply the equivalence of both problems in all groups.

Different abstract models of computation to formalize the notion of generic group algorithms have been proposed in the literature [1,2,3]. While the work of Nechaev [1] is targeted solely to the discrete logarithm problem, and thus (to our best knowledge) no further application of the model has shown up in the literature, the models proposed by Shoup [2] and Maurer [3] are more flexible. In this work we consider the models presented in [2] and [3]. Derivatives, especially of [2], have been adapted repeatedly in the literature, e.g. [4,5,10,11,21,22].

Although all models aim to capture the same notion, it is not obvious that a security proof in one model implies security in the other model.Thus the validity of a proven statement may depend on the choice of the model. This situation is not satisfactory, since a cryptographer needs to decide on using one model and then has to worry about whether solely the choice of the model has influenced the validity of the result. Alternatively, all results (including existing results) have to be proven in both models, which is not satisfactory as well. Thus it is important to consider the relationship between both models in order to be able to interpretate proofs in the GGM. An equivalence proof of two coexisting models is especially interesting, since this enables the prover to choose the adequate model for the desired result, without the need of worrying about whether the validity of the proof depends on the choice of the model. In this paper we prove the equivalence of the two GGM models proposed in [2] and [3].

1.1 Related Work

The generic group model was introduced by Nechaev [1].Nechaev used a proof technique based on the theory of graphs to prove an exponential lower bound on the time complexity of generic algorithms for the discrete logarithm problem. Later, Shoup used a more general and more flexible generic model to re-prove the generic hardness of the discrete logarithm problem and, in addition, the

Diffie-Hellman problem [2]. As noted by Shoup, a proof in the model described in [2] implies that the proven result holds in the model of Nechaev [1] as well. Another variant of the generic group model was proposed by Maurer [3], who augmented the results of Shoup, for instance by considering the hardness of the discrete logarithm problem in presence of a decisional Diffie-Hellman oracle.

The GGM is commonly used to give evidence that well-known or newly introduced cryptographic assumptions are indeed valid. Dent [10] has proven that the Diffie-Hellman knowledge problem is secure in the GGM. Boneh and Boyen [11] extended the Shoup model to prove the hardness of the SDH assumption in groups G_1, G_2, G_T where there exists a bilinear pairing map $e : G_1 \times G_2 \rightarrow G_T$. Leander and Rupp [5] adapted the GGM to rings to prove the generic equivalence of breaking the strong low-exponent RSA assumption and factoring integers. Later Altmann et al. [22] augmented these techniques to show that solving the black-box ring extraction problem is at least as hard as factoring integers. Moreover, the GGM served as a tool to prove the security of cryptosystems with high practical relevance, such as signed ElGamal [7], ECIES [8], and ECDSA [9].

Fischlin [17] and Dent [18] shed some criticism on the GGM by constructing computational problems and cryptosystems which are provably secure in the GGM, but insecure whenever the generic group is replaced with a concrete group. Koblitz and Menezes note in [23] that the constructions considered in [17] and [18] are contrived and violate standard cryptographic practise, thus these results can be seen as reminders that a proof in the generic group model should be viewed with the same caution as security proofs in the random oracle model, rather than rendering the GGM approach useless.

1.2 Our Contribution

In this paper we consider the generic group models proposed by Shoup [2] and Maurer [3], and prove the equivalence of both models. More precisely, we show for if for some computational problem π there exists an efficient algorithm in model A, then there exists an efficient algorithm for π in model B. This implies the validity of upper complexity bounds as well as lower complexity bounds on generic algorithms in both models, when proven in one model.

2 Generic Group Models

There are two basic requirements when the notion of generic group algorithms is modeled formally:

1. The algorithm which intends to solve a given problem instance must not be able to exploit any property of a given representation of group elements, i.e. the group representation must be hidden.
2. Nevertheless, the algorithm must be able to perform computations on group elements. That is, the algorithm must (at least) be able to perform the group operation and check for equality of elements *without* knowing a concrete representation of the group elements.

When considering the models exactly as described in [2] and [3] we see that both are actually not comparable. The reason for this is that Maurer has formulated the model in [3] in high generality, allowing the model to be specified for the computation of arbitrary functions and relations on group elements, while Shoup has modeled only a few operations and relations explicitly (i.e., performing the group operation and checking for equality of elements) without modeling the possibility of extending the model by additional operations. In the following section we describe the Shoup model in a more general way, thus slightly deviating from the original description, but still using the basic concept of [2].

2.1 Shoup's Generic Group Model

The generic group model described in this section was proposed by Shoup in [2]. The original model allows only for computing the group operation and checking for equality of elements. Several extensions of this model have been used in the literature, e.g. [5,22]. Subsequently we will describe a simple generalization of the original model from [2] in order to capture the possibility of extending the model by additional operations, and to make the Shoup model comparable to the Maurer model [3].

The construction is motivated by the fact that the elements of a group must be represented in some way in order to be able to perform computations on group elements. A general way of representing elements, e.g. in a computer, are bitstrings. Thus, a representation of a group can be seen as a bijective map from the group to the set of bistrings without loss of generality. Shoup has based his model on this notion in the following way:

Let $(G, \circ)$ be a group of order n and let $S_n \subseteq \{0,1\}^{\lceil \log |G| \rceil}$ be a set of n different bit strings. Let

$$\sigma : G \to S_n$$

be a bijective *encoding function*, chosen at random among all possible functions, which encodes group elements as random, but unique binary strings. The random encoding ensures that the group G has only the defined properties of an abstract group.

In order to be able perform computations on randomly encoded group elements we assume an oracle $\mathcal{O}$ that computes operations from some operation set Π on bit strings representing group elements. A typical operation, for instance, would be the binary function computing the group law on encoded elements, i.e. $\Pi = \{\circ\}$. Moreover, the oracle may compute relations from some relation set Σ on encoded elements. For instance, a query to a decisional Diffie-Hellman oracle can be modeled as a relation. The equality relation is always included in the relation set implicitly, since the bijectivity of the encoding function allows to check for equality of elements by checking for equality of encodings.

Definition 1 (Generic Group Algorithm in Shoup's Model). *A* generic group algorithm $\mathcal{A}$ *is a (possibly probabilistic) algorithm which takes as input a* r*-tuple of encoded group elements* $(\sigma(x_1), \ldots, \sigma(x_r))$*,* $x_i \in G$*,* $1 \leq i \leq r$*. The oracle may query a* generic group oracle $\mathcal{O}$ *to perform computation operations in* Π *and relations in* Σ *on encoded group elements.*

Thus, the algorithm receives a r-tuple of encoded group elements as input, and may query the oracle $\mathcal{O}$ to perform computations on randomly encoded elements. The input to a generic group algorithm is usually the public part of a problem instance that should be solved by the algorithm.

An algorithm $\mathcal{A}$ makes a computation query to the oracle $\mathcal{O}$ by specifying a t-ary operation $f \in \Pi$ and t encoded group elements $\sigma(x_1), \ldots, \sigma(x_t)$. The oracle $\mathcal{O}$ computes $z = f(x_1, \ldots, x_t)$ and returns $\sigma(z)$. $\mathcal{A}$ makes a relation query to $\mathcal{O}$ by specifying t encoded group elements $\sigma(x_1), \ldots, \sigma(x_t)$ and a t-ary relation $\rho \in \Sigma$. $\mathcal{O}$ returns $\rho(x_1, \ldots, x_t)$.

It is assumed that $\mathcal{A}$ has unbounded memory capacity, the time complexity of $\mathcal{A}$ is measured by the number of oracle queries.

Example 1. The discrete logarithm problem in a cyclic group $(G, \circ)$ of order n in Shoup's generic group model can be stated as follows: Given a primitive element $\sigma(1)$, an encoded group element $\sigma(x)$ with $x \in_R \mathbb{Z}_n$, and the group order n, determine x. The oracle $\mathcal{O}$ computes the group law, i.e. $\Pi := \{\circ\}$ and $\Sigma := \{=\}$.

Example 2. The Diffie-Hellman problem in a cyclic group $(G, \circ)$ of order n in presence of a decisional Diffie-Hellman oracle (gap Diffie-Hellman problem) can be stated as: Given a primitive element $\sigma(1)$, encoded group elements $\sigma(x)$ and $\sigma(y)$ with $x, y \in_R \mathbb{Z}_n$, and the group order n, compute $\sigma(z)$, where $z \equiv xy \bmod n$. The oracle $\mathcal{O}$ computes the group law and answers decisional Diffie-Hellman queries, i.e. $\Pi := \{\circ\}$ and $\Sigma := \{=, DDH(\cdot,\cdot,\cdot)\}$ with $DDH(\sigma(x), \sigma(y), \sigma(z)) = 1$ if $z \equiv xy \bmod n$ and $DDH(\sigma(x), \sigma(y), \sigma(z)) = 0$ if $z \not\equiv xy \bmod n$.

2.2 Maurer's Generic Group Model

The generic group model described in this section was proposed by Maurer in [3]. The model is characterized by an oracle $\mathcal{O}$ that maintains an internal list $L \subseteq G$, where $(G, \circ)$ is a group. Let L_i denote the i-th element of the list. The list L is initialized with r values $L_1, \ldots, L_r$ corresponding to the given problem instance.

The algorithm $\mathcal{A}$ receives a set of pointers to the list elements L_i for $i \in \{1, \ldots, r\}$ as input and may query the oracle for two types of operations:

1. Computation operations, for instance application of the group law on group elements stored in the list L.
2. Queries on the internal state, i.e., queries for relations on stored elements (e.g., the equality relation).

Definition 2 (Generic Group Algorithm in Maurer's Model). *A generic algorithm is a (possibly probabilistic) algorithm $\mathcal{A}$ which takes as input r indices $(1, ..., r)$ pointing to list elements $L_1, ..., L_r$. The algorithm may query the oracle $\mathcal{O}$ for computation operations on internal state variables from Π, and queries on the internal state from a relation set Σ.*

The algorithm queries a computation operation by selecting a t-ary operation $f \in \Pi$ as well as $t+1$ indices $i_1, ..., i_{t+1}$. The oracle computes $f(L_{i_1}, ..., L_{i_t})$ and

stores the result in the variable $L_{i_{t+1}}$. To query a relation the algorithm selects a t-ary relation $\rho \in \Sigma$ as well as t indices $i_1, ..., i_t$. The query is replied by $\mathcal{O}$ with $\rho(L_{i_1}, ..., L_{i_t})$. We assume that the equality relation "=" is always included in the relation set Σ. The complexity of an algorithm is measured by the number of oracle queries.

Example 3. The discrete logarithm problem in a cyclic group $(G, \circ)$ of order n in Maurer's generic group model can be stated as follows: Let $L_1 := 1$, $L_2 := x$ with $x \in_R \mathbb{Z}_n$, determine x. The oracle $\mathcal{O}$ computes only the group law and the equality relation, i.e. $\Pi := \{\circ\}$ and $\Sigma := \{=\}$.

Example 4. The Diffie-Hellman problem in a cyclic group $(G, \circ)$ of order n in presence of a decisional Diffie-Hellman oracle can be stated as: Let $L_1 := 1$, $L_2 := x$ and $L_3 := y$ with $x, y \in_R \mathbb{Z}_n$, output i such that $L_i \equiv xy \bmod n$. The oracle $\mathcal{O}$ computes the group law and the equality relation, and provides a decisional Diffie-Hellman oracle, i.e. $\Pi := \{\circ\}$ and $\Sigma := \{=, DDH(\cdot,\cdot,\cdot)\}$, where $DDH(x, y, z) = 1$ if $z \equiv xy \bmod n$ and $DDH(x, y, z) = 0$ if $z \not\equiv xy \bmod n$.

3 The Equivalence of Generic Group Models

In this section we will show that both models described above may be utilized equivalently. Throughout this section let $\mathcal{A}_S$ and $\mathcal{O}_S$ be an algorithm and an oracle according to the description of Shoup's model in Section 2.1, and $\mathcal{A}_M$ and $\mathcal{O}_M$ be an algorithm and an oracle according to Maurer's model as described in Section 2.2. We suppose that both oracles provide identical operation sets Π and Σ, and that both oracles use isomorphic groups for the internal representation of elements.

We construct polynomial-time interfaces $\mathcal{I}_{M2S}$ and $\mathcal{I}_{S2M}$ converting the input and output of $\mathcal{O}_S$ to the input and output of $\mathcal{O}_M$, and vice versa, such that $\mathcal{A}_S$ can interact with $\mathcal{O}_M$ using $\mathcal{I}_{S2M}$, and $\mathcal{A}_M$ can interact with $\mathcal{O}_S$ using $\mathcal{I}_{M2S}$. This implies

1. **Equivalence of upper complexity bounds:** If there exists an algorithm running running in polynomial time in some parameter κ (i.e. performing at most $m = poly(\kappa)$ oracle queries) in model A, then there exists an algorithm running in model B in time polynomial in κ.
2. **Equivalence of lower complexity bounds:** Suppose a lower complexity bound (e.g. exponential/subexponential in some parameter κ) has been proven in one model. Then this implies a lower bound in the other model that differs at most by some factor polynomial in κ. This can easily be proven by contradiction. For instance, suppose there exists a exponential-time lower complexity bound in model A and a polynomial-time algorithm in model B. Since a polynomial-time algorithm in model B implies a polynomial-time algorithm in model A this leads to a contradiction.

3.1 From Maurer's GGM to Shoup's GGM

At first we show that there exists an efficient interface $\mathcal{I}_{M2S}$ that has access to $\mathcal{O}_S$ and simulates the oracle $\mathcal{O}_M$ efficiently.

Theorem 1. *Suppose there exists an algorithm $\mathcal{A}_M$ that makes at most m queries to $\mathcal{O}_M$ and has success probability α. Then there exists an algorithm $\mathcal{A}_S$ performing at most m queries to an oracle $\mathcal{O}_S$ having success probability α.*

Proof. We construct an interface $\mathcal{I}_{M2S}$ that internally uses an $\mathcal{O}_S$ for the computation of the operations in Π and relations in Σ, and provides an input and output interface that is equivalent to an $\mathcal{O}_M$ which provides access to the same group as $\mathcal{O}_S$ and the same operation set Π and relation set Σ. $\mathcal{I}_{M2S}$ uses only operations that can be performed by an algorithm in Shoup's model. Thus we can regard the combination of $\mathcal{A}_M$ and $\mathcal{I}_{M2S}$ as an algorithm $\mathcal{A}_S$ compliant to the Shoup model.

$\mathcal{I}_{M2S}$ maintains a list $L \subseteq S_n$, where S_n is the set of encoding bit strings used by $\mathcal{O}_S$. At the beginning of the game the interface receives a r-tuple $(\sigma(x_1), \ldots, \sigma(x_r))$ of encoded elements from $\mathcal{O}_S$. The interface stores the encodings in the list, such that $L_i = \sigma(x_i)$ for $1 \leq i \leq r$. The algorithm $\mathcal{A}_M$ receives the indices $1, \ldots, r$ as input.

The interface now answers the oracle queries of $\mathcal{A}_M$ in the following way:

1. Whenever the algorithm specifies indices $i_1, \ldots, i_{t+1}$ and a t-ary operation $f \in \Pi$ to query a computation, the interface determines the respective encodings by list lookup and queries the oracle to perform the computation $f(L_{i_1}, \ldots, L_{i_t})$. The result is stored at $L_{i_{t+1}}$.
2. When the algorithm queries a t-ary relation $\rho \in \Sigma$ by specifying t list indices $i_1, \ldots, i_t$, the interface determines the respective encodings by list lookup and queries the oracle $\mathcal{O}_S$ for the relation $\rho(L_{i_1}, \ldots, L_{i_t})$. The reply of $\mathcal{O}_S$ is forwarded to $\mathcal{A}_M$.

It is easy to see that the view of $\mathcal{A}_M$ when interacting with $\mathcal{I}_{M2S}$ is identical to the view when interacting with a real oracle $\mathcal{O}_M$. For each query of an operation $f \in \Pi$ and for each query of a relation $\rho \in \Sigma$ made by $\mathcal{A}_M$ the interface performs one oracle query to $\mathcal{O}_S$. Since $\mathcal{I}_{M2S}$ provides a perfect simulation of $\mathcal{O}_M$, the success probability of $\mathcal{A}_M$ is equal when interacting either with $\mathcal{O}_M$ or with $\mathcal{I}_{M2S}$. □

3.2 From Shoup's GGM to Maurer's GGM

Now consider the case where the interface has access to an oracle $\mathcal{O}_M$ and simulates an oracle $\mathcal{O}_S$. We show that there exists an efficient interface $\mathcal{I}_{S2M}$ that has access to $\mathcal{O}_M$ and simulates the oracle $\mathcal{O}_S$ efficiently.

Theorem 2. *Suppose there exists an algorithm $\mathcal{A}_S$ making at most m queries to oracle $\mathcal{O}_S$ and having success probability α. Then there exists an algorithm $\mathcal{A}_M$ having success probability α and performing at most $m^2 + r^2$ queries to an oracle $\mathcal{O}_M$, where r is the number of group elements that $\mathcal{A}_S$ receives as input.*

Proof. We construct an interface $\mathcal{I}_{S2M}$ that internally uses an $\mathcal{O}_M$ for the computation of the operations in Π and relations in Σ, and provides an input and output interface that is equivalent to an $\mathcal{O}_S$. Now $\mathcal{I}_{S2M}$ uses only operations that can be performed by an algorithm in Maurer's model. Thus we can regard the combination of $\mathcal{A}_S$ and $\mathcal{I}_{S2M}$ as an algorithm $\mathcal{A}_M$ compliant to the Shoup model. $\mathcal{I}_{S2M}$ maintains a list $E \subseteq \{0,1\}^{\lceil \log |G| \rceil}$. The list internally used by $\mathcal{O}_M$ is denoted L. We denote the i-th element in E and L with E_i and L_i, respectively.

At the beginning of the game the interface receives a r-tuple of indices from $\mathcal{O}_M$. Now the interface queries the oracle $\mathcal{O}_M$ for the equality relation on all pairs of received indices. More precisely, for i from 1 to r the oracle proceeds as follows:

1. The oracle checks whether there exists $j \in \{1, \ldots, r\}$ such that $j < i$ and $L_j = L_i$. If this is true, the oracle sets $E_i := E_j$.
2. Otherwise a new random encoding $E_i \in_R \{0,1\}^{\lceil \log |G| \rceil} \backslash E$ is assigned to i.

Note that this procedure can be performed by querying $\mathcal{O}_M$ for the equality relation at most $r(r-1)/2 < r^2$ times.

The interface answers the queries of $\mathcal{A}_S$ in the following way:

1. Whenever the algorithm queries the computation of a t-ary operation $f \in \Pi$ by submitting t encodings $E_{i_1}, \ldots, E_{i_t}$, the oracle performs the following steps:
   (a) The interface receives $E_1, \ldots, E_t$ as input and determines the respective indices $i_1, \ldots, i_t$ by list lookup. Note that the same group element may appear multiple times in L, and thus a group element may have multiple equivalent indices pointing to the same group element. The interface simply takes the first index that is found.
   (b) Then the interface queries the oracle $\mathcal{O}_M$ for the computation $L_{|L|+1} = f(i_1, \ldots, i_t)$, where $|L|$ denotes the length of the list L. The result is stored in the next free list element $L_{|L|+1}$ of $\mathcal{O}_M$.
   (c) For each index $1 \leq i \leq |L|$ the interface queries the oracle whether $L_i = L_{|L|+1}$. If this equality holds for some L_i, then the respective encoding is appended to the list again: $E_{|L|+1} := E_i$. The algorithm receives $E_{|L|+1}$ as reply.
   (d) Otherwise a new encoding $E_{|L|+1} \in_R \{0,1\}^{\lceil \log |G| \rceil} \backslash E$ is chosen at random. The algorithm receives $E_{|L|+1}$ as reply.
2. Whenever the algorithm queries a t-ary relation $\rho \in \Sigma$ by submitting t encodings $E_{i_1}, \ldots, E_{i_t}$, the interface determines the respective indices by list lookup and queries the oracle $\mathcal{O}_M$ for the relation $\rho(i_1, \ldots, i_t)$. The reply of $\mathcal{O}_M$ is forwarded to $\mathcal{A}_S$.

Note that the time complexity of an interface simulating $\mathcal{O}_S$ is larger than the complexity of an interface which simulates $\mathcal{O}_M$. This is due to the fact that the interface has to check for equality of elements after each computation operation. Suppose that an algorithm queries at most m queries. After the i-th computation operation the interface has to query the equality relation at most $i - 1$ times

in order determine equal elements. There are at most m elements in the list, therefore the total number of equality checks is bounded by $m(m-1)/2 < m^2$. Note that $\mathcal{I}_{S2M}$ simulates a real $\mathcal{O}_S$ oracle perfectly, thus the success probability of $\mathcal{A}_S$ is equal when interacting either with $\mathcal{O}_S$ or with $\mathcal{I}_{S2M}$. □

4 Conclusions

We have considered the two generic group models proposed in [2] and [3] and have proven that both models are equivalent. The proof itself is quite straightforward, especially after having given easily comparable descriptions of both models in Sections 2.1 and 2.2. Hence the main contribution of this paper shall be merely seen as evidence that a cryptographer intending to prove a statement in the generic group model may choose the model that suits the used proving technique and the desired result best, without worrying about whether the validity of the proof depends on the choice of the model.

Moreover, though the model of Maurer has not yet received as much attention as the Shoup model, we believe that the way of formalizing the notion of generic groups used by Maurer in [3] is sometimes slightly advantageous in the sense that the prover has to deal with less technical details. To give a short example, a typical proving technique in the GGM introduces a simulation game where group elements are replaced with polynomials which are implicitly evaluated with some group elements that correspond to the given problem instance, e.g. see [2,3,5]. Now it may happen that several polynomials correspond to the same group element. In Shoup's model an algorithm submits random encodings corresponding to some group elements in order to query a computation or relation. However, since one group element may correspond to several polynomials, it is not clear to which polynomials the queried computation operation should be applied. Thus, the prover has to deal with this problem by describing a procedure that determines to which polynomial the queried operation should be applied (though it usually suffices to simply take the first polynomial with matching encoding). In contrast, in Maurer's model a prover does not have to deal with this, since the algorithm submits list indices, each corresponding to unique polynomials.

References

1. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes 55(2), 165–172 (1994)
2. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
3. Maurer, U.M.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)
4. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 256–271. Springer, Heidelberg (2002)

5. Leander, G., Rupp, A.: On the equivalence of RSA and factoring regarding generic ring algorithms. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 241–251. Springer, Heidelberg (2006)
6. Aggarwal, D., Maurer, U.: Factoring is equivalent to generic RSA. Cryptology ePrint Archive, Report 2008/260 (2008), http://eprint.iacr.org/
7. Schnorr, C.P., Jakobsson, M.: Security of signed elgamal encryption. In: [24], pp. 73–89
8. Smart, N.P.: The exact security of ECIES in the generic group model. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 73–84. Springer, Heidelberg (2001)
9. Brown, D.R.L.: Generic groups, collision resistance, and ECDSA. Des. Codes Cryptography 35(1), 119–152 (2005)
10. Dent, A.W.: The hardness of the DHK problem in the generic group model. Cryptology ePrint Archive, Report 2006/156 (2006), http://eprint.iacr.org/.
11. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)
12. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
13. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21, 120–126 (1978)
14. Shanks, D.: Class number, a theory of factorization, and genera. In: Lewis, D.J. (ed.) 1969 Number Theory Institute. Proceedings of Symposia in Pure Mathematics, Providence, Rhode Island, vol. 20, pp. 415–440. American Mathematical Society (1971)
15. Pollard, J.M.: A Monte Carlo method for factorization. BIT 15, 331–334 (1975)
16. Odlyzko, A.M.: Discrete logarithms in finite fields and their cryptographic significance. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 224–314. Springer, Heidelberg (1985)
17. Fischlin, M.: A note on security proofs in the generic model. In: [24] pp. 458–469
18. Dent, A.W.: Adapting the weaknesses of the random oracle model to the generic group model. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 100–109. Springer, Heidelberg (2002)
19. Maurer, U.M.: Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 271–281. Springer, Heidelberg (1994)
20. Boneh, D., Lipton, R.J.: Algorithms for black-box fields and their application to cryptography (extended abstract). In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 283–297. Springer, Heidelberg (1996)
21. Maurer, U.M., Wolf, S.: Lower bounds on generic algorithms in groups. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 72–84. Springer, Heidelberg (1998)
22. Altmann, K., Jager, T., Rupp, A.: On black-box ring extraction and integer factorization. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 437–448. Springer, Heidelberg (2008)
23. Koblitz, N., Menezes, A.: Another look at generic groups. Advances in Mathematics of Communications 1, 13–28 (2007)
24. Okamoto, T. (ed.): ASIACRYPT 2000. LNCS, vol. 1976. Springer, Heidelberg (2000)

The Analysis of an Efficient and Provably Secure ID-Based Threshold Signcryption Scheme and Its Secure Version*

ZhenChao Zhu[1], Yuqing Zhang[2,**], and Fengjiao Wang[2]

[1] Key Lab of Computer Networks and Information Security of Ministry of Education, Xidian University, Xi'an, 710071, P.R. China
[2] National Computer Network Intrusion Protection Center, GUCAS, Beijing 100049, P.R. China
zhangyq@gucas.ac.cn

Abstract. In this paper,we analyze an identity-based threshold signcryption (IDTSC)scheme proposed in ICCCAS'2008, although Li and Yu pointed out that the scheme is the first provably secure scheme which is secure against adaptive chosen ciphertext attacks and secure in the sense of unforgeability, we show that the signcryption in the scheme is easily forged by the appointed clerk who is one of the members , the clerk can impersonate the members to forge valid signcryption to any receiver, then we give a secure version which we prove its confidentiality under the Decisional Bilinear Diffie-Hellman assumption and its unforgeability under the Computational Diffie-Hellman assumption in the random oracle model. Scheme turns out to be more efficient than the previously proposed schemes.

Keywords: Identity based cryptography, signcryption , provably secure.

1 Introduction

In 1984, Shamir introduced identity-based cryptosystems [1]. The idea was to allow using users' identity information serve as public key. These systems have a trusted private key generator (PKG) whose task is to initialize the system and generate master/public key pair. Given the master secret key, the PKG can compute users' private key from their identity information. In such schemes, the key management procedures are simplified. Since then, several practical identity-based signature schemes have been proposed, but a satisfying identity-based encryption scheme [2] from bilinear maps was first introduced untill 2001.

Signcryption is a new paradigm in public key cryptography, which was first proposed by Zheng in 1997 [3]. It can simultaneously fulfill both the functions of digital signature and public key encryption in a logically single step, and with a cost

* This work is supported in part by The National Natural Science Foundation of China (60573048, 60773135, 90718007); The High Technology Research and Development Program of China (863 Program) (2007AA01Z427, 2007AA01Z450).
** Corresponding author.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 210–225, 2008.

significantly lower than that required by the traditional signature then-encryption approach. Since then, several efficient signcryption schemes were proposed. Malone-Lee [4] proposed an identity-based signcryption scheme that can offer non-repudiation. Libert and Quisquater [5] proposed an identity–based signcryption scheme that satisfies semantic security. Combining the threshold technique with identity-based signcryption approach, Duan et al. [6] devised an identity-based threshold signcryption scheme. However, Duan et al.'s scheme still adopts the method that the master key is shared among n parties, and non-repudiation and semantic security cannot be provided. In 2005, Peng and Li [7] proposed an ID-based threshold signcryption scheme based on Libert and Quisquater's ID-based signcryption scheme [8]. However, Peng and Li' scheme [7] does not provide the forward security, that is, anyone who obtains the sender's private key can recover the original message of a signcrypted text. In addition, both Duan et al.'s scheme [6] and Peng and Li's scheme [7] do not consider the formal models and security proofs. Ma et al. [9] also proposed a threshold signcryption scheme using the bilinear pairings. However, Ma et al.'s scheme [9] is not ID-based. In 2008, Li and Yu first gives a provably secure ID-based threshold signcryption scheme [10] based on the bilinear pairings and proves its confidentiality under the Decisional Bilinear Diffie-Hellman assumption and its unforgeability under the Computational Diffie-Hellman assumption in the random oracle model.

Our contributions: In this paper, we point out that LY scheme [10] is not secure for the signcryption can be easily forged by the appointed clerk, who is one of the members and the clerk can impersonate the members to forge valid signcryption to any receiver, then we give a secure version which satisfies confidentiality under the DBDHP assumption and unforgeability under the CDHP assumption in the random oracle model. Our scheme also turns out to be more efficient than the previously proposed schemes.

Roadmap: The paper is organized as follows: In section 2 we give some mathematical background which will be used in the paper, the formal model and security notions of ID-based threshold signcryption scheme will be given in the section 3, LY scheme[10] will be presented and discussed in section 4, then we present our secure scheme in section 5, the security of the proposed scheme is proved under the DBDH and CDHP assumptions and the efficiency is discussed in this section too, section 6 concludes the paper.

2 Preliminaries

2.1 Bilinear Pairings

Bilinear pairing is an important primitive for many cryptographic schemes. In this section, we briefly review some preliminaries that will be used throughout this paper.

Let G_1 be an additive group of prime order q, generated by p, and let G_2 be a multiplicative group with the same order q. We assume that there is a bilinear map e from $G_1 \times G_1 \to G_2$ with the following properties:

(1) Bilinearity: Which means given elements $A_1, A_2, A_3 \in G_1$, we have $e(A_1 + A_2, A_3) = e(A_1, A_3) \cdot e(A_2, A_3)$, $e(A_1, A_2 + A_3) = e(A_1, A_2) \cdot e(A_1, A_3)$;

(2) Non-degeneracy: There exists $A_1, A_2 \in G_1$ such that $e(A_1, A_2) \neq 1_{G_2}$;

(3) Computability: Which means that there exists an efficient algorithm to compute $e(A_1, A_2)$, $\forall A_1, A_2 \in G_1$.

Such pairings can be derived from Weil or Tate pairings on an elliptic curve over finite field.

2.2 Related Complexity Assumptions

We consider the following problems in the group G_1 of prime order *q*, generated by p.

Definition 1. The Decisional Bilinear Diffie-Hellman problem (DBDHP) is, given a generator p of a group G, a tuple (aP, bP, cP) and an element $h \in G_2$, to decide whether $h = e(P, P)^{abc}$.

Definition 2. Given a generator p of a group G and a tuple (aP, bP), the Computational Diffie-Hellman problem (CDHP) is to compute abP.

Definition 3. The modified Generalized Bilinear Inversion (mGBI) problem: Given $h \in G_2$ and $P \in G_1$, computes $S \in G_1$ such that $h = e(P, S)$.

3 Formal Model and Security Notions of IDTSC Scheme

3.1 Generic Scheme

A generic ID-based threshold signcryption scheme consists of the following five algorithms.

Setup: Given a security parameter k, the private key generator (PKG) generates the system's public parameters $params$. Among the parameters produced by Setup there is a key P_{Pub} that is made public. There is also a corresponding master key *s* which is kept secret.

Extract: Given an identity ID, the PKG computes the corresponding private key S_{ID} and transmits it to its owner in a secure way.

Keydis: Given a private key S_{ID} associated with an identity ID, the number of signcryption members n and a threshold parameter t, this algorithm generates n shares of S_{ID} and provides each one to the signcryption members $M_1, ...M_n$. It

also generates a set of verification keys that can be used to check the validity of each shared private key. We denote the shared private keys and the matching verification keys by $\{S_i\}_{i=1,\ldots n}$ and $\{y_i\}_{i=1,\ldots n}$, respectively. Note that each (S_i, y_i) is sent to M_i, then M_i publishes y_i but keeps S_i secret.

Signcrypt: Give a message *m*, the private keys of t members $\{S_i\}_{i=1,\ldots n}$ in a sender group U_A with identity ID_A, a receiver's identity ID_B, it outputs an ID-based (t, n) threshold signcryption σ on the message m.

Unsigncrypt: Give a ciphertext σ, the private key of the receiver S_{ID_B}, the identity of the sender group ID_A, it outputs the plaintext *m* or the symbol"$\perp$". If σ is an invalid ciphertext between the group U_A and the receiver. We make the consistency constraint that

If $\sigma = Signcrypt(m, \{S_i\}_{i=1,\ldots n}, ID_B)$, then $m = Unsigncrypt(\sigma, ID_A, S_{ID_B})$.

3.2 Security Notions

We use the security notions for ID-based threshold signcryption schemes from LY scheme [10] directly, they are indistinguishability against adaptive chosen ciphertext attacks, unforgeability against adaptive chosen messages attacks and the robustness.

Definition 4 (Confidentiality). An ID-based threshold signcryption scheme (IDTSC) is said to have the indistinguishability against adaptive chosen ciphertext attacks property (IND-IDTSC-CCA2) if no polynomially bounded adversary has a non-negligible advantage in the following game.

1) The challenger C runs the Setup algorithm with a security parameter *k* and sends the system parameters to the adversary A.

2) A performs a polynomially bounded number of queries (these queries may be made adaptively, i.e. each query may depend on the answers to the previous queries).

Key extraction queries: A chooses an identity ID. C computes S_{ID} and sends it to A.

Signcryption queries: A produces a sender group U_i with identity ID_i, an identity ID_j and a plaintext *m*. C computes $S_{ID_i} = Extract(ID_i)$ and runs Keydis to output *n* shared private keys $\{S_i\}_{i=1,\ldots n}$. C sends $Signcrypt(m, \{S_i\}_{i=1,\ldots n}, ID_j)$ to *A*.

Unsigncryption queries: A produces a sender group U_i with identity ID_i, an identity ID_j, and a ciphertext σ, C generates the private key $S_{ID_j} = Extract(ID_j)$ and sends the result of $Unsigncrypt(\sigma, ID_i, S_{ID_j})$ to A (this result can be the"$\perp$"symbol if σ is an invalid ciphertext)

3) A generates two equal length plaintexts m_0, m_1, a sender group U_A with identity ID_A, and an identity ID_B on which he wants to be challenged. He can not have asked the private key corresponding to ID_B in the first stage.

4) C takes a bit $b \in_R \{0,1\}$ and runs Keydis to output n shared private keys $\{S_i\}_{i=1,...n}$, C sends the result of $\sigma = Signcrypt(m_b, \{S_i\}_{i=1,...n}, ID_B)$ to A.

5) A can ask a polynomially bounded number of queries adaptively again as in the first stage. 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, A produces a bit $b' \in_R \{0,1\}$ and wins the game if $b = b'$. The advantage of A is defined as $Adv(A) = 2P_r[b = b'] - 1$, where $Adv(A)$ 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. 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 5 (Unforgeability). An ID-based threshold signcryption scheme (IDTSC) is said to have the existential unforgeability against adaptive chosen messages attacks (EUF-IDTSC-CMA) if no polynomially bounded adversary has a non-negligible advantage in the following game.

1) The challenger C runs the Setup algorithm with a security parameter *k* and sends the system parameters to A.

2) A corrupts $t-1$ members in the sender group.

3) 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: As that in the Definition 4.

Private keys queries to the corrupted members: A chooses an identity ID. C computes $S_{ID} = Extract(ID)$ and runs Keydis to output *n* shared private keys $\{S_i\}_{i=1,...n}$, C sends S_i for $i = 1,...t-1$ to A.

Signcryption (Unsigncryption) queries: As that in the Definition 4.

4) Finally, A produces a new triple (ID_A, ID_B, σ) (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 and wins the game if the result of the $Unsigncrypt(\sigma, ID_A, S_{ID_B})$ is not the"$\perp$"symbol.

The advantage of 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 [11].

Definition 6 (Robustness). An ID-based (t,n) threshold signcryption scheme (IDTSC) is said to be robust if it computes a correct output even in the presence of a malicious adversary that makes the $t-1$ corrupted members deviate from the normal execution.

4 The LY Scheme and Its Security Analysis

4.1 The LY Scheme

The following paragraphs show the details of LY scheme [10].

Setup: Given k, PKG chooses groups G_1 and G_2 of prime order q (with G_1 additive and G_2 multiplicative), a generator P of G_1, a bilinear map $\hat{e}: G_1 \times G_1 \to G_2$, a secure symmetric cipher (E,D) and hash functions $H_1: \{0,1\}^* \to G_1, H_2: G_2 \to \{0,1\}^{n_1}$, $H_3: \{0,1\}^* \to Z_q^*$. PKG chooses a master-key $s \in_R Z_q^*$ and computes $P_{pub} = sP$. Then PKG publishes system parameters $\{G_1, G_2, n_1, \hat{e}, P, P_{pub}, E, D, H_1, H_2, H_3\}$ and keeps s secret.

Extract: Given an identity ID, PKG computes $Q_{ID} = H_1(ID)$ and the private key $S_{ID} = sQ_{ID}$. Then PKG sends the private key to its owner in a secure way.

Keydis: Suppose a threshold t and n satisfy $1 \le t \le n \le q$. To share the private key S_{ID_A} among the group U_A, the trusted dealer performs the steps below:

1) Choose $F_1, ...F_{t-1} \in_R G_1^*$, construct $F(x) = S_{ID_A} + xF_1 + ... + x^{t-1}F_{t-1}$ and compute $S_i = F(i)$ for $i = 0, ..., n$. Note that $S_0 = S_{ID_A}$.

2) Send S_i to M_i for $i = 0, ..., n$ secretly. Broadcast $y_0 = \hat{e}(S_{ID_A}, P)$ and $y_j = \hat{e}(F_j, P)$ for $j = 1, ..., t-1$.

3) Each M_i then checks whether his share S_i is valid by checking whether $\hat{e}(S_i, P) = \prod_{j=0}^{t-1} y_j^{i^j}$. If S_i is not valid, M_i broadcasts an error and requests a new one.

Signcrypt: We assume that $M_1, ...M_t$ are the t members who want to cooperate to signcrypt a message m on behalf of the group U_A, each M_i $(1 \le i \le t)$ uses Cheng et al.'s ID-based signature scheme[14] to generate the partial signature and an appointed

clerk C who is one of the t members, combines the partial signatures to generate the final threshold signcryption.

1) Each M_i chooses $x_i \in_R Z_q^*$, computes $R_{1i} = x_i P$ and $R_{2i} = x_i P_{pub}$ and sends (R_{1i}, R_{2i}) to C.

2) C computes $R_1 = \sum_{i=1}^{t} R_{1i}$, $R_2 = \sum_{i=1}^{t} R_{2i}$ $\tau = \hat{e}(R_2, Q_{ID_B})$, $k = H_2(\tau)$, $c = E_k(m)$, $h = H_3(m, R_1, k)$. Then C sends h to M_i, $i = 1,...,t$.

3) M_i computes the partial signature $W_i = x_i P_{pub} + h\eta_i S_i$ and sends it to C, where $\eta_i = \prod_{j=1, j\neq i}^{t} -j(i-j)^{-1} \bmod q$.

4) When receiving M_i's partial signature W_i, C verifies its correctness by checking if the following equation holds: $\hat{e}(P, W_i) = \hat{e}(R_{1i}, P_{pub}) \cdot (\prod_{j=0}^{t-1} y^{i^j})^{h\eta_i}$.

If all signatures are verified to be legal, C computes $W = \sum_{i=1}^{t} W_i$, otherwise rejects it and requests a valid one. The final threshold signcryption is $\sigma = (c, R_1, W)$

Unsigncrypt: When receiving $\sigma = (c, R_1, W)$, Bob follows the steps below.

1) Compute $\tau = \hat{e}(R_1, S_{ID_B})$ and $k = H_2(\tau)$.

2) Recover $m = D_k(c)$.

3) Compute $h = H_3(m, R_1, k)$ and accept σ if and only if the following equation holds: $\hat{e}(W, P) = \hat{e}(R_1 + hQ_{ID_A}, P_{pub})$.

4.2 Security Analysis of the LY Scheme

The security risks depend on whether the clerk C is an honest clerk, we suppose C is the M_j, in the first signcryption process, C receives (R_{1i}', R_{2i}') from M_i $i = 1,...t, i \neq j$, C chooses $x_j' \in_R Z_q^*$, computes $R_{1j}' = x_j' P$, $R_{2j}' = x_j' P_{pub}$, $R_1' = \sum_{i=1}^{t} R_{1i}'$, $R_2' = \sum_{i=1}^{t} R_{2i}'$, $\tau' = \hat{e}(R_2', Q_{ID_B})$, $k' = H_2(\tau')$, $c' = E_{k'}(m_1)$, $h_1 = H_3(m_1, R_1', k')$. Then C sends h_1 to M_i for $i = 1,...,t$. Each M_i computes the partial signature $W_i' = x_i' P_{pub} + h_1 \eta_i S_i$, $\eta_i = \prod_{j=1, j\neq i}^{t} -j(i-j)^{-1} \bmod q$ and

sends it to C. When receiving M_i's partial signature W_i', C verifies by checking if the following equation holds: $\hat{e}(P,W_i') = \hat{e}(R_{1i}', P_{pub}) \cdot (\prod_{j=0}^{t-1} y^{i^j})^{h\eta_i}$, otherwise rejects it and requests a valid one .If all signatures are verified to be legal, C computes $W' = \sum_{i=1}^{t} W_i'$. The final signcryption is $\sigma' = (c', R_1', W')$.

In the second threshold signcryption process, C receives (R_{1i}^*, R_{2i}^*) from every M_i, C will chooses $x_j^* \in_R Z_q^*$, computes $R_{1j}^* = x_j^* P$, $R_{2j}^* = x_j^* P_{pub}$, $R_1^* = \sum_{i=1}^{t} R_{1i}^*$, $R_2^* = \sum_{i=1}^{t} R_{2i}^*$, $\tau^* = \hat{e}(R_2^*, Q_{ID_B})$, $k^* = H_2(\tau^*)$, $h_2 = H_3(m_2, R_1^*, k^*)$, C will checks whether h_1 and h_2 are coprime, if not, chooses another $x_j^{**} \in_R Z_q^*$, if yes, C computes $c^* = E_{k^*}(m_2)$ and sends h_2 to $M_i, i = 1,...,t$. Each M_i computes the partial signature $W_i^* = x_i^* P_{pub} + h_2 \eta_i S_i, \eta_i = \prod_{j=1, j\neq i}^{t} -j(i-j)^{-1} \bmod q$ and sends it to C. When receiving M_i's signature W_i^*, C verifies its correctness by checking whether the following equation holds: $\hat{e}(P,W_i^*) = \hat{e}(R_{1i}^*, P_{pub}) \cdot (\prod_{j=0}^{t-1} y^{i^j})^{h\eta_i}$. If all signatures are verified to be legal, C computes $W^* = \sum_{i=1}^{t} W_i^*$, otherwise rejects it and requests a valid one. The final threshold signcryption is $\sigma^* = (c^*, R_1^*, W^*)$. In the third threshold signcryption process, the clerk can impersonate all the t members to forge any valid signcryption to any receiver, for the h_1 and h_2 are coprime, so there are two numbers $a, b \in_R Z_q^*$ which make the following equation holds: $a \cdot h_1 + b \cdot h_2 = 1$. So we get the value $\eta_i S_i$, $i = 1,...,t, i \neq j$ from the following equations:

$$\begin{cases} W_i^* = x_i^* P_{pub} + h_2 \eta_i S_i \\ \mathrm{W_i}' = x' P_{pub} + h_1 \eta_i S_i \\ R_{2i}^* = x_i^* P_{pub} \\ R_{2i}' = x' P_{pub} \end{cases} \Rightarrow \begin{cases} W_i^* - x_i^* P_{pub} = W_i^* - R_{2i}^* = h_2 \eta_i S_i \\ \mathrm{W_i}' - x' P_{pub} = \mathrm{W_i}' - R_{2i}' = h_1 \eta_i S_i \end{cases}$$

$$\Rightarrow a \cdot h_1 \eta_i S_i + b \cdot h_2 \eta_i S_i = a \cdot (\mathrm{W_i}' - R_{2i}') + b \cdot (W_i^* - R_{2i}^*)$$
$$= (a \cdot h_1 + b \cdot h_2) \cdot \eta_i S_i = \eta_i S_i, i = 1,...,t, i \neq j$$

So to M_i's new signature $W_i = x_i P_{pub} + h\eta_i S_i$, the part of $x_i P_{pub}$ can be chosen by C itself, and the part of $h\eta_i S_i$ can be computed by $h \cdot \eta_i S_i$ while the values h and $\eta_i S_i$ are known to C. C can impersonate all the t members to forge any valid signcryption to any receiver.

5 Our Improved Scheme and Its Analysis

In this section, we first present our improved scheme, and then give the security proofs based on the DBDHP and CDHP, the efficiency analysis will be in the last part of this section.

5.1 Our Scheme

The proposed scheme involves four roles: PKG, a trusted dealer, a sender group $U_A = \{M_1, ...M_n\}$ with the identity ID_A and a receiver Bob with identity ID_B. The following shows the details of our scheme.

Setup, Extract, **Keydis:** These algorithms are same as the LY scheme [10];

Signcrypt: Without loss of generality, we assume that $M_1, ...M_t$ are the *t* members who want to cooperate to signcrypt a message *m* on behalf of the group U_A, each M_i generate the partial signature and an appointed clerk C combines the partial signatures to generate the final threshold signcryption.

1) M_i chooses $x_i \in_R Z_q^*$, computes $R_{1i} = x_i P$, $R_{2i} = \hat{e}(Q_{ID_B}, x_i P_{pub})$ and sends (R_{1i}, R_{2i}) to C;

2) C computes $R_1 = \sum_{i=1}^{t} R_{1i}$, $R_2 = \prod_{i=1}^{t} R_{2i}$, $k = H_2(R_2)$, $c = E_k(m)$, $h = H_3(m, k)$. Then C sends h to M_i for $i = 1, ..., t$;

3) M_i computes $W_i = x_i P_{pub} + h\eta_i S_i$, $\eta_i = \prod_{j=1, j\neq i}^{t} -j(i-j)^{-1} \bmod q$;

4) When receiving M_i's signature W_i, C verifies its correctness by checking whether equation holds: $\hat{e}(P, W_i) = \hat{e}(R_{1i}, P_{pub}) \cdot (\prod_{j=0}^{t-1} y^{i^j})^{h\eta_i}$, if all partial signatures are verified to be legal, C computes $W = \sum_{i=1}^{t} W_i$, otherwise rejects it and requests a valid one. The final signcryption is $\sigma = (c, R_1, W)$.

Unsigncrypt: When receiving $\sigma = (c, R_1, W)$, Bob follows the steps below.

1) Computes $R_2 = \hat{e}(R_1, S_{ID_B})$, $k = H_2(R_2)$;

2) Recovers $m = D_k(c)$;

3) Computes $h = H_3(m, k)$ and accepts σ if and only if the following equation holds: $\hat{e}(W, P) = \hat{e}(R_1, P_{pub}) \cdot \hat{e}(Q_{ID_A}, P_{pub})^h = \hat{e}(P_{pub}, R_1 + h \cdot Q_{ID_A})$.

5.2 Security Analysis

The correctness can be easily verified, the security analysis will focus on the following theorems.

5.2.1 Confidentiality

Theorem 1 (Confidentiality). We assume that an adversary A who is able to distinguish ciphertext during the game of Definition 4 with an advantage ε when running in a time t and asking at most q_{H_1} H_1 queries, at most q_{H_2} H_2 queries, at most q_{H_3} H_3 queries, at most q_K key extraction queries, at most q_S signcryption queries and at most q_U unsigncryption queries. There will be a distinguisher C which can solve the DBDHP with an advantage $Adv(C) \geq (\frac{(\varepsilon+1)}{2}(1-\frac{q_U}{2^k})-\frac{1}{2})(\frac{1}{q_{H_1}}) = \frac{\varepsilon(2^k - q_U) - q_U}{q_{H_1} \cdot 2^{k+1}}$ in a time $O(t + (q_{H_3} \cdot q_S + {q_S}^2 + 3q_U)T_{\hat{e}})$, where $T_{\hat{e}}$ denotes the computation time of the bilinear map.

The proof will be presented in the Appendix A.

5.2.2 Unforgeability

The acceptable security notion for signature schemes is unforgeability under chosen message attack. Hess[12] presented an unforgeability notion for identity-based signature against chosen message attack (UF-IDS-CMA). After that, Beak and Zheng[13] defined unforgeability notion for IDTHS against chosen message attack (UF-IDTHS-CMA) and gave the relationship between UF-IDS-CMA and UF-IDTHS-CMA through Simulatability, and they had proved the following Lemma 1. From these results, we can prove the following Theorem 2.

Definition 7. An ID-based threshold signature scheme is said to be simulatable if the following conditions hold.

1) The private key distribution is simulatable: given the system parameters *params* and the identity ID, there exists a simulator which can simulate the view of the adversary on an execution of private key distribution.

2) The threshold signature generation is simulatable: given the system parameters *params*, the identity ID, the message m, the corresponding signature

(R_1, W), $t-1$ shares of the private key that matches to ID of the corrupted members, and the corresponding verification keys, there exists a simulator which can simulate the view of the adversary on an execution of threshold signature generation.

Lemma 1. If the identity-based threshold signature (IDTHS) scheme is simulatable and the corresponding identity-based signature scheme is UF-IDS-CMA secure, then the IDTHS is UF-IDTHS-CMA secure.

Theorem 2. Our identity-based threshold signcryption scheme satisfies unforgeability under CDHP in the random oracle model.

Proof. The proposed scheme uses Cheng et al.'s ID based signature scheme[14]. Cheng et al.'s scheme has been proved to be secure in the sense of unforgeability under the Computational Diffie-Hellman (CDH) assumption in the random oracle model. Therefore, we only need to prove the proposed scheme is simulatable. Our scheme uses Baek and Zheng's private key distribution scheme[14]. Baek and Zheng have proved that their private key distribution scheme is simulatable in [13]. Now, we prove the threshold signature generation is simulatable. Given the system parameters *params*, the identity ID_A, the message m, the encryption key k, the corresponding signature (R_1, W), $t-1$ shares $\{S_i\}_{i=1,2,...t-1}$ of the private key S_{ID_A}, the corresponding verification keys $\{y_j\}_{j=o,....t}$. The adversary computes $h = H_3(m,k), W_i = x_i P_{pub} + h\eta_i S_i, i = 1,...,t-1$. Let $f(x)$ be a polynomial of degree $t-1$ such that $f(0) = W$, $f(i) = W_i, i = 1,...,t-1$. The adversary can compute $f(i) = W_i, i = t,.....n$. So, the proposed scheme is secure in the sense of unforgeability.

Theorem 3 (Robustness). The proposed ID-based threshold signcryption scheme is robust against an adversary which is allowed to corrupt any $t-1$ members, where $n \geq 2t-1$.

Proof: In the Keydis phase, each member M_i can validate his private key share S_i using the published verification keys $\{y_j\}_{j=0,1,....t-1}$. In the Signcrypt phase, any $t-1$ or fewer members can not generate a valid signcryption, and only t or more members can generate a valid signcryption. The clerk C first verifies all the partial signatures by $\hat{e}(P, W_i) = \hat{e}(R_{1i}, P_{pub}) \cdot (\prod_{j=0}^{t-1} y^{i^j})^{h\eta_i}$ and then chooses the valid ones to generate a threshold signcryption. Even if having corrupted up to $t-1$ members, the adversary still cannot produce a valid threshold signcryption. While the clerk C can get t valid partial signatures, thus can produce a valid threshold signcryption.

5.3 Efficiency Analysis

We mainly consider the pairing operations which cost far more than the others operations such as point scalar multiplications, exponentiations, and etc. From the Table.1 in the Appendix B, we can see that both Duan et al.'s scheme[6] and Peng and Li's scheme[7] need $3t + 4$ pairing computations while our scheme only needs $2t+3$ pairing computations. The proposed scheme is more efficient than Duan et al.'s scheme[6] and Peng and Li's scheme[7]. Notice that the schemes proposed in [6] and [7] do not give provable security proofs, also we notice that our scheme is a little more efficient than the LY scheme [10].

6 Conclusion

Although Li and Yu pointed out that the scheme [10] they proposed in ICCCAS'2008 is the first provably secure ID-based threshold signcryption scheme, which is secure against adaptive chosen ciphertext attacks and secure in the sense of unforgeability, we show that signcryption is easily forged by the appointed clerk who is one of the members and then give a secure version which we prove its confidentiality under the Decisional Bilinear Diffie-Hellman assumption and its unforgeability under the Computational Diffie-Hellman assumption in the random oracle model. The scheme turns out to be more efficient than the previously proposed schemes.

References

[1] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 341–349. Springer, Heidelberg (1985)

[2] Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

[3] Zheng, Y.: Digital signcryption or how to achieve cost(signature& encryption) cost(signature) +cost(encryption). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)

[4] Malone-Lee, J.: Identity-based signcryption. Cryptology ePrint Archive (2002), http://eprint.iacr.org/2002/098/

[5] Libert, B., Quisquater, J.J.: New identity based signcryption schemes from Pairings.Cryptology ePrint Archive (2003), http://eprint.iacr.org/2003/023/

[6] Duan, S., Cao, Z., Lu, R.: Robust ID-based threshold signcryption scheme form Pairings. In: Proceedings of the 3rd international conference on Information security(Infosecu 2004), pp. 33–37. ACM Press, New York (2004)

[7] Peng, C., Li, X.: An identity-based threshold signcryption scheme with semantic security. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3802, pp. 173–179. Springer, Heidelberg (2005)

[8] Libert, B., Quisquater, J.J.: A new identity based signcryption schemes from pairings. In: Proc. 2003 IEEE information theory workshop, Paris, France, pp. 155–158 (2003)

[9] Ma, C., Chen, K., Zheng, D., Liu, S.: Efficient and proactive threshold signcryption. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 233–243. Springer, Heidelberg (2005)
[10] Li, F., Yu, Y.: An efficient and provably secure ID-Based threshold signcryption scheme. In: Proc. International Conference on Communications, Circuits and Systems 2008 (IEEE ICCCAS 2008) (2008); Cryptology ePrint Archive, http://eprint.iacr.org/2008/187
[11] An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
[12] Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
[13] Baek, J., Zheng, Y.: Identity-based threshold signature scheme from the bilinear pairings. In: IAS 2004 track of ITCC 2004. IEEE Computer Society, pp. 124–128 (2004)
[14] Cheng, X., Liu, J., Wang, X.: An identity-based signature and its threshold version. In: Proc.19th International Conference on Advanced Information Networking and Applications -AINA 2005, pp. 973–977 (2005)

Appendix A

Proof. We assume that C receives a random instance $(P, aP, bP, cP; h)$ of the DBDHP, his goal is to decide whether $h = \hat{e}(P,P)^{abc}$ or not. C will run A as a subroutine and act as A's challenger in the IND-IDTSC-CCA2 game. During the game, A will consult C for answers to the random oracles H_1, H_2 and H_3. These answers are randomly generated, but to maintain the consistency and to avoid collision, C keeps three lists L_1, L_2, L_3 respectively to store the answers. The following assumptions are made.

1) 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 A in an unsigncryption query.

At the beginning of the game, C gives A the system parameters with $P_{pub} = cP$, c is unknown to C. This value simulates the master-key value for the PKG. Then, C chooses a random number $j \in \{1, 2, ..., q_{H_1}\}$, A asks a polynomially bounded number of H_1 queries on identities of his choice. At the j-th H_1 query, C answers by $H_1(ID_j) = bP$. For queries $H_1(ID_e)$ with $e \neq j$, C chooses $b_e \in_R Z_q^*$, puts the pair $(ID_e; b_e)$ in list L_1 and answers $H_1(ID_e) = b_e P$. We now explain how the other kinds of queries are treated by C.

H_2 **queries:** On a $H_2(R_{2,e})$ query, C searches a pair $(R_{2,e}, k_e)$ in the list L_2. If such a pair is found, C answers k_e, otherwise he answers A by a random binary sequence $k \in_R \{0,1\}^{n_1}$ such that no entry (R_e, k) exists in L_2 (in order to avoid collisions on H_2) and puts the pair (R_e, k) into L_2.

H_3 **queries:** On a $H_3(m_e, k_e)$ query, C checks if there exists $(m_e, k_e, h_{3,e})$ in L_3. If such a tuple is found , C answers $h_{3,e}$, otherwise he chooses $h_e \in_R Z_q^*$, gives it as an answer to the query and puts the tuple $(m_e, R_{1,e}, h_e)$ into L_3.

Key extraction queries: A asks a question $Extract(ID_e)$, if $ID_e = ID_j$, then 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 C previously answered $H_1(ID_e) = b_e P$ on an H_1 query on ID_e. The private key corresponding to ID_e is $b_e \cdot P_{pub} = cb_e P$, it is computed by C and returned to A.

Signcryption queries: At any time, A can perform a signcryption query for a plaintext m, a sender group U_A with identity ID_A and a receiver with identity ID_B. We have the following three cases to consider.

– Case 1: $ID_A \neq ID_j$. C computes the private key S_{ID_A} corresponding to ID_A by running the key extraction query algorithm. C runs Keydis to output n shared private keys $\{S_i\}_{i=1,\ldots,n}$, finally, C answers the query by a call to $Signcrypt(m, \{S_i\}_{i=1,\ldots,t}, Q_{ID_B})$.

– Case 2: $ID_A = ID_j$ and $ID_B \neq ID_j$. C chooses $x, h'' \in_R Z_q^*$ and computes $R_1 = xP - h'' \cdot Q_{ID_A}$, $W = x \cdot P_{pub}$ and $R_2 = \hat{e}(R_1, S_{ID_B})$ (C could obtain S_{ID_B} from the key extraction algorithm because $ID_B \neq ID_j$). C runs the H_2 simulation algorithm to find $k = H_2(R_2)$ and computes $c = E_k(m)$. C then checks if L_3 already contains a tuple (m_e, k_e, h') with $h'' \neq h'$. In this case, C repeats the process with another random pair (x, h''') until finding a tuple (m_e, k_e, h''') 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 (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, h'') is found, the ciphertext

$\sigma = (c, R_1, W)$ appears to be valid from A's viewpoint. C has to compute one pairing operation for each iteration of the process.

– Case 3: $ID_A = ID_j$ and $ID_B = ID_j$. C chooses $x', h' \in_R Z_q^*$, computes $R_1' = x'P - h' \cdot Q_{ID_A}$, $W' = x' \cdot P_{pub}$, and chooses $R_2' \in_R G_2$, $k' \in_R \{0,1\}^{n_1}$ such that no entry $(\cdot, k')$ is in L_2 and computes $c = E_{k'}(m)$. C then checks if L_3 already contains a tuple (m, R_1', h), If not, C puts the tuple (m, R_1', h) into L_3 and (R_2', k') into L_2. Otherwise, C chooses another random pair (x', h') and repeats the process as above until he finds a tuple (m, R_1', h) whose first two elements do not appear in an entry of L_3. Once an appropriate pair (x', h') is found, C gives the ciphertext $\sigma' = (c', R_1', W')$ to A. As 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 $\sigma^* = (c^*, R_1^*, W^*)$ between a sender group with identity ID_A and a receiver with identity ID_B. We have the following two cases to consider.

–Case 1: $ID_B = ID_j$. C always answers A that σ^* is invalid.

–Case 2: $ID_B \neq ID_j$. C computes $R_2^* = \hat{e}(R_1^*, S_{ID_B})$ (C could obtain S_{ID_B} from the key extraction algorithm, because $ID_B \neq ID_j$). C then runs the H_2 simulation algorithm to obtain $k^* = H_2(R_2^*)$ and computes $m^* = D_{k^*}(c)$. Finally, C runs the H_3 simulation algorithm to obtain $h^* = H_3(m^*, k^*)$ and checks whether $\hat{e}(W^*, P) = \hat{e}(R_1^*, P_{pub}) \cdot \hat{e}(Q_{ID_A}, P_{pub})^{h^*}$ holds. If the above equation does not hold, C rejects the ciphertext. Otherwise C returns m^*. It is easy to see that, for all queries, the probability to reject a valid ciphertext does not exceed q_U / q_{H_1}.

After the first stage, A picks a pair of identities on which he wishes to be challenged. Note that *C* fails if *A* has asked a key extraction query on ID_j during the first stage. We know that the probability for C not to fail in this stage is $(q_{H_1} - q_K) / q_{H_1}$, furthermore, with a probability exactly $1/(q_{H_1} - q_K)$, A

chooses to be challenged on the pair (ID_i, ID_j) with $i \neq j$. Hence the probability that A's response is helpful to C is $1/q_{H_1}$. If A has submitted a key extraction query on ID_j, C fails, because he is unable to answer the question. On the other hand, if A does not choose (ID_i, ID_j) as target identities, C fails too. A outputs two plaintexts m_0 and m_1. C chooses $b \in_R \{0,1\}$ and signcrypts m_b and sets $R_1' = aP$, obtains $k' = H_2(h)$ (where h is C candidate for the DBDH problem) from the H_2 simulation algorithm, and computes $c_b = E_{k'}(m_b)$. Then C chooses $W' \in_R G_2$ and sends the ciphertext $\sigma' = (c_b, R_1', W')$ to A.

A then performs a second series of queries. At the end of the simulation, he produces a bit b' for which he believes the relation $\sigma' = Signcrypt(m_{b'}, \{S_i\}_{i=1,\ldots,t}, ID_j)$ holds. At this moment, if $b' = b$, C outputs $h = \hat{e}(P,P)^{abc}$ is true, and the DBDH problem has been solved, otherwise *C* stops and outputs "failure".

Taking into account all the probabilities that C will not fail its simulation, the probability that A chooses to be challenged on the pair (ID_i, ID_j), and also the probability that A wins the IND-IDTSC-CCA2 game, the value of $Adv(C)$ is calculated as follows:

$$Adv(C) \geq (\frac{(\varepsilon+1)}{2}(1-\frac{q_U}{2^k}) - \frac{1}{2})(\frac{1}{q_{H_1}}) = \frac{\varepsilon(2^k - q_U) - q_U}{q_{H_1} \cdot 2^{k+1}}$$

Appendix B

Table 1. Efficiency comparison

The schemes	Signcrypt	Unsigncrypt	Provable security (Y/N)	Against Clerk attacks (Y/N)
Duan et al. [6]	3t	4	N	Y
Peng and Li [7]	3t	4	N	Y
LY[10]	2t+1	3	Y	N
Our scheme	2t	3	Y	Y

Leaky Random Oracle
(Extended Abstract)

Kazuki Yoneyama[1,⋆], Satoshi Miyagawa[2,⋆⋆], and Kazuo Ohta[1]

[1] The University of Electro-Communications.
[2] NTT DoCoMo, Inc.
yoneyama@ice.uec.ac.jp

Abstract. This work focuses on vulnerability of hash functions due to sloppy usage or implementation in the real world. If our cryptographic research community succeeded in development of perfectly secure random function as random oracle, it might be broken in some sense by invalid uses. In this paper, we propose a new variant of the random oracle model in order to analyze security of cryptographic protocols under the situation of an invalid use of hash functions. Our model allows adversaries to obtain contents of the hash list of input and output pairs arbitrarily. Also, we analyze security of several prevailing protocols (FDH, OAEP, Cramer-Shoup cryptosystem, Kurosawa-Desmedt cryptosystem, NAXOS) in our model. As the result of analyses, we clarify that FDH and Cramer-Shoup cryptosystem are still secure but others are insecure in our model. This result shows the separation between our model and the standard model.

Keywords: random oracle model, standard model, hash list, provable security, leakage.

1 Introduction

Hash functions are one of most important building blocks of cryptographic protocols. Indeed, hash functions are widely used various protocols, e.g., digital signature, public-key cryptosystem, authenticated key exchange, etc.

In the practical sense, hash functions are used in order to hide private information to other parities in the protocol. The spreading use of transaction by small electronic devices has been encouraging researchers to develop an efficient and practical security system in a limited resources environment. Since computational costs of hash functions are lower than that of public-key cryptosystem, hash functions is received much attention to construct protocols for such low-power devices.

In the theoretical sense, hash functions are frequently modeled as *random oracles* [1]. Random oracle is an idealized random function which is usable for

⋆ Supported by JSPS Research Fellowships for Young Scientists.
⋆⋆ This work was partially done while the author was a student at the University of Electro-Communications, Japan.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 226–240, 2008.

parties and adversaries in the protocol. We use random oracle model (ROM) (i.e., executed with random oracles) as a technique in order to prove security of various protocols. Mostly, proofs with ROM are easier than the model without random oracles, i.e., the standard model (SM), and can provide tight security reductions. Thus, ROM is a useful tool for the provable security.

On the other hand, Canetti et al. [2,3] showed that there are digital signature schemes and public-key cryptosystems which are secure in ROM but insecure if random oracles are instantiated by real hash functions. Thus, recently, proofs with ROM may seem to be unfavorable for practical uses. However, since to prove security of protocols in SM is generally hard, ROM still has an important role to design new protocols as the guideline for the provable security.

1.1 Motivation

Canetti et al.'s impossibility result would be avoidable if a perfectly secure hash function which has all capabilities of random oracles was developed. If so, does the protocol which is proved security in ROM keep its security if random oracles are instantiated by such perfectly secure hash functions? However, in the practical scenario, an unexpected event may occur on hash functions but does not occur on random oracles. In particular, we focus on vulnerability of hash functions due to sloppy usage or implementation in the real world.

Canetti and Krawczyk [4] formulated a security notion of authenticated key exchange. Their definition captured security under the situation where an ephemeral secret information (local randomness) is leaked. These leakages may occur in the case of that a storage or memory which save a local randomness is attacked by various types of attack or the case of that a randomness generator is corrupted. Note that, such a type of vulnerability is not caused by errors of protocol itself but caused by sloppy usages or implementations.

In this work, we apply this view to hash functions. That is, we consider the situation that pairs of inputs and outputs (contents of the hash list) of hash functions can be leaked to adversaries. These leakages may also occur by sloppy usages or implementations. For example, the hash list may remain in the memory for reuse of hash values in order to reduce computational costs or for failing to release temporary memory area, then contents of the memory may be revealed by various attacks, e.g., malicious Trojan Horse programs, Cold Boot Attacks [5]. Thus, even if we successfully developed exceedingly secure hash functions, such a leakage might be possible.

In this paper, we formulate a new model capturing the above situation in order to discuss security (or insecurity) of protocols which use hash functions as building blocks when such a leakage of the hash list occurs. In order to concentrate effects of the leakage, we suppose that hash functions are ideal as random oracles but contents of the hash list can be leaked to adversaries. By using this model, we give a new criterion of security in ROM and analyze several prevailing protocols.

1.2 Our Contribution

Our main contributions are formulating a new variant of ROM, named *leaky random oracle model* (LROM), to capture the leakage of the hash list and analyzing security and insecurity of prevailing protocols in our model.

Leaky Random Oracle Model. Our model (LROM) allows adversaries to obtain contents of the hash list of input and output pairs in arbitrary timings. Thus, virtually, adversaries always can observe the addition of each hash value and can know the timing of the addition. We model the ordinary query in order to obtain a hash value to (leaky) random oracle as *hash query* and the special query in order to obtain contents of hash list to leaky random oracle as *leak query*. Therefore, LROM is trivially stronger than ROM, i.e., a secure protocol in LROM is also secure in ROM.

Security Analyses of Protocols. By using LROM, we can confirm whether each cryptographic protocol is secure or not if the leakage of the hash list occurs. In this paper, we choose five prevailing protocols for analyzing security in LROM and obtain the result of analyses as follows;

- **FDH:** is secure in both ROM and LROM,
- **OAEP:** is secure in ROM but insecure in LROM,
- **Cramer-Shoup cryptosystem:** is secure in both SM and LROM,
- **Kurosawa-Desmedt cryptosystem:** is secure in SM but insecure in LROM, and
- **NAXOS:** is secure in ROM but insecure in LROM.

Separation from the standard model. Our result of analyses shows the separation between our model and the standard model because of two following observations;

- FDH is secure in LROM under the assumption of trapdoor permutation. However, Dodis et al. [6] showed that FDH is not provable in SM under the same assumption.
- Kurosawa-Desmedt cryptosystem is secure in SM under the DDH assumption, the assumption of universal hash function family and the assumption of symmetric key encryption. However, Kurosawa-Desmedt cryptosystem is insecure in LROM by instantiating hash functions by leaky random oracles under same assumptions.

Difference from randomness revealing. Also, our result shows the difference between our model and ROM under randomness revealing because of the following observation;

- NAXOS is secure in ROM under the leakage of local randomness. However, NAXOS is insecure in LROM even if there is no leakage of local randomness.

1.3 Related Works

Some studies consider modeling of weak random oracles and analyze protocols in their model. Nielsen [7] introduced the non-programmable random oracle model by restricting the simulation of random oracle as the simulator can only set answers of random oracles according to some restriction. Liskov [8] showed the way to construct weak hash functions by adding an additional oracle which can break some property of random oracles, e.g., one-wayness and collision-resistance. Pasini and Vaudenay [9] applied Liskov's model into analyses of the hash-and-sign paradigm. Unruh [10] formulated a variant of ROM by giving oracle-dependent auxiliary inputs to adversaries. In this setting, adversaries can get an auxiliary input that can contain information about the random oracle. Numayama et al. [11] relaxed Liskov's model and analyzed digital signature schemes in their model.

Therefore, previous works studied on effects into security of various protocols by vulnerability of hash functions itself. On the other hand, our LROM is different with these on the point of that LROM focuses on vulnerability of hash functions due to sloppy usages or implementations.

2 Leaky Random Oracle Model

ROM is one of techniques for provable security under idealized hash functions by using random oracles. Random oracle models truly idealized hash function which locally has the hash list of inputs and outputs. LROM is a variant of ROM which allows adversaries to obtain contents of the hash list in arbitrary timing. Thus, adversaries can correspond an input to the random oracle and an output (hash value). The definition of LROM is as follows;

Definition 1 (Leaky Random Oracle Model). *LROM is a model assuming the leaky random oracle. We suppose a hash function* $H : X \to Y$ *such that* $x_i \in X$, $y_i \in Y$ *(i is an index), and* X *and* Y *are both finite sets. Also, let* $\mathcal{L}_H$ *be the hash list of* H*. We say* H *is a leaky random oracle if* H *can be simulated by the following procedure;*

Initialization: $\mathcal{L}_H \leftarrow \perp$
Hash query: *For a hash query* x_i *to* H*, behave as follows;*
$<$*If* $x_i \in \mathcal{L}_H$ $>$
Find y_i *corresponding to* x_i *from* $\mathcal{L}_H$ *and output* y_i *as the answer to the hash query.*
$<$*If* $x_i \notin \mathcal{L}_H$ $>$
Choose $y_i \in Y$ *randomly, add the pair* (x_i, y_i) *to* $\mathcal{L}_H$ *and output* y_i *as the answer to the hash query.*
Leak query: *For a leak query to* H*, output all contents of the hash list.*

3 Security Analysis of FDH in LROM

Full Domain Hash (FDH) [1] is secure signature scheme in ROM. In this section, we consider security of FDH in LROM.

3.1 FDH

FDH is based on trapdoor one-way permutations. The description of FDH is as follows:

Key generation: For input k, output a signing key ($sk = f^{-1}$) and a verification key ($vk = f$) such that $(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k)$ where Dom is the domain of f and $\mathcal{G}$ is a trapdoor permutation generator.

Signature generation: For input a message $m \in \{0,1\}^*$, compute $y = H(m)$ and output a signature $\sigma = f^{-1}(y)$ where $H : \{0,1\}^* \rightarrow Dom$ is a hash function.

Signature verification: For inputs a message m and a signature σ, compute $y' = f(\sigma)$, verify $y' \stackrel{?}{=} H(m)$. If the verification is valid, output 1, otherwise, output 0.

In [1], security of FDH in ROM is proved as follows;

Lemma 1 (Security of FDH in ROM [1]). *If a trapdoor permutation f is one-way, then FDH is existentially unforgeable under adaptively chosen message attacks (EUF-ACMA) where H is modeled as the random oracle.*

3.2 Security of FDH in LROM

We can also prove the security of FDH in LROM like in ROM.

Theorem 1 (Security of FDH in LROM). *If a trapdoor permutation f is one-way, then FDH satisfies EUF-ACMA.*[1]

Proof. Let $\mathcal{F}$ be a forger which breaks EUF-ACMA of FDH. We construct an inverter $\mathcal{I}$ which breaks one-way security of the trapdoor permutation f, i.e., given (f, Dom, y) $\mathcal{I}$ outputs $f^{-1}(y)$. We suppose that $\mathcal{F}$ does not repeat the same query as previous hash queries to the leaky random oracle H or as previous signing queries to the signing oracle SO. Let $\mathcal{L}_H$ be the local hash list of the leaky random oracle H. $\mathcal{L}_H$ consists of tuples $(x_i, H(x_i), z_i)$ $(0 \leq i \leq q_H)$ where z_i is an intermediate value.[2] The concrete construction of $\mathcal{I}$ is as follows. Note that, " $*$ " in a tuple $(x, *, *)$ means wildcard.

Input: (f, Dom, y) s.t. $(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k)$ and $y \stackrel{R}{\leftarrow} Dom$
Output: $f^{-1}(y)$
Step 0: $i^* \stackrel{R}{\leftarrow} \{0, q_H - 1\}$, $\mathcal{L}_H \leftarrow \perp$ ($\perp$ is *null string*), $i \leftarrow 0$.
Step 1: Send f to $\mathcal{F}$ as the input.
Step 2: When $\mathcal{F}$ asks a hash query x_i to H, then behave as follows:
<If $((x_i, *, *) \notin \mathcal{L}_H) \wedge (i^* \neq i)$ >

[1] In this paper, we omit concrete security bounds in the proof owing to lack of space.
[2] The hash list which $\mathcal{F}$ can access has the different form (i.e., the hash list consists of $(x_i, H(x_i))$) than $\mathcal{L}_H$ because z_i is only used for the proof and does not appear in the real protocol.

Generate $z_i \in Dom$ and compute $w_i = f(z_i)$. Add (x_i, w_i, z_i) to $\mathcal{L}_H$ and return w_i to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.
<If $((x_i, *, *) \notin \mathcal{L}_H) \wedge (i^* = i) >$
Generate $w_i \in Dom$. Add $(x_i, y, error)$ to $\mathcal{L}_H$ and return y to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.
<If $(x_i, *, *) \in \mathcal{L}_H >$
Find w' corresponding to x_i from $\mathcal{L}_H$ and return w' to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.

Step 3: When $\mathcal{F}$ asks a signing query x_i to SO, then behave as follows:
<If $(x_i, *, *) \in \mathcal{L}_H >$
Find z' corresponding to x_i from $\mathcal{L}_H$. If $z' = error$, then abort. Otherwise, return z' to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.
<If $(x_i, *, *) \notin \mathcal{L}_H >$
Generate $z_i \in Dom$ and compute $w_i = f(z_i)$. Add (x_i, w_i, z_i) to $\mathcal{L}_H$ and return z_i to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.

Step 4: When $\mathcal{F}$ asks a leak query to H, then hand all pairs of input and output $\{(x, w)\}$ to $\mathcal{F}$. Note that, do not hand intermediate value z.

Step 5: When $\mathcal{F}$ outputs (x^*, σ^*), then check $y \stackrel{?}{=} f(\sigma^*)$. if $y = f(\sigma^*)$, then output σ^* as $f^{-1}(y)$. Otherwise, abort.

We show the success probability of $\mathcal{I}$.

In Step 4, $\mathcal{I}$ has to return the hash list to $\mathcal{F}$ as this simulation is indistinguishable from the output of the leaky random oracle. Then, each output value w is uniformly distributing on Dom because z is uniformly chosen from Dom and f is a permutation. Thus, this simulation is perfect.

$\mathsf{Abort1}$ denote the event which $\mathcal{I}$ aborts for any query in Step 3, $\mathsf{Abort2}$ denote the event which $\mathcal{I}$ aborts in Step 5 and let $\mathsf{Abort} = \mathsf{Abort1} \vee \mathsf{Abort2}$. Then, we estimate the probability which $\mathcal{I}$ does not abort ($\Pr[\neg\mathsf{Abort1}]$ and $\Pr[\neg\mathsf{Abort2}]$).

By the simulation, the event which $\mathcal{I}$ aborts in Step 3 occurs with $\frac{1}{q_H+q_S}$ per every query to the signing oracle. Therefore, the probability that the event which $\mathcal{I}$ does not abort in Step 3 occurs for all queries to the signing oracle ($\Pr[\neg\mathsf{Abort1}]$) is $(1 - \frac{1}{q_H+q_S})^{q_S}$.

By the simulation, the event which $\mathcal{I}$ does not abort in Step 5 occurs with $\frac{1}{q_H}$ because the event occurs only in the case of that $y = f(\sigma^*)$ holds.

Thus, we obtain

$$\begin{aligned}
\epsilon' &= \Pr[\mathbf{Ver}^{FDH}(x^*, \sigma^*, f) = 1 \wedge \neg\mathsf{Abort}] \\
&= \Pr[\mathbf{Ver}^{FDH}(x^*, \sigma^*, f) = 1 | \neg\mathsf{Abort}] \cdot \Pr[\neg\mathsf{Abort}] \\
&= \epsilon \cdot \Pr[\neg\mathsf{Abort}] \\
&= \epsilon \cdot \Pr[\neg\mathsf{Abort1} \wedge \neg\mathsf{Abort2}] \\
&= \epsilon \cdot \Pr[\neg\mathsf{Abort1}] \Pr[\neg\mathsf{Abort2}] \\
&= \epsilon \cdot \big(1 - \frac{1}{q_H + q_S}\big)^{q_S} \cdot \frac{1}{q_H}
\end{aligned}$$

where $\mathbf{Ver}^{FDH}$ is the verification algorithm of FDH, ϵ' is the success probability of $\mathcal{I}$ and ϵ is the success probability of $\mathcal{F}$. □

By the same reason, we can also prove security of PFDH [12] in LROM.

4 Security Analysis of OAEP in LROM

Optimal Asymmetric Encryption Padding (OAEP) [13] is secure padding scheme for asymmetric encryptions in ROM. In this section, we consider security of OAEP in LROM.

4.1 OAEP

OAEP is based on trapdoor partial-domain one-way permutations. We omit the detailed definition of trapdoor partial-domain one-way permutations. Please refer to [14].

The description of OAEP is as follows:

Key generation: For input k, output an encryption key ($ek = f$) and a decryption key ($dk = f^{-1}$) such that $(f, f^{-1}, Dom = \{0,1\}^{k_0} \times \{0,1\}^{k_1}) \leftarrow \mathcal{G}(1^k)$ where $\mathcal{G}$ is a trapdoor permutation generator and $k_0 + k_1 < k$.

Encryption: For input a message $m \in \{0,1\}^n$, generate randomness $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, compute $x = (m||0^{k_1}) \oplus G(r)$ and $y = r \oplus H(x)$, and output a ciphertext $c = f(z)$ for $z = x||y$ where " $||$ " means concatenation, $H : \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$ and $G : \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$ are hash functions, and $n = k - k_0 - k_1$.

Decryption: For inputs a ciphertext c, compute $z = f^{-1}(c)$, parse z as $x||y$ and reconstruct $r = y \oplus H(x)$ where $|x| = n+k_1$ and $|y| = k_0$. If $[x \oplus G(r)]_{k_1} \stackrel{?}{=} 0^{k_1}$ holds, output $m = [x \oplus G(r)]^n$ as the plaintext corresponding to c where $[a]^b$ denotes the b least significant bits of a and $[a]_b$ denotes the b most significant bits of a. Otherwise, reject the decryption as an invalid ciphertext.

In [14], security of OAEP in ROM is proved as follows;

Lemma 2 (Security of OAEP in ROM [14]). *If the trapdoor permutation f is partial-domain one-way, then OAEP satisfies IND-CCA where H and G are modeled as random oracles.*

4.2 Security of OAEP in LROM

OAEP is secure in ROM but, indeed, is insecure in LROM. More specifically, we can show OAEP does not even satisfy one-wayness under chosen-plaintext attacks (OW-CPA) in LROM.

Theorem 2 (Security of OAEP in LROM). *Even if the trapdoor permutation f is partial-domain one-way, OAEP does not satisfy OW-CPA where H and G are modeled as leaky random oracles.*

Proof. We construct an adversary $\mathcal{A}$ which successfully plays OW-CPA game by using leak queries to H and G. Let $\mathcal{L}_H$ and $\mathcal{L}_G$ be hash lists of leaky random oracles H and G respectively. $\mathcal{L}_H$ contains tuples of $(x_i, H(x_i))$ $(0 \leq i \leq q_H - 1)$ and $\mathcal{L}_G$ contains tuples of $(r_j, G(r_j))$ $(0 \leq j \leq q_G - 1)$ where q_H is the number of queries to H and q_G is the number of queries to G. The construction of $\mathcal{A}$ is as follows;

Input : f
Output : m^*
Step 1 : In arbitrary timing, output $(challenge, state)$ and obtain the challenge ciphertext c^* of a plaintext m^*.
Step 2 : Given input f and c^*, ask the leak query to H and G, obtain $\mathcal{L}_H$ and $\mathcal{L}_G$, and compute m^* as follows; For each content $(x_i, H(x_i))$, $(r_j, G(r_j))$ of $\mathcal{L}_H$ and $\mathcal{L}_G$, compute $c' = f(x_i || (r_j \oplus H(x_i)))$. If find the pair $((r^*, G(r^*)), (x^*, H(x^*)))$ such that $((c' = c^*) \wedge ([x_i \oplus G(r_j)]_{k_1} = 0^{k_1}))$ holds, compute $m^* = [x^* \oplus G(r^*)]^n$.
Step 3 : Output m^* as the plaintext of c^*.

Therefore, $\mathcal{A}$ can obtain m^* corresponding to c^*.

We show the success probability of $\mathcal{A}$. When m^* is encrypted to c^*, r^* and x^* such that $x^* = (m^* || 0^{k_1}) \oplus G(r^*)$ are certainly asked to G and H respectively because c^* is generated obeying the protocol description. Thus, $\mathcal{L}_H$ and $\mathcal{L}_G$ contain the pair $((r^*, G(r^*)), (x^*, H(x^*)))$ such that $((c' = c^*) \wedge ([x_i \oplus G(r_j)]_{k_1} = 0^{k_1}))$ holds, and $\mathcal{A}$ can obtain m^* without fail. Therefore, $\mathcal{A}$ successfully plays the OW-CPA game. □

By the similar procedure (i.e., same procedure as the simulation of the decryption oracle in the proof in ROM), we can also show insecurity of Fujisaki-Okamoto conversion [15] in LROM.

5 Security Analysis of Cramer-Shoup cryptosystem in LROM

Cramer-Shoup cryptosystem [16] is secure asymmetric encryption scheme in SM. In this section, we consider security of Cramer-Shoup cryptosystem in LROM.

5.1 Cramer-Shoup Cryptosystem

Cramer-Shoup cryptosystem is based on the Decisional Diffie-Hellman (DDH) assumption and universal one-way hash function family. We omit the definition of DDH assumption and universal one-way hash function. Please refer to [16].

The description of Cramer-Shoup cryptosystem is as follows:

Key generation: For input k, generate a k-bit prime q. Choose $g_1, g_2 \in \mathbb{G}$ randomly and generate a decryption key $(dk = (x_1, x_2, y_1, y_2, z) \in \mathbb{Z}_q^5)$ and public information (c, d, h) such that $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$ and $h = g_1^z$. Next, choose a hash function H from a family of universal one-way hash functions and output an encryption key $ek = (g_1, g_2, c, d, h, H)$ and the decryption key dk.

Encryption: For input a message $m \in \mathbb{G}$, choose $r \in_R \mathbb{Z}_q$, compute $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r m$, $\alpha = H(u_1, u_2, e)$ and $v = c^r d^{r\alpha}$, and output a ciphertext $c = (u_1, u_2, e, v)$.

Decryption: For inputs a ciphertext $c = (u_1, u_2, e, v)$, compute $\alpha = H(u_1, u_2, e)$ and verify whether $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \stackrel{?}{=} v$ holds or not by using $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$. If the verification holds, then output the message $m = \frac{e}{u_1{}^z}$ by using $z \in \mathbb{Z}_q$. Else if, reject the decryption as an invalid ciphertext $\perp$.

In [16], security of Cramer-Shoup cryptosystem in SM is proved as follows;

Lemma 3 (Security of Cramer-Shoup cryptosystem in SM [16]). *If the hash function H is chosen from a family of universal one-way hash functions and the DDH assumption of the group $\mathbb{G}$ holds, then Cramer-Shoup cryptosystem satisfies IND-CCA.*

5.2 Security of Cramer-Shoup Cryptosystem in LROM

We can also prove security of Cramer-Shoup cryptosystem in LROM like in SM.

Theorem 3 (Security of Cramer-Shoup cryptosystem in LROM). *If the DDH assumption of the group $\mathbb{G}$ holds, then Cramer-Shoup cryptosystem satisfies IND-CCA where H is modeled as a leaky random oracle.*

Owing to lack of space, we will give the proof of Theorem 3 in the full version. The outline of the proof is similar to the proof of Lemma 3.

The intuition of the reason why we can prove the security of Cramer-Shoup cryptosystem in LROM as same as SM is as follows; In the case of OAEP, we can construct the successful adversary by applying the simulation of the decryption oracle in the proof in SM. However, in the case of Cramer-Shoup cryptosystem, the simulation of the decryption oracle does not need information of the hash lists. Moreover, all inputs and outputs of hash function H are publicly known because a ciphertext contains (u_1, u_2, e) which are the inputs to the hash function. Naturally, adversaries can know the input and the output in each session. Therefore, the leak query in LROM cannot be advantage of adversaries.

6 Security Analysis of Kurosawa-Desmedt Cryptosystem in LROM

Kurosawa-Desmedt cryptosystem [17] is secure hybrid encryption scheme in SM. In this section, we consider security of Kurosawa-Desmedt cryptosystem in LROM.

6.1 Kurosawa-Desmedt Cryptosystem

Kurosawa-Desmedt cryptosystem is based on the DDH assumption, a special type of universal one-way hash functions, and a symmetric key encryption scheme which satisfies IND-CCA and ϵ-rejection secure for negligible ϵ. We omit the

detailed definition of IND-CCA and ϵ-rejection for symmetric key encryption schemes. Please refer to [17]. The description of Kurosawa-Desmedt cryptosystem is as follows:

Key generation: For input k, randomly choose two distinct generators g_1, g_2 of $\mathbb{G}$ and $(x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, and compute $a = g_1^{x_1} g_2^{x_2}, b = g_1^{y_1} g_2^{y_2}$. Next, choose a hash function H from a family of a special type of universal one-way hash functions, and output an encryption key $ek = (g_1, g_2, a, b, H)$ and the decryption key $dk = (x_1, x_2, y_1, y_2)$.

Encryption: For input a message $m \in \{0,1\}^n$, generate randomness $r \xleftarrow{R} \mathbb{Z}_q$, compute $u_1 = g_1^r$, $u_2 = g_2^r$, $\alpha = H(u_1, u_2)$, $v = a^r b^{r\alpha}$, $K = G(v)$ and the encryption χ of m under the key K using a symmetric key encryption scheme **SKE**, and output a ciphertext $c = (u_1, u_2, \chi)$.

Decryption: For inputs a ciphertext c, compute $\alpha = H(u_1, u_2)$, $v = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$, $K = G(v)$. Next, decrypt χ under the key K using **SKE** and output the resulting decryption.

In [17], security of Kurosawa-Desmedt cryptosystem in SM is proved as follows;

Lemma 4 (Security of Kurosawa-Desmedt cryptosystem in SM [17]) *If the hash function H is chosen from a family of a special type of universal one-way hash functions, the hash function G is uniformly distributed over $\{0,1\}^k$ if v is uniformly distributed over $\mathbb{G}$,* **SKE** *satisfies IND-CCA and ϵ-rejection secure for negligible ϵ, and the DDH assumption of the group $\mathbb{G}$ holds, then Kurosawa-Desmedt cryptosystem satisfies IND-CCA.*

6.2 Security of Kurosawa-Desmedt Cryptosystem in LROM

Kurosawa-Desmedt cryptosystem is secure in ROM but, indeed, is insecure in LROM. More specifically, we can show Kurosawa-Desmedt cryptosystem does not even satisfy OW-CPA in LROM.

Theorem 4 (Security of Kurosawa-Desmedt cryptosystem in LROM). *Even if* **SKE** *satisfies IND-CCA and ϵ-rejection secure for negligible ϵ, and the DDH assumption of the group $\mathbb{G}$ holds, Kurosawa-Desmedt cryptosystem does not satisfy OW-CPA where H and G are modeled as leaky random oracles.*

Proof. We construct an adversary $\mathcal{A}$ which successfully plays OW-CPA game by using the leak query to G. Let $\mathcal{L}_H$ and $\mathcal{L}_G$ be hash lists of leaky random oracles H and G respectively. $\mathcal{L}_H$ contains tuples of $((u_1, u_2)_i, H((u_1, u_2)_i))$ $(0 \le i \le q_H - 1)$ and $\mathcal{L}_G$ contains tuples of $(v_j, G(v_j))$ $(0 \le j \le q_G - 1)$ where q_H is the number of queries to H and q_G is the number of queries to G. The construction of $\mathcal{A}$ is as follows;

Input: g_1, g_2, a, b
Output: m^*
Step 1: Ask the leak query to G, obtain $\mathcal{L}_G$. Then, immediately, output $(challenge, state)$ and obtain the challenge ciphertext $c^* = (u_1, u_2, \chi)$ of a plaintext m^*.

Step 2: Ask again the leak query to G, obtain $\mathcal{L}'_G$, and compare $\mathcal{L}_G$ and $\mathcal{L}'_G$. If there is a content $(v^*, G(v^*))$ in $\mathcal{L}'_G$ but $(v^*, G(v^*))$ is not in $\mathcal{L}_G$, deal with $G(v^*)$ as K^*. Then, decrypt χ under the key K^* using **SKE** and output the resulting decryption m^*.

Step 3: Output m^* as the plaintext of c^*.

Therefore, $\mathcal{A}$ can obtain m^* corresponding to c^*.

We show the success probability of $\mathcal{A}$. When m^* is encrypted to c^*, v^* such that $K^* = G(v^*)$ is certainly asked to G because c^* is generated obeying the protocol description. Thus, $\mathcal{L}_G$ contains $(v^*, G(v^*))$ such that c^* is the ciphertext of the plaintext m^*, and $\mathcal{A}$ can obtain m^* without fail by observing the hash list of G step by step. Therefore, $\mathcal{A}$ successfully plays the OW-CPA game. □

7 Security Analysis of NAXOS in LROM

NAXOS [18] is a secure authenticated key exchange scheme against the leakage of ephemeral private keys (session-specific secret information) in ROM. In this section, we consider security of NAXOS and similar schemes in LROM.

7.1 Security Notion of Authenticated Key Exchange Schemes

Security definitions of authenticated key exchange schemes are studied in many literatures. NAXOS is proven to be secure in the sense of a strong definition, called strong AKE security. Strong AKE security captures various desirable security requirements like resistance to the leakage of ephemeral private keys. We omit the detailed definition of strong AKE security. Please refer to [18]. Here, we define a very weak security notion of authenticated key exchange schemes as follows.

Definition 2 (One-way security against passive attacks). *An authenticated key exchange scheme for parties I and R is one-way secure against passive attacks if the following property holds; For any adversary $\mathcal{A}$,* $\Pr[\,(SK, transcript) \leftarrow \langle I \Leftrightarrow R\rangle;\ SK' \leftarrow \mathcal{A}(transcript);\ SK' = SK] \leq negl.$*, where $\langle I \Leftrightarrow R\rangle$ is a honest execution of the scheme outputting the transcript between I and R and the session key SK.*

Note that, this definition only captures the minimum security requirement for authenticated key exchange schemes.

7.2 NAXOS

NAXOS is based on the Gap Diffie-Hellman (GDH) assumption. We omit the detailed definition of the GDH assumption. Please refer to [19]. The description of NAXOS is as follows:

Interaction: For input k, the parties I and R pick ephemeral secret keys esk_I and esk_R at random from $\{0,1\}^k$. Then the parties exchange values $g^{H(esk_I, sk_I)}$ and $g^{H(esk_R, sk_R)}$ where sk_I and sk_R are static secret keys of I and R respectively, and $H : \{0,1\}^* \to \mathbb{Z}_q$ is a hash function.

Key derivation: The parties check if received values are in the group $\mathbb{G}$ and only compute the session keys if the check succeeds. The session key $SK \in \{0,1\}^k$ is computed as $G(g^{H(esk_R,sk_R)sk_I}, g^{H(esk_I,sk_I)sk_R}, g^{H(esk_I,sk_I)H(esk_R,sk_R)}, I, R)$ where $G : \{0,1\}^* \to \{0,1\}^k$ is a hash function.

In [18], security of NAXOS in ROM is proved as follows;

Lemma 5 (Security of NAXOS in ROM [18]). *If the GDH assumption of the group $\mathbb{G}$ holds, then NAXOS satisfies strong AKE security where H and G are modeled as random oracles.*

7.3 Security of NAXOS in LROM

NAXOS is secure in ROM but, indeed, is insecure in LROM. More specifically, we can show NAXOS does not even satisfy one-way security against passive attacks in LROM.

Theorem 5 (Security of NAXOS in LROM). *Even if the GDH assumption of the group $\mathbb{G}$ holds, NAXOS does not satisfy one-way security against passive attacks where H and G are modeled as leaky random oracles.*

Proof. We construct an passive adversary $\mathcal{A}$ which successfully plays one-way security game by using leak queries to H and G. Let $\mathcal{L}_H$ and $\mathcal{L}_G$ be hash lists of leaky random oracles H and G respectively. $\mathcal{L}_H$ contains tuples of $((esk, sk)_i, H((esk, sk)_i))$ $(0 \le i \le q_H - 1)$ and $\mathcal{L}_G$ contains tuples of $((G_1, G_2, G_3, ID_1, ID_2)_j, G((G_1, G_2, G_3, ID_1, ID_2)_j))$ $(0 \le j \le q_G - 1)$ where q_H is the number of queries to H and q_G is the number of queries to G. The construction of $\mathcal{A}$ is as follows;

Input: $transcript^* = (g^{H(esk_I^*,sk_I^*)}$ and $g^{H(esk_R^*,sk_R^*)})$

Output: SK^*

Step 1: Ask the leak query to H and obtain $\mathcal{L}_H$. For each content $((esk, sk)_i, H((esk, sk)_i))$ of $\mathcal{L}_H$, if find the pair i_1 and i_2 such that $g^{H((esk,sk)_{i_1})} = g^{H(esk_I^*,sk_I^*)}$ and $g^{H((esk,sk)_{i_2})} = g^{H(esk_R^*,sk_R^*)}$ then compute $G_1^* = (g^{H(esk_R^*,sk_R^*)})^{sk_{i_1}}$, $G_2^* = (g^{H(esk_I^*,sk_I^*)})^{sk_{i_2}}$ and $G_3^* = g^{H((esk,sk)_{i_1})H((esk,sk)_{i_2})}$.

Step 2: Ask the leak query to G and obtain $\mathcal{L}_G$. For each content $((G_1, G_2, G_3, ID_1, ID_2)_j, G((G_1, G_2, G_3, ID_1, ID_2)_j))$ of $\mathcal{L}_G$, if find j such that $G_1 = G_1^*$, $G_2 = G_2^*$ and $G_3 = G_3^*$ then deal with $G((G_1, G_2, G_3, ID_1, ID_2)_j)$ as SK^*.

Step 3: Output SK^* as the session key.

Therefore, $\mathcal{A}$ can obtain SK^* of the challenge session.

We show the success probability of $\mathcal{A}$. When SK^* is generated, $(esk, sk)_{i_1}$, $(esk, sk)_{i_2}$ and $(G_1^*, G_2^*, G_3^*, I, R)_j$ such that $g^{H((esk,sk)_{i_1})} = g^{H(esk_I^*,sk_I^*)}$, $g^{H((esk,sk)_{i_2})} = g^{H(esk_R^*,sk_R^*)}$, $G_1 = G_1^*$, $G_2 = G_2^*$ and $G_3 = G_3^*$ are certainly asked to H and G because SK^* is generated obeying the protocol description. Thus, $\mathcal{L}_G$ contains $((G_1, G_2, G_3, ID_1, ID_2)_j, G((G_1, G_2, G_3, ID_1, ID_2)_j))$ such that $SK^* = G((G_1, G_2, G_3, ID_1, ID_2)_j)$, and $\mathcal{A}$ can obtain SK^* without fail. Therefore, $\mathcal{A}$ successfully plays the one-way security game under the passive attack. □

By the similar procedure (i.e., obtaining all secret information), we can also show insecurity of CMQV [20] in LROM.

8 Discussion

8.1 Difference of Effects on Security

In LROM, though FDH and Cramer-Shoup cryptosystem can be proven security, OAEP, Kurosawa-Desmedt cryptosystem and NAXOS are insecure.

Our attack to OAEP in LROM is based on the simulation of the decryption oracle in the proof of Lemma 2. Most of asymmetric encryption schemes which are secure in ROM realize the simulation of the decryption oracle in the proof without knowledge of the decryption key by using contents of hash lists. Therefore, by the same behavior as the simulator in the proof in ROM, the adversary can decrypt the challenge ciphertext without knowledge of the decryption key because the adversary can observe contents of the hash list in LROM. Our attack to Kurosawa-Desmedt cryptosystem is simpler than one to OAEP. Since Kurosawa-Desmedt cryptosystem can be proven security in SM, the simulation of the decryption oracle does not need to use contents of hash lists. However, the hash value of G has to be secret for external entities because the hash value is used the key of symmetric key encryption part. Thus, if contents of the hash list are leaked, the plaintext of any ciphertext is easily decrypted by adversaries. Also, NAXOS falls into the similar condition as Kurosawa-Desmedt cryptosystem, i.e., all secret information of parties are leaked from the hash list.

On the other hand, FDH is different with these protocols in the following points; Firstly, though we have to simulate the signing oracle without knowledge of the signing key in the security proof of FDH, the simulation does not need to use contents of the hash list. Secondly, in the signing procedure the input of the hash function and the corresponding hash value are not secret information because the input is the message to be signed. Therefore, if contents of the hash list is available to adversaries, it does not become any advantage of adversaries. Thus, we can prove security of FDH in LROM. The case of Cramer-Shoup cryptosystem is also similar to the case of FDH. Since Cramer-Shoup cryptosystem can be proven security in SM, we can simulate the decryption oracle without contents of the hash list. Moreover, in the encryption procedure the inputs of the hash function and the corresponding hash value are not secret information because the inputs are contained in the ciphertext. Thus, we can prove security of Cramer-Shoup cryptosystem in LROM.

Hence, in order to prove security of a protocol in LROM, it is important that we can realize all the simulation without contents of the hash list, and the input of the hash function and the corresponding hash value are not secret information.

8.2 Relation between the Standard Model

From the modeling, the proof of a protocol in LROM implies the proof of the protocol in ROM trivially. Moreover, LROM is independent from SM. Our result of analyses shows the separation between LROM and SM because of two

following observations; Firstly, FDH is secure in LROM under the assumption of trapdoor permutation. However, Dodis et al. [6] showed that FDH is not provable in SM under the same assumption. Thus, we obtain that the proof of a protocol in LROM does not implies the proof of the protocol in SM.

Next, Kurosawa-Desmedt cryptosystem is secure in SM under the DDH assumption, the assumption of universal hash function family and the assumption of symmetric key encryption. However, Kurosawa-Desmedt cryptosystem is insecure in LROM by instantiating hash functions by leaky random oracles under same assumptions. Thus, we obtain that the proof of a protocol in SM does not implies the proof of the protocol in LROM.

Therefore, LROM is independent from SM. We have to check whether a new protocol is secure in LROM even if the protocol is known to be secure in SM.

8.3 Relation between Randomness Revealing

At first sight, it may seem that the leakage of contents of the hash list is corresponding to randomness revealing because it often happen that the inputs of hash functions contains local randomness. Thus, it may seem that LROM is same as ROM under randomness revealing. Indeed, these are different. Our result of analyses shows the difference between LROM and ROM under randomness revealing because of the following observation;

NAXOS is secure in ROM under the leakage of local randomness. However, NAXOS is insecure in LROM even if there is no leakage of local randomness. Thus, we obtain that the proof of a protocol in ROM under randomness revealing does not implies the proof of the protocol in LROM.

Therefore, we have to check whether a new protocol is secure or not in LROM even if the protocol is known to be secure in ROM under randomness revealing.

9 Further Works

A remaining problem of future works is more detailed analyses of protocols under the leakage. For example, though OAEP is insecure if both random oracles H and G are instantiated by leaky random oracles, OAEP may be secure if either of two random oracles is only instantiated by the leaky random oracle. Indeed, Boldyreva and Fischlin [21] showed that OAEP is secure if either of two random oracles is instantiated by the real hash function and the other remain as the random oracle.

References

1. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security 1993, pp. 62–73 (1993)
2. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited (Preliminary Version). In: STOC 1998, pp. 131–140 (1998)
3. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. J. ACM 51(4), 557–594 (2004)

4. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
5. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest We Remember: Cold Boot Attacks on Encryption Keys. In: 17th USENIX Security Symposium, pp. 45–60 (2008)
6. Dodis, Y., Oliveira, R., Pietrzak, K.: On the Generic Insecurity of the Full Domain Hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
7. Nielsen, J.B.: Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
8. Liskov, M.: Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
9. Pasini, S., Vaudenay, S.: Hash-and-Sign with Weak Hashing Made Secure. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 338–354. Springer, Heidelberg (2007)
10. Unruh, D.: Random Oracles and Auxiliary Input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007)
11. Numayama, A., Isshiki, T., Tanaka, K.: Security of Digital Signature Schemes in Weakened Random Oracle Models. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 268–287. Springer, Heidelberg (2008)
12. Coron, J.S.: Optimal Security Proofs for PSS and Other Signature Schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
13. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
14. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP Is Secure under the RSA Assumption. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 260–274. Springer, Heidelberg (2001)
15. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
16. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
17. Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
18. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Provsec 2007, pp. 1–16 (2007)
19. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
20. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. In: Des. Codes Cryptography, vol. 46(3), pp. 329–342 (2008)
21. Boldyreva, A., Fischlin, M.: Analysis of Random Oracle Instantiation Scenarios for OAEP and Other Practical Schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 412–429. Springer, Heidelberg (2005)

How to Use Merkle-Damgård — On the Security Relations between Signature Schemes and Their Inner Hash Functions

Emmanuel Bresson[1], Benoît Chevallier-Mames[1], Christophe Clavier[2], Aline Gouget[3], Pascal Paillier[3], and Thomas Peyrin[4]

[1] DCSSI, 51 bd. De la Tour Maubourg, 75700 Paris Cedex 07, France
[2] Gemalto, 6 rue de la Verrerie, 92190 Meudon, France
[3] Gemalto, Avenue du Jujubier, 13705 La Ciotat, France
[4] Orange Labs, 38-40 rue du Général Leclerc, 92794 Issy-les-Moulineaux, France

Abstract. This paper reports a thorough standard-model investigation on how attacks on hash functions impact the security of *hash-and-sign* signature schemes. We identify two important properties that appear to be crucial in analyzing the nature of security relations between signature schemes and their inner hash functions: primitiveness and injectivity. We then investigate the security relations in the general case of *hash-and-sign* signatures and in the particular case of *first-hash-then-sign* signatures, showing a gap of security guarantees between the two paradigms. We subsequently apply our results on two operating modes to construct a hash function family from a hash function based on the well-known *Merkle-Damgård* construction (such as MD5 and SHA-1). For completeness, we give concrete attack workloads for attacking operating modes used in practical implementations of signature schemes.

1 Introduction

Undoubtedly, hash functions constitute an essential component of all sorts of systems and constructions in cryptography. Recently, a number of attacks on practical hash functions such as MD5 or SHA-1 have been reported [4,24,27]. The role played by hash functions in the overall security of a system is, in many constructions, so badly understood that it is not obvious at all to see how that system suffers from being based on broken or weakened hash functions.

The goal of our work is to explore the interplay between the security of a cryptographic system $\mathcal{S} = \mathcal{S}[H_1, \ldots, H_n]$ based on hash functions $H_1, \ldots, H_n$ and the security of $H_1, \ldots, H_n$. Assuming for instance that a scheme $\mathcal{S}$ involves a unique hash function H, we want to determine how the security of H relates to the one of $\mathcal{S}$. This amounts to computationally connect security notions for $\mathcal{S}$ with security notions for H. These connections between $\mathcal{S}$ and H can be categorized into four types: *attack*, *security proof*, *impossibility of attack* and *impossibility of security proof*. In this paper, we focus on the first two links.

In Section 2, we give several definitions on provable security and on hash function and signature security notions. In Section 3, we analyze the security link between signature schemes and hash functions, for two types of signature schemes we define. In

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 241–253, 2008.

Section 4, we refine our study on the special case of Merkle-Damgård construction. As expressed in the conclusion, the security relations we expose in this paper confirm the intuition that all signature schemes are *not* implicated on the same level by the recent attacks on hash functions.

2 Preliminaries

2.1 Provable Security Statements

We adopt the concrete-security setting [20], as opposed to the asymptotic one. A concrete black-box reduction $\mathcal{R}$ between two computational problems P_1 and P_2 is a probabilistic algorithm $\mathcal{R}$ which solves P_1 given black-box access to an oracle which solves P_2. We write $P_1 \Leftarrow_{\mathcal{R}} P_2$ when $\mathcal{R}$ is known to reduce P_1 to P_2 with efficient (or tight) reduction cost. $P_1 \Leftarrow P_2$ states that an *efficient* $\mathcal{R}$ exists such that $P_1 \Leftarrow_{\mathcal{R}} P_2$ and $P_1 \Leftrightarrow P_2$ means $P_1 \Leftarrow P_2$ and $P_1 \Rightarrow P_2$. All reductions considered in this paper are concrete and fully black-box.

A provable-security statement $P_1 \Leftarrow P_2$, where P_1, P_2 are two computational problems, indicates that if an efficient solver for P_2 exists then an efficient solver for P_1 exists as well, which is meaningful when P_1 is believed to be hard. When efficient solvers for P_1 are known to exist, by contrast, we claim that $P_1 \Leftarrow P_2$ is still meaningful if *(a)* an efficient solver for P_1 exists but is *unknown*[1], *(b)* the proof of $P_1 \Leftarrow P_2$ is *constructive* meaning that an explicit black-box reduction $\mathcal{R}$ is given.

2.2 Hash Functions and Related Security Notions

In the following, $\{0,1\}^*$ denotes the set of finite bitstrings, and $\{0,1\}^{\mathsf{m}}$ denotes the set of m-bit strings. A function H is a *hash function* if it maps $\{0,1\}^*$ to $\{0,1\}^{\mathsf{m}}$ for some integer $\mathsf{m} > 0$ called the output size of H. For $n > 0$, let U_n be the uniform distribution over $\{0,1\}^n$ and for any $0 \leq t \leq 2^n$, let us pose $S_{n,t} = \{m \in \{0,1\}^{\mathsf{m}} : |H^{-1}(m) \cap \{0,1\}^n| = t\}$. We define the bias of H, denoted by $\delta_n(H)$, as the statistical L_1-distance between $H(U_n)$ and U_{m}, *i.e.* $\delta_n(H) = \|H(U_n) - U_{\mathsf{m}}\|_1 = \sum_{t=0}^{2^n} |S_{n,t}| \left|\frac{t}{2^n} - \frac{1}{2^{\mathsf{m}}}\right|$.

A *hash function family* is a function $F : \{0,1\}^* \times \{0,1\}^{\mathsf{r}} \to \{0,1\}^{\mathsf{m}}$ for integers $\mathsf{m}, \mathsf{r} > 0$ such that for any $r \in \{0,1\}^{\mathsf{r}}$, $F(\cdot, r)$ is a hash function of output size m. The second argument r in $F(M, r)$ is called the *index* (or *key*) of F. Let $n > 0$, we define two statistical bias for F: $\delta_n(F) = \|F(U_n, U_{\mathsf{r}}) - U_{\mathsf{m}}\|_1$ and $\delta_n^+(F) = \max_{r \in \{0,1\}^{\mathsf{r}}} \|F(U_n, r) - U_{\mathsf{m}}\|_1$ (hence $\delta_n(F) \leq \delta_n^+(F)$).

We adopt the convention that security problems are parameterized by the bitsize of their inputs (written as subscripts) and/or their outputs (written as superscripts). The classical security notions for hash function are preimage resistance (PRE), 2nd-preimage resistance (SEC) and Collision resistance (COL).

	Given	Find	Such that
$\mathsf{PRE}^n[H]$	m	M	$H(M) = m$
$\mathsf{SEC}_{n_1}^{n_2}[H]$	M_1	$M_2 (\neq M_1)$	$H(M_2) = H(M_1)$
$\mathsf{COL}^{n_1,n_2}[H]$	–	M_1, M_2	$M_1 \neq M_2$ and $H(M_1) = H(M_2)$
where $m \in \{0,1\}^{\mathsf{m}}, M \in \{0,1\}^n, M_i \in \{0,1\}^{n_i}$			

[1] This relates to Rogaway's human ignorance paradigm, see [20].

When extending the security notions PRE, SEC and COL to hash function families, three definitional variants appear which we call existential (E-$\cdots$), universal (U-$\cdots$) and absolute (A-$\cdots$), as indicated below.

	Given	Find	Such that
E-PRE^{n} $[F]$	m	M, r	$F(M,r) = m$
U-PRE^{n} $[F]$	m, r	M	$F(M,r) = m$
E-SEC$^{n_2}_{n_1}$ $[F]$	M_1	M_2, r	$F(M_2,r) = F(M_1,r)$
U-SEC$^{n_2}_{n_1}$ $[F]$	M_1, r	M_2	$F(M_2,r) = F(M_1,r)$
A-SEC$^{n_2}_{n_1}$ $[F]$	M_1	M_2	$\forall r, F(M_2,r) = F(M_1,r)$
E-COL^{n_1,n_2} $[F]$	–	M_1, M_2, r	$F(M_2,r) = F(M_1,r)$
U-COL^{n_1,n_2} $[F]$	r	M_1, M_2	$F(M_2,r) = F(M_1,r)$
A-COL^{n_1,n_2} $[F]$	–	M_1, M_2	$\forall r, F(M_2,r) = F(M_1,r)$

where $m \in \{0,1\}^{\mathsf{m}}, M \in \{0,1\}^{n}, M_i \in \{0,1\}^{n_i}, r \in \{0,1\}^{\mathsf{r}}, M_1 \neq M_2$

The relations between the security notions for a hash function (resp. a hash function family) are summarized in Theorem 1.

Theorem 1. *Let H be a hash function. Then, for any $n_1, n_2 > 0$, we have:*

$$\mathsf{COL}^{n_1,n_2}[H] \Leftarrow \mathsf{SEC}^{n_2}_{n_1}[H] \Leftarrow^{\star} \mathsf{PRE}^{n_2}[H] ,$$

where the $\Leftarrow^{\star}$ reduction is tight only if H has sufficiently small bias[2] *$\delta_{n_1}(H)$. Let F be a hash function family. Then, for any $n_1, n_2 > 0$, we have:*

$$\begin{array}{ccccc}
\mathsf{E\text{-}PRE}^{n_2}[F] & \Leftarrow & \mathsf{U\text{-}PRE}^{n_2}[F] & & \\
 & & \Downarrow^{\star} \text{ if } \delta^{+}_{n_1}(F) \text{ is small enough} & & \\
\mathsf{E\text{-}SEC}^{n_2}_{n_1}[F] & \Leftarrow & \mathsf{U\text{-}SEC}^{n_2}_{n_1}[F] & \Leftarrow & \mathsf{A\text{-}SEC}^{n_2}_{n_1}[F] \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{E\text{-}COL}^{n_1,n_2}[F] & \Leftarrow & \mathsf{U\text{-}COL}^{n_1,n_2}[F] & \Leftarrow & \mathsf{A\text{-}COL}^{n_1,n_2}[F]
\end{array}$$

Most of the relations given in Theorem 1 are rather intuitive and can be proved using very standard reduction techniques.

2.3 Signature Schemes and Related Security Notions

A signature scheme Σ with message space $\mathcal{M} \subseteq \{0,1\}^*$ is defined as $\Sigma \triangleq (\Sigma.\mathsf{Gen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Ver})$ such that $\Sigma.\mathsf{Gen}()$ is a probabilistic algorithm that outputs a pair of strings $(\mathsf{pk}, \mathsf{sk})$, a signature on message $M \in \mathcal{M}$ is an s-bit string $\sigma = \Sigma.\mathsf{Sign}(\mathsf{sk}, M, u)$ where $u \leftarrow \{0,1\}^{\mathsf{u}}$, and $\Sigma.\mathsf{Ver}(\mathsf{pk}, M, \sigma)$ outputs 1 if $\sigma = \Sigma.\mathsf{Sign}(\mathsf{sk}, M, u)$ for some $u \in \{0,1\}^{\mathsf{u}}$, 0 otherwise.

We now take a closer look at the inner computations of a signature scheme by defining the notion of *two-step signature* which seems totally unrestrictive as we do not know any example of a *non*-two-step signature scheme.

Definition 2 (Two-step signature scheme). *A signature scheme $\Sigma \triangleq (\Sigma.\mathsf{Gen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Ver})$ with message space $\mathcal{M} \subseteq \{0,1\}^*$ is said to be* two-step *if there exist four deterministic algorithms Σ_1, Σ_2, Υ_1 and Υ_2 such that:*

[2] Well-known counterexamples can be constructed when this condition is not fulfilled, see [17].

(i) *For any pair* $(M, u) \in \mathcal{M} \times \{0,1\}^{\mathsf{u}}$, $\Sigma.\mathsf{Sign}(\mathsf{sk}, M, u) = \Sigma_2(\mathsf{sk}, M, r, \mathsf{aux})$ *where* $(r, \mathsf{aux}) = \Sigma_1(\mathsf{sk}, u)$, $\mathsf{aux} \in \{0,1\}^*$ *and* $r \in \{0,1\}^{\mathsf{r}}$.
(ii) $\Sigma.\mathsf{Ver}(\mathsf{pk}, M, \sigma) = \Upsilon_2(\mathsf{pk}, M, \sigma, \hat{r})$ *where* $\hat{r} = \Upsilon_1(\mathsf{pk}, \sigma)$;
(iii) *If there exists* $u \in \{0,1\}^{\mathsf{u}}$ *such that* $\sigma = \Sigma.\mathsf{Sign}(\mathsf{sk}, M, u)$ *then* $r = \hat{r}$.

The type *of* Σ *is* $(\mathsf{m}, \mathsf{u}, \mathsf{r}, \mathsf{s})$ *if* $\mathcal{M} = \{0,1\}^{\mathsf{m}}$ *and* $(*, \mathsf{u}, \mathsf{r}, \mathsf{s})$ *if* $\mathcal{M} = \{0,1\}^*$.

We now properly define the notion of *hash-and-sign* signature that captures virtually all known signature schemes.

Definition 3 (Hash-and-Sign). *A* hash-and-sign signature scheme *is a pair* $\mathcal{S} = \langle F, \Sigma \rangle$ *where* $F : \{0,1\}^* \times \{0,1\}^{\mathsf{r}} \to \{0,1\}^{\mathsf{m}}$ *is a hash function family and* Σ *is a two-step signature scheme of type* $(\mathsf{m}, \mathsf{u}, \mathsf{r}, \mathsf{s})$ *for integers* $\mathsf{u}, \mathsf{s} > 0$. *The key generation of* $\mathcal{S}$ *is identical to the one of* Σ. *Let* $\Sigma_1, \Sigma_2, \Upsilon_1, \Upsilon_2$ *be the inner functions of* Σ. *A signature on* $M \in \{0,1\}^*$ *is computed as* $\sigma = \Sigma_2(\mathsf{sk}, m, r, \mathsf{aux})$ *where* $m = F(M, r)$, $(r, \mathsf{aux}) = \Sigma_1(\mathsf{sk}, u)$ *and* $u \leftarrow \{0,1\}^{\mathsf{u}}$. $\mathcal{S}.\mathsf{Ver}(\mathsf{pk}, M, \sigma)$ *first computes* $\hat{r} = \Upsilon_1(\mathsf{pk}, \sigma)$, *then* $\hat{m} = F(M, \hat{r})$ *and outputs* $\Upsilon_2(\mathsf{pk}, \hat{m}, \sigma, \hat{r})$.

We finally consider the notion of *first-hash-then-sign* signatures as a particular case of hash-and-sign signatures. Both paradigms are depicted on Fig. 1. A first-hash-then-sign signature is defined to be a hash-and-sign signature where the index space of F is not used *i.e.* $F(M, r) = H(M)$. Hence the hash function family collapses to a single hash function. The notion of first-hash-then-sign signature captures several well-known signature designs including Full Domain hash (FDH) and Boneh-Lynn-Shacham (BLS).

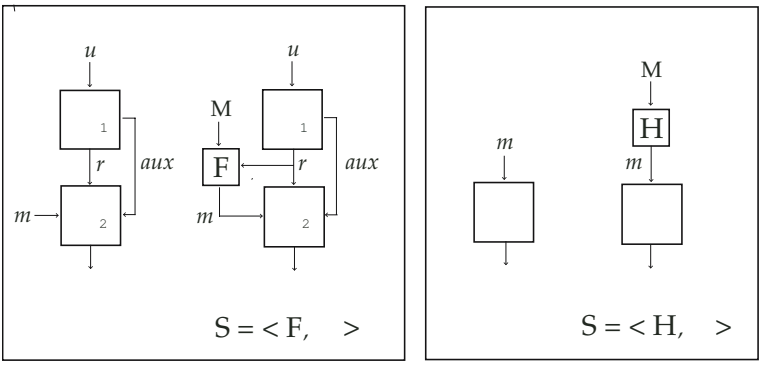

Fig. 1. The hash-and-sign (left) and first-hash-then-sign (right) paradigms

We extend the common security notions of unforgeability, elaborating on the seminal work of Goldwasser *et al.* [10]. We consider the three usual adversarial resources: *Key Only Attack* (KOA) where the adversary is given nothing but a public key, *Known Message Attack* (KMA), where the adversary is given a list of random and pairwise distinct message-signature pairs $(M_1, \sigma_1), \ldots, (M_\ell, \sigma_\ell)$, and *Chosen Message Attack* (CMA) where the adversary is given adaptive access to a signing oracle. KMA_n denotes the restriction of KMA to messages $M_1, \ldots, M_\ell$ uniformly distributed over $\{0,1\}^n$ (we

define CMA_n similarly). We distinguish four adversarial goals: *Universal Forgery* (UF), *Existential Forgery* (EF), *Universal Repudiation* (UR) and lastly *Existential Repudiation* (ER) defined as indicated below.

	Given	Find	Such that
UF UF_n	$M \in \mathcal{M}$ $M \in \{0,1\}^n$	σ σ	σ is a valid signature on the message M
EF EF_n	– –	$M \in \mathcal{M}, \sigma$ $M \in \{0,1\}^n, \sigma$	σ is a valid signature on the message M
UR $\mathsf{UR}_{n_1}^{n_2}$	$M_1 \in \mathcal{M}$ $M_1 \in \{0,1\}^{n_1}$	$M_2 \in \mathcal{M}, \sigma$ $M_2 \in \{0,1\}^{n_2}, \sigma$	$M_2 \neq M_1$ and σ is valid on both M_1 and M_2
ER ER^{n_1,n_2}	– –	$M_1 \in \mathcal{M}, M_2 \in \mathcal{M}, \sigma$ $M_1 \in \{0,1\}^{n_1}, M_2 \in \{0,1\}^{n_2}, \sigma$	$M_2 \neq M_1$ and σ is valid on both M_1 and M_2
where $n, n_1, n_2 > 0$			

We view security notions as computational problems; this allows to relate these notions using reductions. For instance $\mathsf{UF\text{-}KMA}\,[\mathcal{S}]$ is the problem of computing a universal forgery under known message attack. In the case of KMA or CMA, we denote by $\ell\text{-}\mathsf{GOAL\text{-}ATK}\,[\mathcal{S}]$ the problem of breaking GOAL in no more than ℓ calls to the resource defined by ATK. In the case of UR and ER, no adversarial resource is defined, as the adversary can choose the secret/public key pair. Fig. 2 displays the well-known black-box (and constructive) reductions among security levels. Indices n_1, n_2 reflecting the case $\mathcal{M} = \{0,1\}^*$ are to be removed if $\mathcal{M} = \{0,1\}^{\mathsf{m}}$.

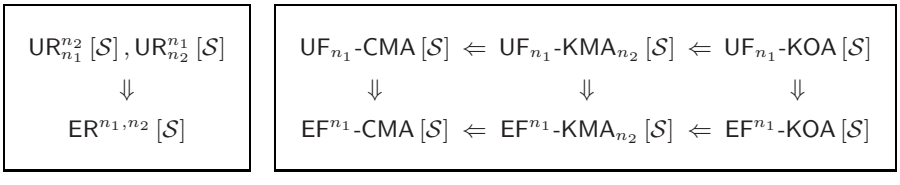

Fig. 2. Black-box relations among security notions for signature schemes

3 Analyzing Security Relations for Hash-and-Sign Signatures

In this section, we first identify two core properties that will be crucial in analyzing security relations between a signature scheme and its inner hash function. We then explicit how some attacks on hash functions can be turned into attacks on signature schemes, and on the positive side, how some attacks on signatures are impossible unless some attacks on hash functions can be made efficient.

3.1 Identified Properties

It appears that most signature schemes used in practice, beside being hash-and-sign schemes, also fulfill two other properties that we call *primitiveness* and *injectivity*.

Definition 4 (Primitiveness). *Let $\mathcal{S} = \langle F, \Sigma \rangle$ be a probabilistic hash-and-sign signature scheme and let Σ_1, Σ_2, Υ_1, Υ_2 be the inner functions of Σ. $\mathcal{S}$ is said to be primitive when a probabilistic algorithm $\mathcal{S}.\mathsf{Prim}$ is known which, for any key pair $(\mathsf{pk}, \mathsf{sk})$, takes pk as input and outputs a random pair $(m, \sigma = \Sigma.\mathsf{Sign}(\mathsf{sk}, m, u))$ such that (a) m is uniformly distributed over $\{0,1\}^{\mathsf{m}}$; (b) u is uniformly distributed over $\{0,1\}^{\mathsf{u}}$.*

Note that $\mathcal{S}$ being primitive implies that Σ is existentially forgeable. In essence, if $\mathcal{S} = \langle F, \Sigma \rangle$ is primitive then replacing F by the identity function *structurally* destroys the EF-KOA security of the modified scheme. This notion also captures the intuition that F somehow plays an important role in the EF-KOA security of $\mathcal{S}$. Furthermore, primitiveness is often the property that allows simulatability and makes security proofs possible in the random oracle model.

Definition 5 (Injectivity). *Let $\mathcal{S} = \langle F, \Sigma \rangle$ be a probabilistic hash-and-sign signature scheme and Σ_1, Σ_2, Υ_1, Υ_2 as above. $\mathcal{S}$ is said to be injective when for any key pair $(\mathsf{pk}, \mathsf{sk})$ and any $\sigma \in \{0,1\}^{\mathsf{s}}$, there exists at most one pair $(m, r) \in \{0,1\}^{\mathsf{m}} \times \{0,1\}^{\mathsf{r}}$ such that $\sigma = \Sigma_2(\mathsf{sk}, m, r, \mathsf{aux})$ and $(r, \mathsf{aux}) = \Sigma_1(\mathsf{sk}, u)$ for some u, aux.*

We review and characterize some of the most common signature schemes. Our classification results are summarized on Table 1.

Table 1. A classification of common signature schemes

Hash-and-Sign Signature Scheme	First-Hash-then-Sign	Primitive	Injective
Schnorr [22]		X	X
FDH [2]	X	X	X
PFDH [6]		X	X
PSS [3]		X	X
EMSA-PSS [3]	X	X	X
BLS [5]	X	X	X
Generic DSA [1,19]	X		X
ElGamal Type I [17]	X	X	X
ElGamal Type II [17]		X	X
GQ [11]		X	X
GHR [9]	X		X
CS [7]	X		

3.2 Attacks and Positive Security Relations for Hash-and-Sign Signatures

Our goal is to relate the security of $\mathcal{S} = \langle F, \Sigma \rangle$ to the one of its hash component F, the underlying signature scheme Σ being seen as a fixed parameter. The first type of connection between security notions for $\mathcal{S}$ with security notions for H is referred to as an *attack*. An attack is defined as an efficient reduction $\mathsf{Break}(H) \Rightarrow \mathsf{Break}(\mathcal{S})$ that makes explicit how an attack of a given type on a hash function H is enough to break the scheme $\mathcal{S}$ in a prescribed way. The attacks we found are summarized in Lemma 6.

Lemma 6 (Black-box attacks). *Let $\mathcal{S} = \langle F, \Sigma \rangle$ be a hash-and-sign signature scheme as per Definition 3. Then for any integers $n_1, n_2 > 0$,*

$$\mathsf{A\text{-}COL}^{n_1,n_2}[F] \Rightarrow 1\text{-}\mathsf{EF}^{n_1}\text{-}\mathsf{CMA}[\mathcal{S}], 1\text{-}\mathsf{EF}^{n_2}\text{-}\mathsf{CMA}[\mathcal{S}] , \quad (1)$$

$$\mathsf{U\text{-}COL}^{n_1,n_2}[F] \Rightarrow \mathsf{ER}^{n_1,n_2}[\mathcal{S}] , \quad (2)$$

$$\mathsf{A\text{-}SEC}^{n_2}_{n_1}[F] \Rightarrow 1\text{-}\mathsf{UF}_{n_1}\text{-}\mathsf{CMA}[\mathcal{S}] , \quad (3)$$

$$\mathsf{U\text{-}SEC}^{n_2}_{n_1}[F] \Rightarrow 1\text{-}\mathsf{EF}^{n_2}\text{-}\mathsf{KMA}_{n_1}[\mathcal{S}], \mathsf{UR}^{n_2}_{n_1}[\mathcal{S}] . \quad (4)$$

Lemma 7 (The case of primitive signatures). *Let $\mathcal{S} = \langle F, \Sigma\rangle$ be a hash-and-sign signature scheme and assume that $\mathcal{S}$ is primitive. Then in addition to the above we have, for any $n > 0$,*

$$\mathsf{U\text{-}PRE}^{n}[F] \Rightarrow \mathsf{EF}^{n}\text{-}\mathsf{KOA}[\mathcal{S}] .$$

We now state positive security relations between $\mathcal{S}$ and F. A *security proof* is defined as an efficient reduction $\mathsf{Break}(H) \Leftarrow \mathsf{Break}(\mathcal{S})$ which means that there is no attack of a certain type on $\mathcal{S}$ unless one finds a particular weakness in H. It seems unlikely (although we do not disprove it) that security proofs exist which relate the unforgeability of $\mathcal{S}$ to F in the general case. However, assuming that $\mathcal{S}$ is injective is enough to show that non-repudiation can be guaranteed under security assumptions on F.

Lemma 8 (The case of injective signatures). *Let $\mathcal{S}$ be a probabilistic hash-and-sign signature scheme and assume $\mathcal{S}$ is injective. Then for any $n_1, n_2 > 0$,*

$$\mathsf{E\text{-}COL}^{n_1,n_2}[F] \Leftarrow \mathsf{ER}^{n_1,n_2}[\mathcal{S}] \quad \textit{and} \quad \mathsf{E\text{-}SEC}^{n_2}_{n_1}[F] \Leftarrow \mathsf{UR}^{n_2}_{n_1}[\mathcal{S}] .$$

The proofs of Lemmas 7 and 8 for hash-and-sign signature schemes can be found in Appendix A.

3.3 Attacks and Security Proof for First-Hash-Then-Sign Signatures

We now move on to the particular case of first-hash-then-sign signatures. Our goal is now to exhaust the security reductions standing between a first-hash-then-sign scheme $\mathcal{S} = \langle H, \Sigma\rangle$ and its inner hash function H. Again, the signature scheme Σ plays the role of a parameter here.

The reductions stated in Lemmas 6 and 7, which show how breaking certain security properties of F allows to break $\mathcal{S}$, can be simplified in the particular case of a first-hash-then-sign scheme $\mathcal{S} = \langle H, \Sigma\rangle$. In this case indeed, in addition to the above, it holds that for any integers $n_1, n_2 > 0$:

$$\mathsf{COL}^{n_1,n_2}[H] \Rightarrow 1\text{-}\mathsf{EF}^{n_1}\text{-}\mathsf{CMA}[\mathcal{S}], 1\text{-}\mathsf{EF}^{n_2}\text{-}\mathsf{CMA}[\mathcal{S}], \mathsf{ER}^{n_1,n_2}[\mathcal{S}] \quad (5)$$

$$\mathsf{SEC}^{n_2}_{n_1}[H] \Rightarrow 1\text{-}\mathsf{UF}_{n_1}\text{-}\mathsf{CMA}[\mathcal{S}], 1\text{-}\mathsf{EF}^{n_2}\text{-}\mathsf{KMA}_{n_1}[\mathcal{S}], \mathsf{UR}^{n_2}_{n_1}[\mathcal{S}] \quad (6)$$

In the same way, the positive security relations between $\mathcal{S}$ and F presented in Lemma 8 can be adapted to the particular case of first-hash-then-sign signatures. A direct corollary of these results and the relations of (5) and (6) is a security proof for injective signatures meaning that the levels of repudiation of $\mathcal{S}$ can be guaranteed under security assumptions on H.

Lemma 9 (Security proofs for injective signatures). *Let $\mathcal{S}$ be a first-hash-then-sign signature scheme and assume $\mathcal{S}$ is injective. Then for any $n_1, n_2 > 0$,*

$$\mathsf{SEC}^{n_2}_{n_1}[H] \Leftrightarrow \mathsf{UR}^{n_2}_{n_1}[\mathcal{S}] \quad \textit{and} \quad \mathsf{COL}^{n_1,n_2}[H] \Leftrightarrow \mathsf{ER}^{n_1,n_2}[\mathcal{S}] \ .$$

In some sense, the general hash-and-sign paradigm inherently offers better security guarantees than the particular case of first-hash-then-sign signatures. For instance, breaking $\mathcal{S}$ in the EF-CMA sense is easy if H is not collision-resistant in the case $\mathcal{S} = \langle H, \Sigma \rangle$. Breaking the same security level for $\mathcal{S} = \langle F, \Sigma \rangle$, however, seems to require the generation of absolute collisions for F, a much stronger weakness. Overall, given a fixed-size signature scheme Σ, it seems largely preferable to domain-extend it in the general hash-and-sign paradigm for which security is obtained under much weaker assumptions on the security of the inner hash component than in the first-hash-then-sign paradigm.

4 Merkle-Damgård-Based Hash Function Families

This section specifically considers Merkle-Damgård (MD) embodiments of F; this is motivated by the fact that most practical hash functions rely on some variant of the MD construction. We then apply the results presented in Section 3 to analyze two practical instantiations of F using a MD-hash function. Based on these reductions, we display the effective workload of the best known attacks against F in every instantiation.

4.1 Hash Function Families Based on Merkle-Damgård

Before considering Merkle-Damgård hash functions, we first relate the security of a hash function family F, constructed from an arbitrary hash function H to the one of H. There are many ways one may attempt to construct a hash function family F using a hash function H. The two most "natural" constructions are $F(M, r) = H(M \| r)$ and $F(M, r) = H(r \| M)$.

Construction of a Hash Function Family. Let $H : \{0,1\}^* \to \{0,1\}^{\mathsf{m}}$ be a hash function. We construct a wide class of hash function families as follows. Let F be the hash function family defined on $M \in \{0,1\}^*$ and $r \in \{0,1\}^{\mathsf{r}}$ (the index bitsize r being arbitrary) as $F(M, r) = H(\llbracket M, r \rrbracket)$, where $\llbracket M, r \rrbracket$ is an $(|M| + \mathsf{r})$-bit (one-to-one) encoding of the pair $(M, r) \in \{0,1\}^* \times \{0,1\}^{\mathsf{r}}$. Then, for any $n_1, n_2 > 0$, we have :

$$\begin{aligned} \mathsf{PRE}^{n_2+\mathsf{r}}[H] &\Leftrightarrow \mathsf{E\text{-}PRE}^{n_2}[F] \\ \mathsf{SEC}^{n_2+\mathsf{r}}_{n_1+\mathsf{r}}[H] &\Leftarrow \mathsf{U\text{-}SEC}^{n_2}_{n_1}[F] \\ \mathsf{COL}^{n_1+\mathsf{r},n_2+\mathsf{r}}[H] &\Leftarrow \mathsf{E\text{-}COL}^{n_1,n_2}[F] \end{aligned}$$

Iterative Hash Functions. Most hash functions used in practice are *iterative hash functions* built from *compression functions*. Let $f : \{0,1\}^{\mathsf{m}} \times \{0,1\}^{\mathsf{b}} \to \{0,1\}^{\mathsf{m}}$ be a compression function. Given an arbitrary function $g : \{0,1\}^* \to (\{0,1\}^{\mathsf{b}})^*$ and $IV_0 \in \{0,1\}^{\mathsf{m}}$, the iterated hash function H constructed from (f, g, IV_0) is the hash function defined for $M \in \{0,1\}^*$ by $H(M) = h_t$ where $g(M) = X_1 || \ldots || X_t$, $h_0 = IV_0$ and $h_i = f(h_{i-1}, X_i)$ for $i \in [1..t]$. We call b the block size of H and we write $H = \mathsf{ITER}[f, g, IV_0]$.

MD-based Hash Functions. The basic and strengthened Merkle-Damgård constructions [8,18] are a specific case of iterated hashing using specific functions $g = g_0$ or $g = g_s$ defined as follows. For any $M \in \{0,1\}^*$, $g_0(M)$ is obtained by appending to M as many 0-bits as necessary to yield a $\lceil |M|/\mathsf{b} \rceil$-block string. Let $\mathsf{a} < \mathsf{b}$ be a size parameter. Given $M \in \{0,1\}^*$, $g_s(M)$ is formed by appending a single 1-bit to M and as many 0-bits as required so that the length of the so-obtained string is congruent to $(\mathsf{b} - \mathsf{a})$ modulo b. Then the bitlength of M, seen as an a-bit string, is appended.

Appending a logical length-block prior to hashing, called MD-*strengthening*, aims at preventing collision and pseudo-collision attacks which find colliding messages of different length (e.g. see [12,14]). The protection it seems to offer justifies the practical use of MD-strengthening in common hash functions.

Let us now fix $f : \{0,1\}^{\mathsf{m}} \times \{0,1\}^{\mathsf{b}} \to \{0,1\}^{\mathsf{m}}$ and $IV_0 \in \{0,1\}^{\mathsf{m}}$. The related Merkle-Damgård hash function without MD-strengthening is the hash function $H_0 = \mathsf{ITER}[f, g_0, IV_0]$. Its counterpart with MD-strengthening is the hash function $H_s = \mathsf{ITER}[f, g_s, IV_0]$.

Definition 10 (Collision-propagating paddings). *Let $k_1, k_2 > 0$. A padding function $g : \{0,1\}^* \to (\{0,1\}^{\mathsf{b}})^*$ is said to be (k_1, k_2)-collision-propagating when for any compression function $f : \{0,1\}^{\mathsf{m}} \times \{0,1\}^{\mathsf{b}} \to \{0,1\}^{\mathsf{m}}$ and any $IV_0 \in \{0,1\}^{\mathsf{m}}$, the hash function $H = \mathsf{ITER}[f, g, IV_0]$ is such that for any k_1-block string $M_1 \in \{0,1\}^{k_1 \cdot \mathsf{b}}$ and k_2-block string $M_2 \in \{0,1\}^{k_2 \cdot \mathsf{b}}$, if $H(M_1) = H(M_2)$ then $H(g(M_1)\|M) = H(g(M_2)\|M)$ for any $M \in \{0,1\}^*$.*

Proposition 11. *It holds that g_0 is a (k_1, k_2)-collision-propagating padding for any $k_1, k_2 > 0$ and g_s is a (k, k)-collision-propagating padding for any $k > 0$.*

Hence, it is rather obvious that MD-strengthening is not enough to thwart attacks based on propagating collisions. In fact, all known second preimage or collision-finding attacks against MD-hash functions with MD-strengthening generate messages with the same block length. It turns out that for any $n = k \cdot \mathsf{b} > 0$, we have $\mathsf{PRE}^{n+b}[H_0] \Leftarrow \mathsf{PRE}^n[H_s]$, $\mathsf{SEC}^n_n[H_s] \Leftarrow \mathsf{SEC}^n_n[H_0]$ and $\mathsf{COL}^{n,n}[H_s] \Leftarrow \mathsf{COL}^{n,n}[H_0]$. We restrict ourselves to instantiations of F based on $H_s = \mathsf{ITER}[f, g_s, IV_0]$, knowing in advance that all recent SEC and COL attacks against H_0, which retrieve same-length colliding messages, equally apply to the simplified setting $H_0 = \mathsf{ITER}[f, g_0, IV_0]$.

4.2 MD Instantiation with Operating Mode $F(M, r) = H_s(M \,\|\, r)$

Let $H_s = \mathsf{ITER}[f, g_s, IV_0]$ be an MD hash function with MD strengthening and F defined on $\{0,1\}^* \times \{0,1\}^{\mathsf{r}}$ as $F(M, r) = H_s(M\|r)$. Then for any $n = k \cdot \mathsf{b} > 0$,

$$\begin{array}{ccccccc} \mathsf{A\text{-}SEC}^{n+\mathsf{b}}_{n+\mathsf{b}}[F] & \Leftarrow & \mathsf{A\text{-}SEC}^n_n[F] & \Leftarrow & \mathsf{SEC}^n_n[H_0] & \Rightarrow & \mathsf{SEC}^n_n[H_s] \\ \Downarrow & & & & \Downarrow & & \\ \mathsf{A\text{-}COL}^{n+\mathsf{b},n+\mathsf{b}}[F] & \Leftarrow & \mathsf{COL}^{n,n}[H_s] & \Leftarrow & \mathsf{COL}^{n,n}[H_0] & & \end{array}$$

It is easily seen that a hash-and-sign signature scheme $\mathcal{S} = \langle F, \Sigma \rangle$ based on the operating mode $F(M, r) = H_s(M\|r)$ yields in reality a first-hash-then-sign scheme on message subspaces $M \in \{0,1\}^{k \cdot \mathsf{b}}$. Indeed posing $H_{s,IV} = \mathsf{ITER}[f, g_s, IV]$, it is

obvious that if $|M| = k \cdot \mathsf{b}$ then $H_s(M\|r) = H_{s,m}(r)$ where $m = H_{0,IV_0}(M)$. Thus, on these subspaces, $\sigma = \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, M, u) = \Sigma_2(\mathsf{sk}, F(M,r), r, \mathsf{aux})$ where $(r, \mathsf{aux}) = \Sigma_1(\mathsf{sk}, u)$ can be reformulated as $\sigma = \Sigma_2(\mathsf{sk}, H_{s,m}(r), r, \mathsf{aux}) = \Sigma_2'(\mathsf{sk}, m, u)$ where $m = H_0(M)$.

Lemma 12. *Let $\mathcal{S} = \langle F, \Sigma\rangle$. If $F(M,r) = H_s(M\|r)$ then $\mathcal{S} = \langle H_0, \Sigma'\rangle$ on message subspaces $\{0,1\}^{k\cdot\mathsf{b}}$ with $k > 0$ for some signature scheme Σ'.*

Thus, in the case $F(M,r) = H_s(M\|r)$, the security benefits inherent to using the general hash-and-sign paradigm are completely lost. For any $n = k \cdot \mathsf{b} > 0$, one has

$$1\text{-}\mathsf{EF}^n\text{-}\mathsf{CMA}\,[\mathcal{S}]\,,\ \mathsf{ER}^n\,[\mathcal{S}] \ \Leftarrow\ \mathsf{COL}^{n,n}\,[H_0]\ ,$$
$$1\text{-}\mathsf{UF}_n\text{-}\mathsf{CMA}\,[\mathcal{S}]\,,\ 1\text{-}\mathsf{EF}^n\text{-}\mathsf{KMA}\,[\mathcal{S}]\,,\ \mathsf{UR}_n\,[\mathcal{S}] \ \Leftarrow\ \mathsf{SEC}_n^n\,[H_0]\ .$$

4.3 MD Instantiation with Operating Mode $F(M,r) = H_s(r \,\|\, M)$

When looking at constructions such as $F(M,r) = H_s(r\|M)$, we first examine the "absolute"-like properties, $\mathsf{A}\text{-}\mathsf{SEC}_n^n\,[F]$ and $\mathsf{A}\text{-}\mathsf{COL}^{n,n}\,[F]$. Following the definition, finding absolute collisions means that the mappings $r \mapsto F(M_i, r)$ are the same for two messages M_1, M_2. We can provide an upper-bound for the security by noticing that there is at most 2^n such mappings (viewed as parameterized by M of length n). The total number of r-bit-to-m-bit mappings being $(2^{\mathsf{m}})^{2^{\mathsf{r}}}$, two mappings coincide with reasonable probability as soon as $n \approx \mathsf{m}2^{\mathsf{r}}$, which is beyond practical lengths.

Claim. Let F be defined as $F(M,r) = H_s(r\|M)$ and let $\mathcal{S} = \langle F, \Sigma\rangle$. Combining the security statements of Lemmas 6 and 7, we conclude that there is **no known way** to break $\mathcal{S}$ in any sense **even** if H_0 is $\mathsf{COL}^{n,n}$-, SEC_n^n- and PRE^n- broken.

Clearly, this strongly suggests to prefer the second operating mode over the first one in practice.

4.4 Concrete Security Figures for Two Instantiations of $F(M,r)$

We display on Table 2 concrete security workloads for effectively attacking $F(m,r)$ under current knowledge for the two instantiations $F(m,r) = H(M\|r)$ and $F(m,r) = H(r\|M)$, where H is chosen to be MD4, MD5, SHA-0 or SHA-1. We evaluate the expected running time τ_{H_s} of the attack algorithm when imposing a success probability ε_{H_s} heuristically close to one. For instance, for $F(M,r) = \mathrm{MD5}(M\|r)$, $\mathsf{U}\text{-}\mathsf{COL}^{n,n}\,[F]$ is known to be $(2^{30}, \simeq 1)$-broken for $n = 2 \cdot 512$ *i.e.* for two-block messages. The message bitsize n is fixed to $2 \cdot 512$ for all figures except for MD4 where $n = 1 \cdot 512$.

The best attacks known are Sasaki's new message difference for MD4 [21], Klima's tunnels for MD5 [13], Wang's attack on SHA-0 [26], and the latest improvements over Wang's SHA-1 attack [27,16]. Complexity 2^{60} for SHA-1 is (announced) in [16]. Complexity 2^{33} for SHA-0 is given in [15]. The target collision attack against MD5 by Stevens et al. [23] can be used to generate solutions for $\mathsf{E}\text{-}\mathsf{SEC}\,[F]$ if r is long enough (at least 4192 bits). A similar attack can certainly be done on MD4. ∞ indicates perfect security. Empty cells denote that we are not aware of an attack more efficient than generic attacks.

Table 2. Expected running time of attacks

	$H_s(M\|\|r)$				$H_s(r\|\|M)$			
	MD4	MD5	SHA-0	SHA-1	MD4	MD5	SHA-0	SHA-1
A-SEC$_n^n$ $[F]$					∞	∞	∞	∞
E-SEC$_n^n$ $[F]$		2^{52}			2^{58}			
A-COL^{n,n} $[F]$	2^1	2^{30}	2^{33}	2^{60}	∞	∞	∞	∞
U-COL^{n,n} $[F]$	2^1	2^{30}	2^{33}	2^{60}	2^1	2^{30}	2^{33}	2^{60}
E-COL^{n,n} $[F]$	2^1	2^{30}	2^{33}	2^{60}	2^1	2^{30}	2^{33}	2^{60}

5 Conclusion

This investigation on how recent attacks on hash functions may impact on signature schemes is particularly fruitful. We have first properly defined hash-and-sign signatures and the particular case of first-hash-then-sign signatures, and identified two core properties; injectivity and primitiveness. For each of these categories (and notably for schemes such as FDH, PSS and Schnorr), we have shown how their security relates to the one of their inner hash component. We then focused on the case of hash functions based on the Merkle-Damgård domain extender (such as MDx and SHAx). Enlightening the security properties of popular operating modes, *i.e.* the ones one would try first to construct a hash function family, we identified $F(M, r) = H_s(M\|r)$ as (by far) the worst operating mode in terms of security guarantees. Thus a concrete conclusion of our work is the suggestion of using $F(M, r) = H_s(r\|M)$ or more intricate operating modes.

Acknowledgments

This work is part of the SAPHIR project (http://www.crypto-hash.fr), partly funded by the French National Research Agency (ANR). The authors would like to thank all the members of the project for fruitful discussions.

References

1. ANSI X9.62-1998. Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (1998)
2. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
3. Bellare, M., Rogaway, P.: PSS: Provably secure encoding method for digital signatures. In: IEEE P1363a (submission) (August 1998)
4. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and Reduced SHA-1. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)

5. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
6. Coron, J.-S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
7. Cramer, R., Shoup, V.: Signature Schemes Based on the Strong RSA Assumption. In: ACM Conference on Computer and Communications Security, pp. 46–51 (1999)
8. Damgård, I.: A Design Principle for Hash Functions. In: McCurley, K.S., Ziegler, C.D. (eds.) Advances in Cryptology 1981 - 1997. LNCS, vol. 1440, pp. 416–427. Springer, Heidelberg (1999)
9. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signature without the random oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
10. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. of Computing 17(2), 281–308 (1988)
11. Guillou, L., Quisquater, J.-J.: A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
12. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
13. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute, http://eprint.iacr.org/2006/105
14. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
15. Manuel, S., Peyrin, T.: Collisions on SHA-0 in one hour. FSE 2008 (to appear, 2008)
16. Mendel, F., Rechberger, C., Rijmen, V.: Update on SHA-1. In: Rump Session of Crypto 2007 (2007)
17. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
18. Merkle, R.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
19. National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS — Publication 186 (May 1994)
20. Rogaway, P.: Formalizing Human Ignorance. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)
21. Sasaki, Y., Wang, L., Ohta, Y., Kunihiro, N.: New messages difference for MD4. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 329–348. Springer, Heidelberg (2007)
22. Schnorr, C.P.: Efficient signatures generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
23. Stevens, M., Lenstra, A., de Weger, B.: Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
24. Wang, X.Y., Yu, H.B.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
25. Wang, X.Y., Lai, X.J., Feng, D., Chen, H., Yu, X.: Cryptanalysis for Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
26. Wang, X.Y., Yu, H.B., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
27. Wang, X.Y., Yin, Y.L., Yu, H.B.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

A Proofs of Lemmas 7 and 8

We adopt the concrete-security setting [20], as opposed to the asymptotic one. Given a computational problem P, a probabilistic algorithm is said to (τ, ε)-solve P when it halts after at most τ elementary steps and outputs a solution of P with success probability at least ε. A concrete black-box[3] reduction $\mathcal{R}$ between two computational problems P_1 and P_2 is a probabilistic algorithm $\mathcal{R}$ which (τ_1, ε_1)-solves P_1 given black-box access to an oracle (τ_2, ε_2)-solving P_2.

A.1 Proof of Lemma 7

Proof ($\mathsf{U\text{-}PRE}^n[F] \Rightarrow \mathsf{EF}^n\text{-}\mathsf{KOA}[\mathcal{S}]$*).* Let us assume a black-box access to $\mathcal{A}_F$ which (τ_F, ε_F)-breaks $\mathsf{U\text{-}PRE}^n[F]$. We build an EF^n-KOA adversary $\mathcal{A}_\mathcal{S}$ which $(\tau_\mathcal{S}, \varepsilon_\mathcal{S})$-breaks $\mathcal{S}$ where $\varepsilon_\mathcal{S} = \varepsilon_F$ and $\tau_\mathcal{S} = \tau_F + \mathsf{Time}(\mathcal{S}.\mathsf{Prim}) + \mathsf{Time}(\mathcal{S}.\mathsf{Ver})$. $\mathcal{A}_\mathcal{S}$ is given a random key $\mathsf{pk} \leftarrow \mathcal{S}.\mathsf{Gen}()$. Since $\mathcal{S}$ is primitive, $\mathcal{A}_\mathcal{S}$ can generate a random pair $(m, \sigma = \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m, u))$ by running $\mathcal{S}.\mathsf{Prim}(\mathsf{pk})$. Note that m is uniformly distributed over $\{0,1\}^{\mathsf{m}}$ and u is uniform over $\{0,1\}^{\mathsf{u}}$, thereby making $r = \Upsilon_1(\mathsf{pk}, \sigma)$ uniform over $\{0,1\}^{\mathsf{r}}$. Now $\mathcal{A}_\mathcal{S}$ runs $\mathcal{A}_F(m, r)$ to construct $M \in \{0,1\}^n$ such that $F(M, r) = m$. $\mathcal{A}_\mathcal{S}$ then outputs (M, σ). □

A.2 Proof of Lemma 8

Proof ($\mathsf{E\text{-}COL}^{n_1,n_2}[F] \Leftarrow \mathsf{ER}^{n_1,n_2}[\mathcal{S}]$*).* Let us assume an adversary $\mathcal{A}_\mathcal{S}$ which $(\tau_\mathcal{S}, \varepsilon_\mathcal{S})$-breaks $\mathsf{ER}^{n_1,n_2}[\mathcal{S}]$. We build an existential collision-finder $\mathcal{A}_F$ which (τ_F, ε_F)-breaks $\mathsf{E\text{-}COL}^{n_1,n_2}[F]$ with $\varepsilon_F = \varepsilon_\mathcal{S}$ and $\tau_F = \tau_\mathcal{S} + \mathsf{Time}(\mathcal{S}.\mathsf{Gen}) + +\mathsf{Time}(\mathcal{S}.\mathsf{Ver})$. $\mathcal{A}_F$ generates a random key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{S}.\mathsf{Gen}()$ and runs $\mathcal{A}_\mathcal{S}(\mathsf{pk}, \mathsf{sk})$. If $\mathcal{A}_\mathcal{S}$ outputs (M_1, M_2, σ) where $M_1 \in \{0,1\}^{n_1}$, $M_2 \in \{0,1\}^{n_2}$, $M_2 \neq M_1$ and σ is a valid signature on M_1 and M_2, then $\mathcal{A}_F$ computes $r = \Upsilon_1(\mathsf{pk}, \sigma)$ and outputs (M_1, M_2, r). If σ is a signature on M_1 and M_2 simultaneously then $\sigma = \Sigma_2(\mathsf{sk}, F(M_1, r_1), r_1, \mathsf{aux}_1) = \Sigma_2(\mathsf{sk}, F(M_2, r_2), r_2, \mathsf{aux}_2)$ for some $\mathsf{aux}_1, \mathsf{aux}_2$ and $r_1, r_2 \in \{0,1\}^{\mathsf{r}}$. Since $\mathcal{S}$ is injective, one must have $r_1 = r_2 = r$ and $F(M_1, r) = F(M_2, r)$ so that (M_1, M_2, r) is an (n_1, n_2)-existential collision. □

Proof ($\mathsf{E\text{-}SEC}^{n_2}_{n_1}[F] \Leftarrow \mathsf{UR}^{n_2}_{n_1}[\mathcal{S}]$*).* Assuming a $(\tau_\mathcal{S}, \varepsilon_\mathcal{S})$-universal repudiator $\mathcal{A}_\mathcal{S}$, we build an algorithm $\mathcal{A}_F$ which (τ_F, ε_F)-breaks $\mathsf{E\text{-}SEC}^{n_2}_{n_1}[F]$ with $\varepsilon_F = \varepsilon_\mathcal{S}$ and $\tau_F = \tau_\mathcal{S} + \mathsf{Time}(\mathcal{S}.\mathsf{Gen}) + \mathsf{Time}(\mathcal{S}.\mathsf{Ver})$. Given a random message $M_1 \leftarrow \{0,1\}^{n_1}$, $\mathcal{A}_F$ generates a random key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{S}.\mathsf{Gen}()$ and runs $\mathcal{A}_\mathcal{S}(\mathsf{pk}, \mathsf{sk}, M_1)$ to construct a pair (M_2, σ) where $M_2 \in \{0,1\}^{n_2}$, $M_2 \neq M_1$ and σ is a valid signature on M_1 and M_2. In this case, $\mathcal{A}_F$ computes $r = \Upsilon_1(\mathsf{pk}, \sigma)$ and outputs (M_2, r). σ being a signature on both M_1 and M_2 implies $\sigma = \Sigma_2(\mathsf{sk}, F(M_1, r_1), r_1, \mathsf{aux}_1) = \Sigma_2(\mathsf{sk}, F(M_2, r_2), r_2, \mathsf{aux}_2)$ for some $\mathsf{aux}_1, \mathsf{aux}_2$ and $r_1, r_2 \in \{0,1\}^{\mathsf{r}}$. Since $\mathcal{S}$ is injective, one must have $r_1 = r_2 = r$ and $F(M_1, r) = F(M_2, r)$ so that (M_2, r) yields an n_2-bit second preimage of $F(M_1, r)$. □

[3] All reductions considered in this paper are concrete and fully black-box.

Can We Construct Unbounded Time-Stamping Schemes from Collision-Free Hash Functions?

Ahto Buldas[1,2,3] and Margus Niitsoo[1,3,⋆]

[1] University of Tartu, Liivi 2, 50409 Tartu, Estonia
ahto.buldas@ut.ee
[2] Tallin University of Technology, Raja 15, 12618 Tallinn, Estonia
[3] Cybernetica AS, Akadeemia tee 21, 12618 Tallinn, Estonia
margus.niitsoo@cyber.ee

Abstract. It has been known for quite some time that collision-resistance of hash functions does not seem to give any actual security guarantees for unbounded hash-tree time-stamping, where the size of the hash-tree created by the time-stamping service is not explicitly restricted. We focus on the possibility of showing that there exist no black-box reductions of unbounded time-stamping schemes to collision-free hash functions. We propose an oracle that is probably suitable for such a separation and give strong evidence in support of that. However, the existence of a separation still remains open. We introduce the problem and give a construction of the oracle relative to which there seem to be no secure time-stamping schemes but there still exist collision-free hash function families. Although we rule out many useful collision-finding strategies (relative to the oracle) and the conjecture seems quite probable after that, there still remains a possibility that the oracle can be abused by some very smartly constructed wrappers. We also argue why it is probably very hard to give a correct proof for our conjecture.

1 Introduction

Suppose you are an inventor and you just had a brilliant idea. You want to protect yourself against someone later claiming to have had that same idea, but earlier. If he honestly claims so there is nothing you can do. To avoid dishonest claims of this type, it would be sufficient to tie your idea to the time at which you discovered it, i.e., to *time-stamp* the idea. Simplistic time-stamping schemes use trusted authorities that receive documents, add time stamps to them and sign the time-stamped documents digitally. Assuming that the signature scheme is secure and the authority is trustworthy, this is a good scheme and has indeed been used in paperwork for hundreds of years—notaries and patent offices follow such a scheme.

Over the centuries, inventors who did not want to reveal their ideas to authorities discovered many ingenious ways of securely time-stamping documents. For example, sending documents to themselves in sealed envelopes and keeping them sealed until the documents should be used as evidence in courts. A digital analog of this scheme

⋆ Both authors supported by Estonian SF grant no. 6944, by EU FP6-15964: "AEOLUS", and by the European Regional Development Fund.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 254–267, 2008.

is meant to send a hash value of the invention to the central authority instead of the document. Assuming that the hash function is hard to inverse, no useful information about the invention is revealed. When the time stamp needs to be verified, the full document is presented and everyone is able to check that the time-stamped hash value represents this particular document.

Even this is insufficient for the most paranoid inventors. The main concern is that it is rather easy for corrupted authorities to issue *back-dated* time stamps, so that it might be possible to create a document today and claim that it was created yesterday. Hence, it is reasonable to require that even the authority itself is unable to forge time stamps. The first time-stamping scheme that achieved this was proposed by Haber and Stornetta [6] and was extensively scrutinized [1,7,2]. The main idea of their scheme was to publish the hash value of the document in daily newspapers instead of letting the authorities sign them. As it is almost impossible to change the contents of yesterday's newspapers the scheme is secure without any trustworthiness assumptions. To increase the efficiency of this scheme, all hash values that were intended to be published can be hashed together to a single hash value by using Merkle hash trees [10] instead of publishing all hash values separately.

It was widely believed that the security of such a hash-and-published scheme is a straightforward implication of the collision-freeness of the hash function until it was shown by Buldas and Saarepera [4] that this is not the case—the conventional black box techniques will certainly fail to give such proofs. It has also been shown [3,5] that the collision-freeness property is probably unnecessary for creating secure hash-and-publish time-stamping schemes.

The main aim of this paper is to explore the possibility that no constructions of secure hash functions for this scheme could be made from collision resistant functions. Section 2 introduces the required notions and definitions. Section 3 describes the hash-and-publish time-stamping scheme and the security condition that is required from the hash function. Section 4 gives an overview of cryptographic reductions and how to prove their impossibility via oracle separation. Section 5 gives a description of an oracle that is the most promising candidate for the separation at hand. Next sections try to argue why this oracle is probably insufficient for finding collisions to every single hash function. Section 6 discusses the possibility of using disperser graphs to abuse the oracle. Section 7 shows that such type of adversary and also a much wider class of adversaries will fail to do so if the oracle is carefully chosen. Sections 8 rules out some more abusing-strategies. Section 10 discusses the possibility of moving towards to the complete solution of the conjecture.

2 Notation and Definitions

If $P(x)$ is a predicate of some kind, then $[P(x)]$ is 1 if $P(x)$ holds and 0 otherwise (the Iverson symbol). By P_n we denote the set of unordered pairs from $\{0,1\}^n$. By $x \leftarrow \mathcal{D}$ we mean that x is chosen randomly according to a distribution $\mathcal{D}$. If A is a probabilistic function or a Turing machine, then $x \leftarrow \mathsf{A}(y)$ means that x is chosen according to the output distribution of A on an input y. By $\mathcal{U}_n$ we denote the uniform distribution on $\{0,1\}^n$. If $\mathcal{D}_1, \ldots, \mathcal{D}_m$ are distributions and $F(x_1, \ldots, x_m)$ is a

predicate, then $\Pr[x_1 \leftarrow \mathcal{D}_1, \ldots, x_m \leftarrow \mathcal{D}_m : F(x_1, \ldots, x_m)]$ denotes the probability that $F(x_1, \ldots, x_m)$ is true after the ordered assignment of $x_1, \ldots, x_m$. For functions $f, g : \mathbb{N} \to \mathbb{R}$, we write $f(k) = O(g(k))$ if there are $c, k_0 \in \mathbb{R}$, so that $f(k) \leq cg(k)$ $(\forall k > k_0)$. We write $f(k) = \omega(g(k))$ if $\lim_{k\to\infty} \frac{g(k)}{f(k)} = 0$. If $f(k) = k^{-\omega(1)}$, then f is *negligible*. A Turing machine M is *polynomial-time* (*poly-time*) if it runs in time $k^{O(1)}$, where k denotes the input size.

By an *oracle Turing machine* we mean an incompletely specified Turing machine S that comprises calls to *oracles*. The description can be completed by defining the oracle as a function $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$. In this case, the machine is denoted by $S^{\mathcal{O}}$. The function $y \leftarrow \mathcal{O}(x)$ is not necessarily computable but may still have assigned a conditional running time $t(x)$, which does not reflect the actual amount of computations needed to produce y from x. The running time of $S^{\mathcal{O}}$ comprises the conditional running time of oracle calls – each call $\mathcal{O}(x)$ takes $t(x)$ steps. An oracle $\mathcal{O}$ is *poly-time* if $t(x) = |x|^{O(1)}$, where $|x|$ denotes the bit-length of x. We say that S is a *poly-time oracle machine* if $S^{\mathcal{O}}$ runs in poly-time, whenever $\mathcal{O}$ is poly-time. A *primitive* $\mathcal{P}$ is a class of (not necessarily computable by ordinary Turing machines) functions intended to perform a security related task (e.g. data confidentiality, integrity etc.). Each primitive $\mathcal{P}$ is characterized by the success $\delta(k)$ of an adversary A. An instance f of a primitive $\mathcal{P}$ is *secure* if every poly-time adversary can break f only with a negligible success. Let $\mathcal{O}$ be an oracle. We say that f is *secure relative to* $\mathcal{O}$ if every poly-time oracle adversary $\mathsf{A}^{\mathcal{O}}$ can break f only with negligible success. A distribution family $\{\mathcal{D}_k\}_{k\in\mathbb{N}}$ is *poly-sampleable* if there is a poly-time D with output distribution $\mathcal{D}(1^k) \equiv \mathcal{D}_k$. We call functions of type $h : \{0,1\}^n \to \{0,1\}^m$ *hash functions* if $n - m = \omega(\log(n))$. We say that $x_1, x_2 \in \{0,1\}^n$ form a *collision* for $h : \{0,1\}^n \to \{0,1\}^m$ if $x_1 \neq x_2$ and $h(x_1) = h(x_2)$. We say that a family χ of hash functions $h : \{0,1\}^n \to \{0,1\}^m$ $(n > m)$ is *collision resistant* if for every poly-time adversary A can find collisions for a randomly chosen $h \leftarrow \chi$ with negligible probability, i.e., for every adversary A we have $\Pr[h \leftarrow \chi, (x_0, x_1) \leftarrow \mathsf{A}(h) : x_0 \neq x_1, h(x_0) = h(x_1)] = m^{-\omega(1)}$.

3 Hash and Publish Time-Stamping

A hash-and-publish time-stamping involves three parties: a Client C, a Server S and a repository $\mathcal{R}$ and gives two procedures - one for creating a time stamp and one for verifying it. Inspired by real newspapers, $\mathcal{R}$ is assumed to be *write-once*, i.e., if a hash value r was once published in $\mathcal{R}$, it cannot be deleted or changed later. The server S is a middle-man between clients and $\mathcal{R}$ and is not assumed to be trusted.

The scheme uses two functions: a *client side hash function* $h_c : \{0,1\}^* \to \{0,1\}^k$ used by clients to get hash values of the documents and a *server side hash function* $h : \{0,1\}^{2k} \to \{0,1\}^k$ that is used for computing the published hash value. In this work, we only study the properties of h by assuming that each client request x_i is a hash $h_c(X_i)$ of a document X_i of arbitrary length. We model h as a function with two inputs and one output of length k and write $h(x_1, x_2) = y$, where $x_1, x_2, y \in \{0,1\}^k$. Hash trees can then be represented as circuits where each *wire* (output of an h-gate) carries k bits simultaneously. A hash circuit transforms a list $x_1, \ldots, x_m$ of k-bit inputs (client requests) to a single k-bit output r_t (*root value*), where t is the time unit (*round*)

during which the requests were received by the server. Having received the requests $x_1, \ldots, x_m$, the server computes r_t and publishes it in the write-once repository $\mathcal{R}$. We assume that the shape of the hash tree is chosen by S and is not restricted anyhow. This is why the scheme is called *unbounded*.

After publishing r_t in $\mathcal{R}$, the server sends each client a *certificate*, that is a 4-tuple $c = (x, t, n, z)$, where x is the request, t is the number of the round, $n = n_1 n_2 \ldots n_l$ is a bit-string that describes the path from x down to the root and $z = (z_1, \ldots, z_l) \in (\{0,1\}^k)^l$ is the list of sibling hash values that are needed to verify the path. For example, the certificate of x_4 in Fig. 1 is $c = (x_4, t, 101, (z_1, z_2, z_3))$.

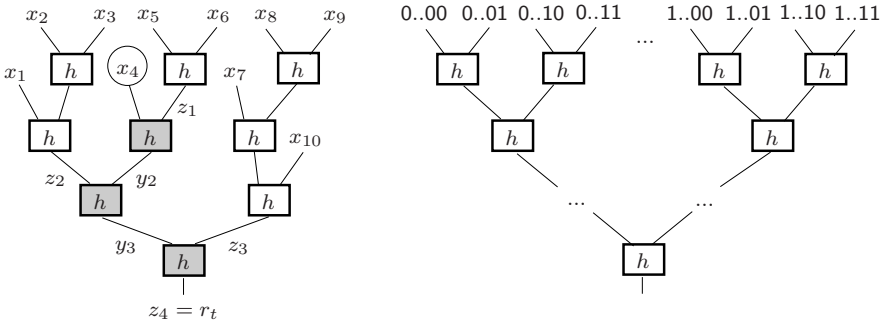

Fig. 1. A tree with a path marked from x_4 (left) and the complete hash tree with 2^k leaves (right)

The verification procedure takes as input a document X and a certificate $c = (x, t, n, z)$. As the first step it is verified that $x = h_c(X)$. The verification then proceeds by defining $y_1 := x$ and then inductively computing a sequence $y_2, \ldots, y_{l+1}$ so that:

$$y_{i+1} := \begin{cases} h(z_i, y_i) & \text{if } n_i = 0 \\ h(y_i, z_i) & \text{if } n_i = 1 \end{cases} . \tag{1}$$

We denote this computation by $y_{l+1} = V(x, n, z)$. Finally, it is checked whether y_{l+1} coincides with the hash value r_t stored in $\mathcal{R}$.

Security condition for time-stamping schemes [4] can directly be transformed into the so-called *chain-resistance* condition for the server-side hash function h:

Definition 1. *A hash function $h : \{0,1\}^{2k} \to \{0,1\}^k$ is* chain resistant *if for every poly-sampleable distribution family $\mathcal{D}_k$ (on $\{0,1\}^k$) with $\omega(\log k)$ bits of min-entropy and for every poly-time $A = (A_1, A_2)$*

$$\Pr[(r, a) = A_1, x \leftarrow \mathcal{D}_k, (n, z) = A_2(x, a) : V(x, n, z) = r] = k^{-\omega(1)} . \tag{2}$$

The question whether such hash functions exist or can be constructed from collision-free hash functions remains an open problem. It was proved [4] that black-box reductions are unable to prove that *every collision-resistant function is chain-resistant*. This paper studies the possibility of going one step further—showing that *chain-resistant functions cannot be constructed from collision-free functions in a black-box way*.

4 Cryptographic Reductions and Oracle Separation

To rule out the possibility of black-box reductions we first have to define black-box reductions properly. Assume we have to construct a primitive $\mathcal{P}$ from a base primitive $\mathcal{Q}$. It is a common cryptographic practice first to give a construction G that uses an implementation f of $\mathcal{Q}$ and creates an implementation $h = G^f$ of $\mathcal{P}$ and then show that if h is insecure as $\mathcal{P}$, then f has to be insecure as $\mathcal{Q}$. This guarantees that if there exist secure implementations of $\mathcal{Q}$, there also exist secure implementations of $\mathcal{P}$, so the existence of $\mathcal{P}$ is reduced to the existence of $\mathcal{Q}$. This leads to the following formalization [8]:

Definition 2. *We say that there exists a black-box reduction of a primitive $\mathcal{P}$ to a primitive $\mathcal{Q}$ if there exist polynomial-time oracle machines G and S such that the next two statements hold:*

Correctness: *For any implementation $f \in \mathcal{F}_{\mathcal{Q}}$ we have that $G^f \in \mathcal{F}_{\mathcal{P}}$.*
Security: *For any implementation $f \in \mathcal{F}_{\mathcal{Q}}$ and any adversary A, if A $\mathcal{P}$-breaks G^f, then $S^{A,f}$ $\mathcal{Q}$-breaks f.*

Here, S is the construction of the adversary that has access to f and A. Most classical reduction results fit into this model.

Oracle Separation. The idea of proving that such a reduction cannot exist comes from complexity theory where the main tool for proving such theorems is the so-called oracle separation method. It exploits the fact that black-box reductions stay valid in all relative worlds, i.e., computational models in which the ordinary Turing machines have access to oracles. To see this, assume that there exists a black box reduction from $\mathcal{P}$ to $\mathcal{Q}$ that is not relativizing. Then there exists such an oracle $\mathcal{O}$ that $\mathcal{Q}$ exists relative to it but $\mathcal{P}$ does not. Let $f \in \mathcal{F}_{\mathcal{Q}}$ be an efficient and secure implementation of $\mathcal{Q}$ relative to $\mathcal{O}$. It then follows from the black-box reduction that there exists $G^f \in \mathcal{F}_{\mathcal{P}}$ for which there is an adversary $A^{\mathcal{O},f}$ that breaks it. Then $S^{A,f,\mathcal{O}}$ is an adversary for f that breaks it, which is a contradiction since we assumed f to be secure relative to $\mathcal{O}$. So, in order to rule out a black-box reduction from $\mathcal{P}$ to $\mathcal{Q}$ it is sufficient to find an oracle that breaks all instances of $\mathcal{P}$ while leaving at least one instance of $\mathcal{Q}$ secure. See [13] for more details about reduction types and separations.

Such an argument was first used by Impagliazzo and Rudich [9] to show that there exist no black-box reductions of key establishment protocols to one-way functions, and was later used many times to rule out other black-box reductions. For example, Simon [14] showed that collision-free hash functions cannot be reduced to one-way permutations in a black-box way. In our case, we need an oracle that breaks chain resistance but leaves at least one function collision-free.

Hsiao and Reyzin [8] gave a rather powerful separation theorem. We extend it even further to show the full power of the approach and demonstrate exactly what would be required to rule out a black-box reduction.

Theorem 1. *If for all polynomial-time oracle machine pairs $R = (G, S)$ there exist A_R and f_R such that*

(a) f_R *implements* $\mathcal{Q}$
(b) There is a polynomial-time oracle machine D^{A_R,f_R} *that breaks* G^{f_R}.
(c) There is no polynomial-time oracle machine B *such that* B^{A_R,f_R} *breaks* f_R,

then there exist no black-box reductions from $\mathcal{P}$ *to* $\mathcal{Q}$.

Proof. Assume to the contrary that a black-box reduction exists. Then there exists a pair of oracle machines $R_0 = (G_0, S_0)$ that realizes the reduction. Then G_0^g implements $\mathcal{P}$ whenever g implements $\mathcal{Q}$ and by the assumptions there exist $f = f_{R_0}$ and $A = A_{R_0}$. Then by (a) f implements $\mathcal{Q}$ and G_0^f implements $\mathcal{P}$. By (b) there exists a polynomial-time oracle machine $D^{A,f}$ that breaks G_0^f. Then by the assumption that we have a fully black-box reduction, $S_0^{D^{A,f},f} = B^{A,f}$ breaks f. However, (c) claims no such B can exist, thus giving us a contradiction. □

As is evident from the theorem, we can actually vary the oracle we construct based on the machines that realize the construction and that the oracle only has to be helpful for one specific function. Also, the G part of the reduction has only a limited access to the oracle, only being able to use the original implementation of primitive $\mathcal{Q}$. This theorem along with these remarks is the main motivation behind the direction of our further analysis.

5 Hash Tree Oracle

There is one natural candidate for an oracle that breaks a function H in terms of chain-resistance—the one that constructs a full tree from all 2^n possible n-bit requests (Fig. 1, right), returns the root value, and is ready to output any certificate on demand. We can formalize this oracle as a triple $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3)$ such that $\mathcal{O}_1$ returns the root value of a tree constructed for the hash function H and $\mathcal{O}_2(x)$ gives a certificate (n, z) for x by taking the path from that tree starting from the input x. It is clear that such an oracle will break H in the chain-resistance sense with probability 1. We also add a third part $\mathcal{O}_3$ to the oracle which would implement a truly random function h from some well-chosen hash function family and assume H is given access to it. By Theorem 1 all we would need in this case is to show that the hash function supplied by $\mathcal{O}_3$ is hard to break even if $\mathcal{O}_1$ and $\mathcal{O}_2$ are available. Note that h does not have to be poly-time.

The approach may seem very promising. However, it turns out that if the oracle indeed gives out the root of a full tree (a tree with all the possible inputs x) from $\mathcal{O}_1$, such an oracle can always be exploited to find a collision. We state the result as a theorem:

Theorem 2. *Let* $h : \{0,1\}^n \to \{0,1\}^m$ *be a hash function* ($m < n$). *Then there exists a poly-time oracle machine* $H^h : \{0,1\}^{4n} \to \{0,1\}^{2n}$ *such that* $\mathcal{O}_1(H^h)$ *returns a collision for* h.

Proof. Define H^h so that on input $x_1\|x_2\|x_3\|x_4$ (where $x_i \in \{0,1\}^n$) the machine H^h returns $x_1\|x_2$ if (x_1, x_2) is a collision for h, otherwise, H^h returns $x_3\|x_4$. It is clear that if a collision is presented to H^h as a left or right input, it will output a collision.

Since $\mathcal{O}_1$ outputs the root value of the full tree, every possible $2n$-bit string (including collisions) is given as input to H^h at some leaf of the hash tree and hence at least one collision will propagate to the root of the tree. Since $m < n$, there certainly exists a collision. □

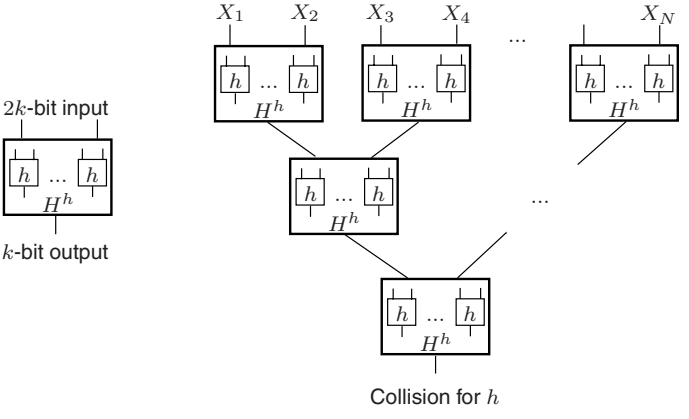

Fig. 2. Hash adversary and how it finds collisions for $h : \{0,1\}^{2k} \to \{0,1\}^k$ by using $N = 2^k/p(k)$ different k-bit inputs $X_1 \ldots X_N$

Theorem 2 shows that the full tree oracle can always be abused to find collisions for any hash function. Note, however, that we do not need to break the chain-resistance property with probability one. This means that we are allowed to fail to produce a certificate for some inputs x for $\mathcal{O}_2$ as long as we can produce a certificate with a non-negligible probability. This means that the tree does not need to be full, but it still needs to be quite large—a polynomial fraction $\frac{1}{p(k)}$ of all possible 2^k inputs. This motivates the following definition:

Definition 3 (Hash-Tree Oracle). *By a hash-tree oracle we mean any oracle* $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3)$ *such that*

- $\mathcal{O}_1$ *returns the root value of a hash-tree of size* $2^k/k^{O(1)}$ *constructed for the hash function* $H\colon \{0,1\}^{2k} \to \{0,1\}^k$
- $\mathcal{O}_2(x)$ *gives a certificate* (n, z) *for* x *by taking the path from that tree starting from the input* x*; and*
- $\mathcal{O}_3$ *gives access to a truly random function* f *chosen from a good collision-resistant family of (possibly non poly-time) hash functions.*

As Theorem 1 is our general guide, the hash-tree oracle is defined in terms of a fixed H. Normally, this H would be constructed by applying the black-box reduction to $h = \mathcal{O}_3$. We thus expect H itself to work in polynomial time where we assume that an $\mathcal{O}_3$ query takes constant time.

As we showed, information leak from $\mathcal{O}_1$ is clearly a problem in some cases. Although we do not have to construct the whole tree, it should still be of exponential size.

Hence, if H is cleverly constructed, then $\mathcal{O}_1$ outputs the result of an exponential computation. We thus turn our main attention to poly-time functions H that have access to h and are constructed for abusing $\mathcal{O}$ to find collisions to h. We call such functions *hash adversaries* (Fig. 2).

6 Disperser Adversary

The number of collisions for h can be much less than a polynomial fraction of all input pairs. For example, if h is a near-regular function, then the collisions form an exponentially small fraction of all input pairs. Thus, the hash-tree oracle can be constructed so that no collisions occur in the leaves of the tree. In order to improve the construction used in Theorem. 2 it is natural to consider the following generalization. Suppose we check more than one pair per each input but still pass on our findings so that if a collision is ever found, it will propagate to the root. It is suitable to explain this idea in terms of graph theory.

Suppose we are trying to construct $H\colon \{0,1\}^{2k} \to \{0,1\}^k$ that finds collisions for $h\colon \{0,1\}^n \to \{0,1\}^m$. Since we are trying to mimic the previous construction, assume that for each $x \in \{0,1\}^k$ the function has a set of pairs in h it will check for collisions, and if it finds one it will pass that same value x down. We can then construct a bipartite graph. Let P_n be the set of unordered pairs from $\{0,1\}^n$ (so $|P_n| = 2^{n-1}(2^n - 1)$) and let $G = (\{0,1\}^k, P_n, E)$ be the bipartite graph such that $(x,(y_1,y_2)) \in E$ iff H checks the pair (y_1,y_2) when given x as input. To find collisions successfully, we would like that every subset of $\{0,1\}^k$ with at least $\frac{2^k}{p(n)}$ elements has nearly all the elements of $\{0,1\}^{2n}$ as its neighbors—so that at least one collision would be among the neighbors. This means that no matter what inputs are given to the tree, assuming that there are at least $\frac{2^k}{p(n)}$ of them, a collision will certainly be found. There is one additional constraint—we can check only a polynomially bounded number of pairs because H is expected to be poly-time. It turns out that such graphs have been considered before in other applications.

Definition 4. *We call a bipartite graph $D = (V_1, V_2, E)$ a (K,ϵ)-disperser if the neighbor set $N(U)$ of every $U \subset V_1$ with cardinality K has at least $(1-\epsilon)|V_2|$ elements.*

Dispersers are mainly used as theoretical tools for randomized complexity classes or for extracting randomness from weak sources. They are generally considered alongside extractors, which have a similar but stronger requirements. Both types of graphs are often used in complexity theory and cryptography and there are numerous good surveys about their properties, constructions and bounds [12]. In our case, we are looking for a $K = \frac{2^k}{p(n)}$ disperser graph with as small ϵ as possible. It turns out that there are well-known lower bounds on all the parameters of disperser graphs.

Let D be the average degree of a vertex in V_1. Note first that in our case to cover enough of V_2, we need $KD \geq (1-\epsilon)|V_2|$ since otherwise there are not enough neighbors. Since most of the time some neighbors overlap, this would be an idealistic scenario. As it turns out, there are much stricter bounds for the parameters. In fact, Radhakrishnan and Ta-Shma give the following theoretical bounds in [11]:

Theorem 3. *Suppose that $G = (V_1, V_2, E)$ is a (K, ϵ)-disperser with $N = |V_1|$ and $M = |V_2|$. Let D be the average degree of a vertex in V_1.*

(a) Assume that $K < N$ and $D < \frac{(1-\epsilon)M}{2}$ (so G is not trivial). If $\frac{1}{M}\epsilon < \frac{1}{2}$, then $D \geq \frac{1}{\epsilon} \log \frac{N}{K-1}$.
(b) Assume that $K \leq \frac{N}{2}$ and $D \leq \frac{M}{4}$. Then $\frac{DK}{M} \geq c \log \frac{1}{\epsilon}$ for some c.

We try to apply these bounds to see how far the previously presented adversary idea could bring us. Let $q(n)$ be the polynomial by which D is bounded. From (a) we then find that $q(n) \geq c\frac{1}{\epsilon} \log(n)$ for some constant c and thus $\epsilon \geq \frac{c \log(n)}{q(n)} \geq \frac{1}{q'(n)}$ for some polynomial $q'(n)$. This means we can guarantee that all but $\frac{1}{q'(n)}$ of the possible pairs are covered. Simple combinatorics states that a hash function $h\colon \{0,1\}^n \to \{0,1\}^m$ has at least $2^{n-1}(2^m - 1)$ collisions which form less than a polynomial fraction of all the input pairs for h. This means that an adversary based on the disperser construction fails to find a collision. In the next section, we prove a more general result that all such collision pair-checking approaches will fail to abuse $\mathcal{O}$ for finding collisions.

7 Infeasibility of the Pair Checking Approach

If we could pick exactly what pairs we checked, then we could guarantee we will find a collision with 2^{n+m+1} queries. This is easy to prove by noting that collision pairs form a graph with 2^m cliques at least one of which has to contain at least 2^{n-m} vertices. Therefore, if we take the Turan graph – that is, a complete $(2^{n-m} - 1)$-partite graph with 2^n vertices divided as evenly as possible among the subsets – it has no cliques of size 2^{n-m} . Then, if we check all the pairs corresponding to edges in its complement graph, finding one collision is guaranteed.

Regardless of potentially exponential amount of computations that the tree oracle is able to perform, the set of pairs that we want to check is still limited. We will prove that any hash adversary that just checks input pairs for collisions will be unsuccessful if we use a well-constructed oracle.

First, we define such an adversary. We note that if it only checks whether certain pairs of inputs form a collision, then we can replace the oracle part $h = \mathcal{O}_3$ with a simple *collision-check oracle* c_h such that $c_h(x, y) = 1$ iff x and y form a collision. In this case, to construct $\mathcal{O}_1$ and $\mathcal{O}_2$, all we have to ensure is that the tree is formed in a way that c_h can always answer 0. In case we have few enough queries and answer all of them negatively, then it follows that the collision-resistance of $h = \mathcal{O}_3$ is preserved. It turns out that if we only use the collision-check oracle, this problem is (nearly) equivalent to trying to find a good enough disperser.

Lemma 1. *Let $\mathcal{F}$ be a family of hash functions $h : \{0,1\}^n \to \{0,1\}^m$ such that for any hash adversary H and for all but an ϵ fraction of $h \in \mathcal{F}$ there exists a subset K of cardinality $\delta 2^{2k}$ such that for all $x \in K$ no call to c_h made by $H(x)$ covers a real collision for h. Then we can construct a tree with at least $\delta 2^{2k}$ inputs for any hash adversary H' such that no collisions are checked during its execution. The reverse also holds.*

Proof. Let $H: \{0,1\}^{2k} \to \{0,1\}^k$ be any hash adversary. We construct a full Merkle tree of H^{c_0} such that every possible input $x \in \{0,1\}^{2k}$ is presented to it exactly once. We use the dummy oracle c_0 that always returns 0. Based on that tree, we define $H'(x)$ to do all the calculations and calls to the oracle c done by H^{c_0} on the path from $H(x)$ down to the root. Since H makes at most a polynomial number of calls to the oracle and the path is of length k, H' is also a hash adversary. We can therefore extract a $\delta 2^{2k}$ cardinality subset K for which H' will not ask for any actual collision of h from c (for all but an ϵ fraction of $h \in \mathcal{F}$). We construct the hash tree for H as the union of all the paths in the Merkle tree that begin from vertices in K. Since no collisions are found on these paths due to the construction of the set K, these paths are the same as those in the original Merkle tree constructed with the dummy oracle c_0. This means that their union indeed forms a proper tree and since no path finds a collision, the tree built as their union also fails to find one.

The inverse is straightforward. Let $H : \{0,1\}^{2k} \to \{0,1\}^k$ be a hash adversary and for some $h : \{0,1\}^n \to \{0,1\}^m$ we can find a tree with at least $\delta 2^{2k}$ different inputs so that no collisions are found. If we just take the set of inputs given to it, we have the required subset of cardinality K for H. □

This lemma shows that instead of trying to construct a tree, we should concentrate all our efforts to just finding a subset of input pairs with the required size and properties, because once we can do so for any hash adversary, we can also construct a tree. This removes one dimension of complexity and allows us to use graph theory again. The proof was carried out for full inputs of length $2k$ but if we can construct a tree from a polynomial fraction of them, it also contains a polynomial fraction of different single inputs of length k.

Theorem 4. *Assume that $H : \{0,1\}^{2k} \to \{0,1\}^k$ is a hash adversary with an oracle c_h for h where $h : \{0,1\}^n \to \{0,1\}^m$ is chosen uniformly from a family of hash functions $\mathcal{F}$. We also assume that for every pair $(x,y) \in \{0,1\}^{2n}$, $x \neq y$ the probability that $c(x,y) = 1$ is $2^{-\omega(\log(n))}$. Then the probability of not being able to construct a tree that does not show any collisions is negligible.*

Proof. Let $N(x_1, x_2)$ be the number of different inputs of H on which it checks for the collision $c(x_1, x_2)$. It is clear that $\sum_{(x,y)\in\{0,1\}^{4n}} N(x,y) \leq p(n)2^{2k}$ since there are a total of 2^{2k} different possible inputs and for each input the number of calls to c_h is bounded by a polynomial $p(n)$. We calculate the expected value of the number of inputs N_p that will lead to a collision being detected in their branch:

$$\begin{aligned} E[N_p] &\leq \sum_{h\in\mathcal{F}} \Pr[h] \sum_{(x,y)\in P_n} N(x,y)[h(x) = h(y)] \\ &= \sum_{(x,y)\in P_n} N(x,y) \sum_{h\in\mathcal{F}} \Pr[h][h(x) = h(y)] = \\ &= \sum_{(x,y)\in P_n} N(x,y) 2^{-\omega(\log(n))} \leq p(n) 2^{2k-\omega(\log(n))} \quad . \end{aligned}$$

Using the Markov inequality we get that $\Pr[N_p \geq 2^{2k-1}] \leq p(n)2^{1-\omega(\log(n))}$. Hence, a subtree with at least half of all the different inputs fed to it that fails to find a collision

can be found for all but a $p(n)2^{1-\omega(\log(n))}$ fraction of h. Using the previous theorem now gives us the promised result with $\delta = \frac{1}{2}$. □

Roughly, this theorem says that given any fixed hash adversary, the oracle can avoid revealing the collisions directly. This does not, however, rule out that one can find collisions from such trees indirectly. We now extend the construction given so far to rule out this as well.

Theorem 5. *Under the assumptions of the previous theorem, the probability of finding a collision can be made negligibly small.*

Proof. Let 2^{-d} be the collision probability in the previous proof. For each $h \in \mathcal{F}$ we randomly choose a set $\mathcal{F}_h \subset \mathcal{F}$ of size $2^{0.5d}$ and try to avoid all of their collisions instead of just those for h itself. The probability of randomly choosing a collision for one of them from the set of all possible pairs is $2^{-d} \cdot 2^{0.5d} = 2^{-0.5d}$ and if $d = \omega(\log n)$, then so is $0.5d$. This means that the argumentation of the theorem will still go through and that we can still avoid the set of forbidden pairs for all but a super-polynomial fraction of $h \in \mathcal{F}$. This, however, also means that each such avoiding tree could be valid for any of the $2^{0.5d}$ possible hash functions in $\mathcal{F}$ instead of just one and considering the construction and size of $\mathcal{F}$, the average chance of guessing a collision correctly only knowing the set $\mathcal{F}_h$ that h belongs to it is still negligible. □

This means that an adversary that only uses the pair-checking approach can always be fooled. This only rules out a one rather simplistic approach for the adversary trying to abuse the oracle. In the next sections we study if our approach could be extended or improved in any way.

8 Other Possible Types of Hash-Adversaries

The result at the end of the previous section means that the approach we tried for constructing the adversary for collision-resistance needs to be altered. We have to abandon the strategy of just checking pairs and try to do something more clever. As it turns out, however, this approach can be used to rule out a few other seemingly promising approaches.

8.1 Input-Output Pair Check

There is a lexicographic total ordering of fixed-length bit-strings. One idea of how to find collisions is that we could try to find the maximum and minimum inputs that produce a given output. This can be done by asking by H^h questions of the form $h(x) =? y$, where $x \in \{0,1\}^n, y \in \{0,1\}^m$. If we formalize that in the form of a so-called *equality oracle* $e_h(x,y)$, the same argument as presented for the collision oracle c can be carried out nearly word for word to show that this approach has as little potential as our previous one, since as before, we can almost always find trees such that only 0 is answered by e. In this case, the infeasibility of collision-finding does not follow as easily though. We note that it would take $2^m - 1$ queries of this type that were answered negatively to

determine the value $h(x)$. Therefore, there is a possibility that the adversary finally obtains the value of $h(x)$ during the tree-computations. However, the argument given for c_h in Theorem 5 that looked at collisions for a large set of a randomly chosen $h' \in \mathcal{F}$ instead of a single h could again be used rather effectively to rule out the equality oracle giving enough useful information. As for the case of multiple calls to the oracle, the same vulnerability clearly exists as with the collision oracle.

8.2 Output Comparison

The approach of finding the minimum or maximum of some values gives us another idea of what to try to rule out. Suppose H only asks questions of the form $h(x) > h(y)$, where $<$ is the lexicographic order. In this case, we can also construct a binary *greater-than* oracle $g(x, y)$ to answer queries of this type. There is one very important difference between this oracle and the previous two (i.e., c and e). In c and e, the answers were clearly asymmetric in the amount of information they gave away, i.e., 0 was a "safe" answer and if we could avoid ever answering 1, the oracle could not be exploited. In this case, both answers 0 and 1 give out a nearly equal information, so always answering 0 is just as dangerous as always answering 1.

This means that in order to fool the hash adversary we have to re-check all previous g-calls. We note that Theorem 1 essentially relies on our ability to "fake" safe answers so we could get a static tree structure. As we remember from Theorem 2, if we honestly answer the queries in the *full tree*, then H is able to guarantee a collision in the root. We also note that the greater-than oracle can be used to emulate the collision oracle as $c(x, y) = 1$ happens exactly when $g(x, y) = 0$ and $g(y, x) = 0$. This means that to use the same technique, we have to find a way to fake the oracle answers consistently enough that a "correct" subtree (where all the answers were correct) could be extracted but yet falsely enough that the hash adversary will not have a collision in the root.

There is indeed a way to do so. We extend $h : \{0,1\}^n \to \{0,1\}^m$ to a new function $h' : \{0,1\}^n \to \{0,1\}^{m+n}$ such that the most significant bits of $h'(x)$ correspond to the output of $h(x)$, but the least significant ones are chosen randomly with a constraint that h' is injective. So, no output corresponds to two different inputs. We then answer $g(x, y)$ queries based on h' instead of h. Clearly, we have to clean the tree from all the answers that say $g(x, y)$ when $h(x) = h(y)$ but this is essentially the same as clearing the tree from all the queries where $c(x, y) = 1$.

Still, a small problem remains—the argument used to prove that the answers given by the oracle do not uniquely determine the collisions by some indirect means does not work as it did before. The original idea needs a slight modification. Instead of choosing a random hash function into $\mathcal{F}_h$, we choose another hash function in such a way that consistent answers can still be given rather easily for g. To be precise, we try to avoid answering queries such that $h(x)$ and $h(y)$ differ only in the last $0.5d$ bits where 2^{-d} is the collision probability. Essentially the same argumentation as before could then be used. Intuition behind this approach is that instead of h we simply consider a hash function $h'' : \{0,1\}^n \to \{0,1\}^{m-0.5d}$ for which we eliminate the collisions in the tree. Since we remove any oracle queries that distinguish between the collisions within a class of size $2^{0.5d}$ of output values, it is easy (but rather technical) to prove that even knowing all the information given out by the oracle the chance of finding a collision is negligible.

8.3 Using All Three Approaches

We note that we can also rule out all the constructions that use a combination of all three oracles c, e, and g, then the last step in Theorem 4 can be changed to use $\delta = \frac{4}{5}$. Then we can choose the sets separately for all three oracles, so that none of them gives enough information and then take their union which is of size at least a fraction $\delta' = \frac{1}{5}$. This can be done even after the tricks used in Theorem 5 and its analogs have been taken into account. Therefore, this paper rules out any hash adversary constructions that use only on these three oracles and do not use any additional types of information. This seems to rule out most of the simple ways of exploiting the oracle, so if the tree oracle does not work, the adversary that can use it to break collision-resistance for $\mathcal{O}_3$ has to be fairly clever.

8.4 Input-Output Comparison

One obvious generalization of the input-output check oracle e would be the so-called input-output comparison oracle that answers questions of type $h(x) > y?$. This oracle has one strength over the others—while it would take $2^m - 1$ queries to know the actual value of $h(x)$ using the e-oracle and even more for the other two, it only takes roughly m queries to find $h(x)$ using the input-output comparison oracle, because we can do *binary search*. This means that we could easily interchange between a binary oracle of this type and a full oracle for h since we can compute one from the other in polynomial time. Therefore, if we could learn to somehow fake this oracle in the same sense as we have faked the three previous ones, it would give us a full proof of the separation we have been seeking. It should then be no surprise that this oracle seems to be much harder to fake and we do not know any efficient ways of doing so.

9 Discussion

The results presented so far leave the possibility of oracle separation between unbounded hash-tree time-stamping and collision-free hash functions in a rather ambiguous state—the proposed hash-tree oracle (Def. 3) seems to be hard to exploit, but at the same time, it seems nearly as impossible to rule out a hash-adversary that nonetheless does it. There is still one more approach that may lead to some further results. We note that for any possible construction presented to the oracle, the probability of it giving any information about the collisions of h supplied by the oracle taken over all possible inputs nearly always has to be negligible—if it is not, we could do without any oracles by simply choosing a random input, using the function on it and trying to deduce a collision based on that. We essentially proved this fact for collision checking adversaries in Theorem 4 and then extended it to other oracle types. Therefore, one might try considering an oracle where the root value supplied by $\mathcal{O}_1$ is generated randomly. In this case, however, the certificates given out by $\mathcal{O}_2$ might help in breaking h a bit more than they do on average and the advantage gained in such a way seems surprisingly hard to bound. The initial results in this direction seem somewhat promising but nothing concrete has come out of that approach yet. At the moment it seems that the problem needs more complicated mathematical machinery, perhaps some information theoretical bounds or

even Kolmogorov Complexity may come into play. However, there may still be better choices for the separation oracle that could lead to easier proofs of separation.

References

1. Bayer, D., Haber, S., Stornetta, W.-S.: Improving the efficiency and reliability of digital time-stamping. In: Sequences II: Methods in Communication, Security, and Computer Science, pp. 329–334. Springer, Heidelberg (1993)
2. Buldas, A., Laud, P., Lipmaa, H., Villemson, J.: Time-Stamping with Binary Linking Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 486–501. Springer, Heidelberg (1998)
3. Buldas, A., Laur, S.: Do broken hash functions affect the security of time-stamping schemes? In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 50–65. Springer, Heidelberg (2006)
4. Buldas, A., Saarepera, M.: On Provably Secure Time-Stamping Schemes. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 500–514. Springer, Heidelberg (2004)
5. Buldas, A., Jürgenson, A.: Does secure time-stamping imply collision-free hash functions? In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 138–150. Springer, Heidelberg (2007)
6. Haber, S., Stornetta, W.-S.: How to time-stamp a digital document. Journal of Cryptology 3(2), 99–111 (1991)
7. Haber, S., Stornetta, W.-S.: Secure Names for Bit-Strings. In: ACM Conference on Computer and Communications Security, pp. 28–35 (1997)
8. Hsiao, C.-Y., Reyzin, L.: Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (2004)
9. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proceedings of 21st Annual ACM Symposium on the Theory of Computing, pp. 44–61 (1989)
10. Merkle, R.C.: Protocols for public-key cryptosystems. In: Proceedings of the 1980 IEEE Symposium on Security and Privacy, pp. 122–134 (1980)
11. Radhakrishnan, J., Ta-Shma, A.: Bounds for dispersers, extractors, and depth-two superconcentrators. SIAM Journal on Discrete Mathematics 13(1), 2–24 (2000)
12. Shaltiel, R.: Recent Developments in Explicit Constructions of Extractors. In: Bulletin of the EATCS, vol. 77, pp. 67–95 (2002)
13. Reingold, O., Trevisan, L., Vadhan, S.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
14. Simon, D.: Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)

Relationship of Three Cryptographic Channels in the UC Framework

Waka Nagao[1], Yoshifumi Manabe[1,2], and Tatsuaki Okamoto[1,2]

[1] Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Kyoto, 606-8501 Japan
w-nagao@ai.soc.i.kyoto-u.ac.jp
[2] NTT Labs, Nippon Telegraph and Telephone Corporation
3-9-11 Midori-cho, Musashino, Tokyo, 180-8585 Japan
{manabe.yoshifumi,okamoto.tatsuaki}@lab.ntt.co.jp

Abstract. The relationship of three cryptographic channels, secure channels (SC), anonymous channels (AC) and direction-indeterminable channels (DIC), was investigated by Okamoto. He showed that the three cryptographic channels are reducible to each other, but did not consider communication schedules clearly as well as composable security. This paper refines the relationship of the three channels in the light of communication schedules and composable security. We model parties by the task-probabilistic input/output automata (PIOA) to treat communication schedules, and adopt the universally composable (UC) framework by Canetti to treat composable security. We show that a class of anonymous channels, two-anonymous channels (2AC), and DIC are reducible to each other under any schedule and that DIC and SC are reducible to each other under some types of schedules, in the UC framework with the PIOA model.

Keywords: Secure Channel (SC), Two-Anonymous Channel (2AC), Direction-Indeterminable Channel (DIC), Universal Composability (UC), Probabilistic Input/Output Automaton (PIOA).

1 Introduction

One of the most important results in cryptography is the relationship among *computational* cryptographic assumptions. For example, several of the most important cryptographic primitives such as pseudo-random generators, secure bit-commitment, and secure signature schemes have been proven to exist if and only if one-way functions exist [8,9,10,11,13].

Apart from the *computational* assumptions made in the above-mentioned works, some *physical* or *unconditionally secure* assumptions and primitives, which rely on no computational condition/assumption, are known to be essential in unconditionally secure (or information theoretically) cryptography. For example, unconditionally secure multi-party protocols can be constructed assuming *secure channels* [1,7].

The relationship of such *physical* (unconditionally secure) assumptions for channels has been studied by [12]. The paper shows that three physical assumptions about channels are equivalent (or reducible to each other). Here, the three physical assumptions are

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 268–282, 2008.

the existence of the anonymous channel(AC), direction-indeterminable channel(DIC), and secure channel(SC).

However, paper [12] did not consider communication schedules clearly as well as composable security, although the communication schedule like synchronous or asynchronous communication is critical in the reductions of the three channels, and composable security is crucial for channels since channels are always lower level components of systems and applications.

In this paper, we refine the relationship of the three channels in the light of communication schedules and composable security.

This paper adopts the universally composable (UC) framework by Canetti [2] to treat composable security, since UC is the most powerful and well-studied framework for composable security and is flexible enough to cover the security of physical (unconditionally secure) primitives like channels.

Although parties are usually modeled by interactive Turing machines (ITMs) in the standard settings in cryptography including the UC framework [2], this paper models parties by not ITMs but by task-probabilistic input/output automata (PIOA) [3,4,5,6] to treat communication schedules. This is because: ITMs cannot treat flexible communication schedules like various types of asynchronous and nondeterministic schedules, but task PIOA is one of the most powerful models to treat a variety of communication schedules. The master schedule in task PIOA can control timing of activation among party with flexible schedule.

This paper shows that a class of anonymous channels, two-anonymous channels (2AC), and DIC are reducible to each other under any schedule in the UC framework with the PIOA model. We also show that DIC and SC are reducible to each other under some types of schedules in the UC framework with the PIOA model.

2 Preliminaries

This section introduces the basic notion of (task) Probabilistic Input/Output Automata (PIOA) and security notion of Universally Composability(UC). (See papers [6] and [2] for PIOA and UC, respectively, if you need more details.)

2.1 (Task) Probabilistic I/O Automata

We recall the basic definitions of PIOA and task-PIOA from [3,4,6].

Definition 1 [Probabilistic I/O Automaton (PIOA)]. *Let Q, $\overline{q}$, I, O, H and D be, respectively, a countable set of states, a start state (satisfying $\overline{q} \in Q$), a countable set of input actions, a countable set of output actions, a countable set of internal actions and a transition relation satisfying $D \subseteq Q \times (I \cup O \cup H) \times Disc(Q)$, where $Disc(Q)$ is the set of discrete probability measures on Q. Let PIOA $\mathcal{P}$ be the tuple of $(Q, \overline{q}, I, O, H, D)$.*

Definition 2 [Task Probabilistic I/O Automaton (Task PIOA)]. *Let $T = (\mathcal{P}, \mathcal{R})$ be a task-PIOA , where $\mathcal{P} = (Q, \overline{q}, I, O, H, D)$ is a PIOA (satisfying the transition determinism and input enabling properties), and $\mathcal{R}$ is an equivalence relation on the locally-controlled actions $L = O \cup H$.*

Execution Fragment and Trace. Let q_i and a_i for $i \in \{0, 1, 2, \cdots\}$ be states and actions, respectively. We consider that an execution fragment of task-PIOA T is the following infinite or finite sequence $\alpha = q_0 a_1 q_1 a_2 \ldots$. If the sequence α is a finite sequence, the last state of α is denoted by $\mathrm{lst}(\alpha)$. If α is a finite sequence with $\mathrm{lst}(\alpha) = q_{i+1}$, for each (q_i, a_{i+1}, q_{i+1}) there exists a transition $(q_i, a_{i+1}, \mu) \in D$ with $q_{i+1} \in \mathrm{supp}(\mu)$, where $\mathrm{supp}(\mu)$ is a support of μ. If there exists an execution fragment α of an automaton P, we denote by $trace(\alpha)$ the input and output (external actions) sequence obtained from α.

In this paper, we formally model parties $P_1, \cdots, P_n$ in a protocol by task-PIOA $T_1, \cdots, T_n$. Each party P_i has a local scheduler ρ_i for the task-PIOA T_i. There exists a master scheduler M for all parties, $P_1, \cdots, P_n$ in a protocol.

Definition 3 [Local Scheduler]. *Let T be a closed task-PIOA for a party P. A local scheduler, ρ, for T is defined to be a finite or infinite sequence of tasks, $t_1, t_2, \cdots$; i.e., $\rho = t_1, t_2, \cdots$. ρ specifies the executing order of tasks in T. (We often omit the explicit description of ρ in the specification of a task-PIOA if ρ is trivial from the specification.)*

Definition 4 [Master Scheduler]. *A master scheduler, M, is defined to be a finite or infinite sequence of party identifiers, $i_1, i_2, \cdots$; i.e., $M = i_1, i_2, \cdots$. M grobally specifies the executing order of tasks in a protocol of $(P_1, \cdots, P_n)$ with preserving the local schedulings of all parties.*

For example, let ρ_i of party i be $t_{i1}, t_{i2}, \cdots$ $(i = 1, 2, 3)$, and $M = 1, 2, 2, 2, 3, 1, 1, 3$. Then the grobal executing order of task is $t_{11}, t_{21}, t_{22}, t_{23}, t_{31}, t_{12}, t_{13}, t_{32}$.

The master schedule is not under the control of adversary although the local schedule is under the control of adversary. In other words, the adversary can not to intervene the master schedule, but he can encumber the local schedule.

2.2 Universal Composability

UC Security. Let *Env*, *Adv*, and *Sim* be an environment, an adversary, and a simulator, respectively. Let *Real* denote the output of environment *Env* when interacting with adversary *Adv* and parties *Init* and *Rec* running channel protocol π. Let *Ideal* denote the output of environment *Env* after interacting in the ideal world with simulator *Sim* and ideal functionality $\mathcal{F}$. We say that *Real* UC-realizes $\mathcal{F}$, if for any adversary $Adv \in PPT$ (probabilistic polynomial time) there exists a simulator $Sim \in PPT$ such that for any environment $Env \in PPT$, $Ideal_{\mathcal{F},Sim,Env} \approx Real_{\pi,Adv,Env}$, where $\approx$ denotes statistically indistinguishable and *PPT* denotes a class of polynomial-time bounded machines.

Hereafter, we use the bold style, Real and Ideal, to express systems of the PIOA notion to distinguish the real and ideal world of UC security. We then use the roman style also means the task-PIOA to divide the notions between task-PIOA and UC notion.

3 Three Cryptographic Channels and Definitions

3.1 Secure Channel (SC)

A secure channel is a channel such that the initiator (message sender) and the receiver can safely transmit messages to each other without the content being retrieved by a third

party or adversary. This secure channel consists of three sessions, establish session, data sending session, and expire session: 1. The establish session creates a session between the initiator and the receiver to start message sending. 2. The data sending session sends a message to the receiver safely. 3. The expire session terminates the the existing session and clears the secret key.

Definition 5. *The code for secure channel,* $\mathrm{F_{SC}}$*, is defined in Fig. 1. (*$\bar{X}$*, where* $(X \in \mathrm{Init}, \mathrm{Rec}\})$*, means that if* $X = \mathrm{Init}$ *then* $\bar{X} = \mathrm{Rec}$ *else if* $X = \mathrm{Rec}$ *then* $\bar{X} = \mathrm{Init}$*.)*

Code for Secure Channel Functionality, $\mathrm{F_{SC}}$, where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ and $\mathtt{sid}_{\mathrm{SC}} = (\mathrm{Init}, \mathrm{Rec}, \mathtt{sid}'_{\mathrm{SC}})$.

State: $estcond_X \in \{\bot, \top\}$, initially $\bot$, $active \in \{\bot, \top\}$, initially $\bot$, $okcond_{\mathrm{Adv}} \in \{\bot, \top\}$, initially $\bot$, $mes \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$, $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Transitions:

– Establish Session:

ESS1. $receive(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$ Precondition: $active = \bot$, Effect: $estcond_X := \top$, If $estcond_X = \top$ for all X then $active := \top$.

ESS2. $send(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$ Precondition:$active = \top$, Effect: none

– Data Sending Session:

DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_X$ Precondition:$active = \top$ and $mes = \bot$, Effect: $mes := m$

DSS2. $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}}$ Precondition: $active = \top$, $mes \neq \bot$, $okcond_{\mathrm{Adv}} = \bot$ and $m := mes$, Effect: none

DSS3. $receive(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, ok)_{\mathrm{Adv}}$ Precondition:$active = \top$ and $okcond_{\mathrm{Adv}} = \bot$, Effect: $okcond_{\mathrm{Adv}} := \top$

DSS4. $send(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, mes)_{\overline{X}}$ Precondition: $active, mes = \top$ and $okcond_{\mathrm{Adv}} \neq \bot$, Effect: mes and $okcond_{\mathrm{Adv}} := \bot$

– Expire Session:

EXS1. $receive(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$ Precondition: $active \neq \bot$ and $mes = \bot$, Effect: $ntask :=$ EXS2

EXS2. $send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$ Precondition: $ntask =$ EXS2, Effect: $active, ntask$ and $estcond_X := \bot$ for all X

Fig. 1. Code for Secure Channel Functionality, $\mathrm{F_{SC}}$

3.2 Two-Anonymous Channel (2AC)

An anonymous channel is one of the three cryptographic channels and is able to send some messages to the receiver from unknown senders (”anonymously”). The adversary can know the identity of the receiver and the message content, but cannot know who sent the message to the receiver. When two senders and a receiver anonymously communicate by a channel, we say the channel is a two-anonymous channel. That is, one of the two senders sends a message to the receiver. Note that two-anonymous channel can also be used when the receiver and one of the senders is the same process.

Definition 6. *The code for two-anonymous channel,* $\mathrm{F_{2AC}}$*, is defined in Fig. 2.*

3.3 Direction-Indeterminable Channel (DIC)

A direction-indeterminable channel is one of the three cryptographic channels and is able to send some messages to the receiver direction-indeterminably. The adversary can

Code for Two Anonymous Channel Functionality, $\mathrm{F}_{\mathrm{2AC}}$, where for $X \in \{\mathrm{Init}_i, \mathrm{Rec}\}$ and $\mathtt{sid}_{\mathrm{2AC}} = (\{\mathrm{Init}_1, \mathrm{Init}_2\}, \mathrm{Rec}, \mathtt{sid}'_{\mathrm{2AC}})$.

State: $estcond_X \in \{\bot, \top\}$, initially $\bot$, $okcond_{\mathrm{Adv}} \in \{\bot, \top\}$, initially $\bot$, $mes \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$, $active \in \{\bot, \top\}$, initially $\bot$, $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Transitions:

– Establish Session:

ESS1. $receive(\mathtt{Establish}_{\mathrm{2AC}}, \mathtt{sid}_{\mathrm{2AC}})_X$ Precondition: $active = \bot$ and $ntask = \bot$, Effect: $estcond_X := \top$ If $estcond_X$ for all X then $active := \top$.

ESS2. $send(\mathtt{SID}, \mathtt{sid}_{\mathrm{2AC}})_{\mathrm{Adv}}$ Precondition: $active = \top$ and $ntask$ = ESS2, Effect: $ntask := \bot$

– Data Sending Session:

DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathrm{2AC}}, m)_{\mathrm{Init}_i}(i \in \{1,2\})$ Precondition: $active = \top$, $mes = \bot$ and $ntask = \bot$, Effect: $mes := m$ and $ntask :=$ DSS2

DSS2. $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{2AC}}, mes)_{\mathrm{Adv}}$ Precondition: $okcond_{\mathrm{Adv}} = \bot$, $mes := m$ and $ntask$ = DSS2, Effect: $ntask :=$ DSS3

DSS3. $receive(\mathtt{Response}, \mathtt{sid}_{\mathrm{2AC}}, ok)_{\mathrm{Adv}}$ Precondition: $ntask$ = DSS3, Effect: $okcond_{\mathrm{Adv}} := \top$ and $ntask :=$ DSS4

DSS4. $send(\mathtt{Receive}, \mathtt{sid}_{\mathrm{2AC}}, mes)_{\mathrm{Rec}}$ Precondition: $ntask$ = DSS4, Effect: $okcond_{\mathrm{Adv}}$, mes and $ntask := \bot$

– Expire Session:

EXS1. $receive(\mathtt{Expire}_{\mathrm{2AC}}, \mathtt{sid}_{\mathrm{2AC}})_X$ Precondition: $active = \top$, mes and $ntask = \bot$, Effect: $ntask :=$ EXS2

EXS2. $send(\mathtt{Expire}_{\mathrm{2AC}}, \mathtt{sid}_{\mathrm{2AC}})_{\mathrm{Adv}}$ Precondition: $ntask$ = EXS2, Effect: $active, estcond_X$ and $ntask := \bot$ for all X

Fig. 2. Code for Two Anonymous Channel Functionality, $\mathrm{F}_{\mathrm{2AC}}$

know the identities of both parties and the transmitted message, but cannot know who the sender was. That is, the direction of message transmission is indeterminable.

Definition 7. *The code for direction-indeterminable channel,* $\mathrm{F}_{\mathrm{DIC}}$*, is defined in Fig.* 3.

3.4 Security Definitions

We define the security notion on PIOA considering the synchronous and asynchronous schedule as follows:

Definition 8 [Perfect Implementation]. *Let* **Env**, **Real** *and* **Ideal** *be an environment task-PIOA, a real protocol task-PIOA system and an ideal functionality task-PIOA system, respectively. Let* sch *be the some (synchronous or asynchronous) schedule. We say that* **Real** *perfectly implements* **Ideal** *under some (synchronous or asynchronous) schedule (or* $\mathbf{Real} \leq_0^{\mathrm{sch.}} \mathbf{Ideal}$*), if trace*(**Real**||**Env**) = *trace*(**Ideal**||**Env**) *for every environment* **Env** *under synchronous or asynchronous schedule.*

Definition 9 [Perfect Hybrid Implementation]. *Let* Hybrid *be a real protocol task-PIOA system with hybrid model. We say that* Hybrid *perfectly hybrid implements* **Ideal** *under some (synchronous or asynchronous) schedule (or* $\mathrm{Hyb.} \leq_0^{\mathrm{sch.}} \mathbf{Ideal}$*), if trace* (**Hybrid**||**Env**) = *trace*(**Ideal**||**Env**) *for every environment* **Env** *under some (synchronous or asynchronous) schedule.*

Code for Direction-Indeterminable Channel Functionality, F_{DIC},
where $X \in \{\text{Init}, \text{Rec}\}$ and $\mathtt{sid}_{DIC} = (\{\text{Init}, \text{Rec}\}, \mathtt{sid}'_{DIC})$.

State: $estcond_X \in \{\bot, \top\}$, initially $\bot$, $mes \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$, $okcond_{Adv} \in \{\bot, \top\}$, initially $\bot$, $active \in \{\bot, \top\}$, initially $\bot$, $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Transitions:

– Establish Session:

ESS1. $receive(\mathtt{Establish}_{DIC}, \mathtt{sid}_{DIC})_X$ Precondition: $active = \bot$ and $ntask = \bot$, Effect: $estcond_X := \top$ If $estcond_X = \top$ for all X then $active := \top$ and $ntask :=$ ESS2.

ESS2. $send(\mathtt{SID}, \mathtt{sid}_{DIC})_{Adv}$ Precondition: $active = \top$ and $ntask =$ ESS2, Effect: $ntask := \bot$

– Data Sending Session:

DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{DIC}, m)_X$ Precondition: $active = \top$, mes and $ntask = \bot$, Effect: $mes := m$ and $ntask :=$ DSS2

DSS2. $send(\mathtt{Send}, \mathtt{sid}_{DIC}, m)_{Adv}$ Precondition: $okcond_{Adv} = \bot$, $mes := m$ and $ntask =$ DSS2, Effect: $ntask :=$ DSS3

DSS3. $receive(\mathtt{Response}, \mathtt{sid}_{DIC}, ok)_{Adv}$ Precondition: $ntask =$ DSS3, Effect: $okcond_{Adv} := \top$ and $ntask :=$ DSS4

DSS4. $send(\mathtt{Send}, \mathtt{sid}_{DIC}, mes)_{\overline{X}}$ Precondition: $ntask =$ DSS4, Effect: $okcond_{Adv}$, mes and $ntask := \bot$

– Expire Session:

EXS1. $receive(\mathtt{Expire}_{DIC}, \mathtt{sid}_{DIC})_X$ Precondition: $active = \top$, $mes = \bot$ and $ntask = \bot$, Effect: $ntask :=$ EXS2

EXS2. $send(\mathtt{Expire}_{DIC}, \mathtt{sid}_{DIC})_{Adv}$ Precondition: $ntask =$ EXS2, Effect: $active$, $estcond_X$ and $ntask := \bot$ for all X

Fig. 3. Code for Direction-Indeterminable Channel Functionality, F_{DIC}

4 Equivalence between DIC and 2AC

In this section, we prove that the direction indeterminable channel (DIC) is equivalent to the two-anonymous channel (2AC) under any schedule. That is, the task-PIOA of DIC perfectly implements the task-PIOA of 2AC under any schedule. To prove this, we show two reductions of DIC to 2AC and 2AC to DIC. Here, we consider the one bit message exchange, that is, $|m| = 1$. Informally, the reduction of DIC to 2AC is proven as follows: The direction-indeterminable property is made by using two 2AC functionalities, F^I_{2AC} and F^R_{2AC}. Here, the two senders of F^I_{2AC} are Init and Rec, and the receiver of F^I_{2AC} is Init. Then, the two senders of F^R_{2AC} are Init and Rec, and the receiver of F^R_{2AC} is Rec. When Init sends a message to the receiver Rec, Init sends the message by F^I_{2AC} and F^R_{2AC}. That is, F^I_{2AC} forwards the message to Init and F^R_{2AC} forwards the message to Rec. The adversary cannot detect the message direction because Init and Rec receive the same message m transfered by the two 2ACs. The other reduction, 2AC to DIC, is proven as follows: First, the message sending party (Init_1 or Init_2) sends a message m to the other party by DIC. Init_1 and Init_2 then send m to the receiver Rec directly. The adversary cannot detect which is the sender because the message direction among senders Init_1 and Init_2 is indeterminable.

4.1 Reduction of DIC to 2AC

Let π_{DIC} be a protocol of direction-indeterminable channel. We assume that $M_{\pi_{DIC}}$, the master schedule of π_{DIC}, is any schedule. Let Init_{DIC} and Rec_{DIC} be the initiator's code

and receiver's code for a real system, respectively, see Fig.4, Fig.5. Let $\overline{\text{Init}_{\text{DIC}}}$ and $\overline{\text{Rec}_{\text{DIC}}}$ be the initiator's code and receiver's code for an ideal system, respectively, see Fig.7 and Fig.8. Finally, let Adv_{DIC} and Sim_{DIC} be the adversary's code and the simulator's code in Fig.6 and Fig.9, respectively. Let $\mathbf{Real}_{\mathbf{DIC}}$ and $\mathbf{Ideal}_{\mathbf{DIC}}$ be a direction-indeterminable channel protocol system and a direction-indeterminable channel functionality system, respectively, defined as follows:

$$\mathbf{Real}_{\mathbf{DIC}} := \text{Init}_{\text{DIC}} || \text{Rec}_{\text{DIC}} || \text{Adv}_{\text{DIC}} || \text{F}^{\text{I}}_{\text{2AC}} || \text{F}^{\text{R}}_{\text{2AC}},$$
$$\mathbf{Ideal}_{\mathbf{DIC}} := \overline{\text{Init}_{\text{DIC}}} || \overline{\text{Rec}_{\text{DIC}}} || \text{Sim}_{\text{DIC}} || \text{F}_{\text{DIC}}.$$

Code for Initiator of Direction-Indeterminable Channel, Init_{DIC}, where $X \in \{\text{I}, \text{R}\}$,
where $\texttt{sid}_{\text{DIC}} = (\{\text{Init}, \text{Rec}\}, \texttt{sid}'_{\text{DIC}})$, $\texttt{sid}^{\text{I}}_{\text{2AC}} = (\{\text{Init}, \text{Rec}\}, \text{Init}, \texttt{sid}'^{\text{I}}_{\text{2AC}})$ and $\texttt{sid}^{\text{R}}_{\text{2AC}} = (\{\text{Init}, \text{Rec}\}, \text{Rec}, \texttt{sid}'^{\text{R}}_{\text{2AC}})$.

State: $smes, rmes \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$, $ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$, $active \in \{\perp, \top\}$, initially $\perp$
Transitions:
– Establish Session:
ESS1. $in(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{Init}}$ Precondition: $active, ntask = \perp$, Effect: $ntask := \text{ESS2}$
ESS2. $send(\texttt{Establish}_{\text{2AC}}, \texttt{sid}^{\text{X}}_{\text{2AC}})_{\text{F}^{\text{X}}_{\text{2AC}}}$ Precondition: $ntask = \text{ESS2}$ (Note that each task for $X \in \{I, R\}$ activates arbitrarily order. Hereafter, we stand by this manner.), Effect: $active := \top$ and $ntask := \perp$
– Data Sending Session:
DSS1. $in(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{Init}}$ Precondition: $active = \top$, $smes$ and $ntask = \perp$, Effect: $smes := m$ and $ntask := \text{DSS2}$
DSS2. $send(\texttt{Send}, \texttt{sid}^{\text{X}}_{\text{2AC}}, m)_{\text{F}^{\text{X}}_{\text{2AC}}}$ Precondition: $m := smes$ and $ntask = \text{DSS2}$, Effect: $ntask := \perp$
DSS3. $receive(\texttt{Receive}, \texttt{sid}^{\text{I}}_{\text{2AC}}, m)_{\text{F}^{\text{I}}_{\text{2AC}}}$ Precondition: $active = \top$, $rmes$ and $ntask = \perp$, Effect: If $smes = \perp$, then $rmes := m$ and $ntask := \text{DSS4}$. Else $smes := \perp$ and $ntask := \perp$
DSS4. $out(\texttt{Receive}, \texttt{sid}_{\text{DIC}}, r)_{\text{Init}}$ Precondition: $r := rmes$ and $ntask = \text{DSS4}$, Effect: $rmes$ and $ntask := \perp$
– Expire Session:
EXS1. $in(\texttt{Expire}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{Init}}$ Precondition: $active = \top$ and $ntask = \perp$, Effect: $ntask := \text{EXS2}$
EXS2. $send(\texttt{Expire}_{\text{2AC}}, \texttt{sid}^{\text{X}}_{\text{2AC}})_{\text{F}^{\text{X}}_{\text{2AC}}}$ Precondition: $ntask = \text{EXS2}$, Effect: $active$ and $ntask := \perp$

Fig. 4. Code for Initiator of Direction-Indeterminable Channel, Init_{DIC}

Tasks $\overline{\text{Init}_{\text{DIC}}}$ and $\overline{\text{Rec}_{\text{DIC}}}$ relay the input messages from the environment to the ideal functionality task and relay the receive messages from the ideal functionality task to the environment as interface parties in the ideal system.

Theorem 1. *Direction-indeterminable channel protocol system* $\mathbf{Real}_{\mathbf{DIC}}$ *perfectly hybrid implements direction-indeterminable channel functionality system* $\mathbf{Ideal}_{\mathbf{DIC}}$ *with respect to adaptive adversary under any master schedule. (A direction-indeterminable channel is reducible to a two-anonymous channel with respect to adaptive adversary under any master schedule.)*

Let ϵ_{R} and ϵ_{I} be discrete probability measures on finite executions of $\mathbf{Real}_{\mathbf{DIC}} || \mathbf{Env}$ and $\mathbf{Ideal}_{\mathbf{DIC}} || \mathbf{Env}$, respectively. We prove the Theorem 1 by showing that ϵ_{R} and ϵ_{I} satisfy the trace distribution property : $tdist(\epsilon_{\text{R}}) = tdist(\epsilon_{\text{I}})$. Here, we define correspondence R between the states in $\mathbf{Real}_{\mathbf{DIC}} || \mathbf{Env}$ and the states in $\mathbf{Ideal}_{\mathbf{DIC}} || \mathbf{Env}$. We say $(\epsilon_{\text{R}}, \epsilon_{\text{I}}) \in R$ if and only if for every $s \in \text{supp.lst}(\epsilon_{\text{R}})$ and $u \in \text{supp.lst}(\epsilon_{\text{I}})$, all of the state correspondences in the Table 1 hold. We then prove R is a simulation relation in Lemma 1.

Code for Receiver of Direction-Indeterminable Channel, Rec_{DIC}, where $\text{sid}_{\text{DIC}} = (\{\text{Init}, \text{Rec}\}, \text{sid}'_{\text{DIC}})$, $\text{sid}^{\text{I}}_{2\text{AC}} = (\{\text{Init}, \text{Rec}\}, \text{Init}, \text{sid}'^{\text{I}}_{2\text{AC}})$ and $\text{sid}^{\text{R}}_{2\text{AC}} = (\{\text{Init}, \text{Rec}\}, \text{Rec}, \text{sid}'^{\text{R}}_{2\text{AC}})$.

State: $smes, rmes \in \{0,1\}^* \cup \{\bot\}$, initially $\bot$, $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$, $active \in \{\bot, \top\}$, initially $\bot$

Transitions:

– Establish Session:

ESS1. $in(\text{Establish}_{\text{DIC}}, \text{sid}_{\text{DIC}})_{\text{Rec}}$ Precondition: $active$ and $ntask = \bot$, Effect: $ntask := \text{ESS2}$

ESS2. $send(\text{Establish}_{2\text{AC}}, \text{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $ntask = \text{ESS2}$, Effect: $active := \top$ and $ntask := \bot$

– Data Sending Session:

DSS1. $in(\text{Send}, \text{sid}_{\text{DIC}}, m)_{\text{Rec}}$ Precondition: $active = \top$, $smes$ and $ntask = \bot$, Effect: $smes := m$ and $ntask :=$ DSS2

DSS2. $send(\text{Send}, \text{sid}^{\text{X}}_{2\text{AC}}, m)_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $m := smes$ and $ntask = \text{DSS2}$, Effect: $ntask := \bot$

DSS3. $receive(\text{Receive}, \text{sid}^{\text{R}}_{2\text{AC}}, m)_{\text{F}^{\text{R}}_{2\text{AC}}}$ Precondition: $active = \top$, $rmes$ and $ntask = \bot$, Effect: If $smes = \bot$, then $rmes := m$ and $ntask :=$ DSS4. Else $smes := \bot$ and $ntask := \bot$.

DSS4. $out(\text{Receive}, \text{sid}_{\text{DIC}}, r)_{\text{Rec}}$ Precondition: $r := rmes$ and $ntask = \text{DSS4}$, Effect: $rmes, ntask := \bot$

– Expire Session:

EXS1. $in(\text{Expire}_{\text{DIC}}, \text{sid}_{\text{DIC}})_{\text{Rec}}$ Precondition: $active = \top$ and $ntask = \bot$, Effect: $ntask := \text{EXS2}$

EXS2. $send(\text{Expire}_{2\text{AC}}, \text{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $ntask = \text{EXS2}$, Effect: $active$ and $ntask := \bot$

Fig. 5. Code for Receiver of Direction-Indeterminable Channel, Rec_{DIC}

Code fot Adversary for Direction Indeterminable Channel, Adv_{DIC}, where $X \in \{\text{I}, \text{R}\}$, where $\text{sid}^{\text{I}}_{2\text{AC}} = (\{I_1, I_2\}, I_1, \text{sid}'_{2\text{AC}})$ and $\text{sid}^{\text{R}}_{2\text{AC}} = (\{I_1, I_2\}, I_2, \text{sid}'_{2\text{AC}})$.

State: $active \in \{\bot, \top\}$, initially $\bot$, $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$, $smes_X \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$

Transitions:

– Establish Session:

ESS1. $receive(\text{SID}, \text{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $active = \bot$, Effect:$active := \top$

– Data Sending Session:

DSS**1.** $receive(\text{Send}, \text{sid}^{\text{X}}_{2\text{AC}}, mes)_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $active = \top$, $ntask = \bot$, Effect: $smes_X := mes$ and $ntask :=$ DSS2

DSS**2.** $send(\text{Response}, \text{sid}^{\text{X}}_{2\text{AC}}, ok)_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $ntask = \text{DSS2}$, Effect: $ntask := \bot$

– Expire Session:

EXS**1.** $receive(\text{Expire}_{2\text{AC}}, \text{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $active = \top$, Effect: $active := \bot$

– Other tasks:

This adversary makes other arbitary tasks.

Fig. 6. Code fot Adversary for Direction Indeterminable Channel, Adv_{DIC}

Lemma 1. *The relation R defined above is a simulation relation from* $\mathbf{Real_{DIC}||Env}$ *to* $\mathbf{Ideal_{DIC}||Env}$. *For each step of* $\mathbf{Real_{DIC}||Env}$*, the step in the establish, data sending and expire session correspond with at most two steps of* $\mathbf{Ideal_{DIC}||Env}$. *This means that there is a mapping corrtasks under the relation R such that, for every* ρ, T, $|corrtasks(\rho, T)| \leq 2$*, where* ρ *is a local schedule.*

Proof. (sketch)

We prove that R is a simulation relation from $\mathbf{Real_{DIC}||Env}$ to $\mathbf{Ideal_{DIC}||Env}$ using the mapping corrtasks : $R^*_{\mathbf{Real_{DIC}||Env}} \times R_{\mathbf{Real_{DIC}||Env}} \rightarrow R^*_{\mathbf{Ideal_{DIC}||Env}}$, which is defined as follows (we write hereafter $T =_{\text{corr.}} T'$ alternating to write corrtasks$(\rho, T) = T'$.):

Code for ideal Initiator of Direction-Indeterminable Channel, $\overline{\text{Init}_{\text{DIC}}}$, where $\texttt{sid}_{\text{DIC}} = (\{\overline{\text{Init}}, \overline{\text{Rec}}\}, \texttt{sid}'_{\text{DIC}})$.

State: *smes*, *rmes* $\in \{0,1\}^* \cup \{\bot\}$, initially $\bot$, *ntask* $\in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$, *active* $\in \{\bot, \top\}$, initially $\bot$
Transitions:
– Establish Session:
ESS1. $in(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\overline{\text{Init}}}$ Precondition: *active*, *ntask* $= \bot$, Effect: *ntask* $:=$ ESS2
ESS2. $send(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$ Precondition: *ntask* = ESS2, Effect: *active* $:= \top$ and *ntask* $:= \bot$
– Data Sending Session:
DSS1. $in(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\overline{\text{Init}}}$ Precondition: *active* $= \top$, *smes* and *ntask* $= \bot$, Effect: *smes* $:= m$ and *ntask* $:=$ DSS2
DSS2. $send(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{F}_{\text{DIC}}}$ Precondition: $m :=$ *smes* and *ntask* = DSS2, Effect: *smes* and *ntask* $:= \bot$
DSS3. $receive(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{F}_{\text{DIC}}}$ Precondition: *active* $= \top$, *rmes* and *ntask* $= \bot$, Effect: *rmes* $:= m$ and *ntask* $:=$ DSS4
DSS4. $out(\texttt{Receive}, \texttt{sid}_{\text{DIC}}, m)_{\overline{\text{Init}}}$ Precondition: $m :=$ *rmes* and *ntask* = DSS4, Effect: *rmes* and *ntask* $:= \bot$
– Expire Session:
EXS1. $in(\texttt{Expire}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\overline{\text{Init}}}$ Precondition: *active* $= \top$ and *ntask* $= \bot$, Effect:*ntask* $:=$ EXS2
EXS2. $send(\texttt{Expire}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$ Precondition: *ntask* = EXS2, Effect: *active* and *ntask* $:= \bot$

Fig. 7. Code for Initiator of Direction-Indeterminable Channel, $\overline{\text{Init}_{\text{DIC}}}$

Code for ideal Receiver of Direction-Indeterminable Channel, $\overline{\text{Rec}_{\text{DIC}}}$, where $\texttt{sid}_{\text{DIC}} = (\{\overline{\text{Init}}, \overline{\text{Rec}}\}, \texttt{sid}'_{\text{DIC}})$.

State: *smes*, *rmes* $\in \{0,1\}^* \cup \{\bot\}$, initially $\bot$, *ntask* $\in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$, *active* $\in \{\bot, \top\}$, initially $\bot$
Transitions:
– Establish Session:
ESS1. $in(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\overline{\text{Rec}}}$ Precondition: *active* and *ntask* $= \bot$, Effect: *ntask* $:=$ ESS2
ESS2. $send(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$ Precondition: *ntask* = ESS2, Effect: *active* $:= \top$ and *ntask* $:= \bot$
– Data Sending Session:
DSS1. $in(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\overline{\text{Rec}}}$ Precondition:*active* $= \top$, *smes* and *ntask* $= \bot$, Effect: *smes* $:= m$ and *ntask* $:=$ DSS2
DSS2. $send(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{F}_{\text{DIC}}}$ Precondition: $m :=$ *smes* and *ntask* = DSS2, Effect: *smes* and *ntask* $:= \bot$
DSS3. $receive(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{F}_{\text{DIC}}}$ Precondition: *rmes* and *ntask* $= \bot$, Effect: *rmes* $:= m$ and *ntask* $:=$ DSS4
DSS4. $out(\texttt{Receive}, \texttt{sid}_{\text{DIC}}, m)_{\overline{\text{Rec}}}$ Precondition: $m :=$ *rmes* and *ntask* = DSS4, Effect: *rmes ntask* $:= \bot$
– Expire Session:
EXS1. $in(\texttt{Expire}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\overline{\text{Rec}}}$ Precondition: *active* $= \top$, *smes*, *rmes* and *ntask* $= \bot$, Effect:*ntask* $:=$ EXS2
EXS2. $send(\texttt{Expire}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$ Precondition: *ntask* = EXS2, Effect: *active* and *ntask* $:= \bot$

Fig. 8. Code for ideal Receiver of Direction-Indeterminable Channel, $\overline{\text{Rec}_{\text{DIC}}}$

For any $(\rho, T) \in (R^*_{\mathbf{Real}_{\mathbf{DIC}}\|\mathbf{Env}} \times R_{\mathbf{Real}_{\mathbf{DIC}}\|\mathbf{Env}})$, the following task correspondences hold.

1. **Establish Session**
   (a) $\text{Init}_{\text{DIC}}.send(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}^{\texttt{X}}_{\texttt{2AC}})_{\text{F}^{\texttt{X}}_{\texttt{2AC}}} =_{\text{corr.}} \overline{\text{Init}_{\text{DIC}}}.send(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$
   (b) $\text{Rec}_{\text{DIC}}.send(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}^{\texttt{X}}_{\texttt{2AC}})_{\text{F}^{\texttt{X}}_{\texttt{2AC}}} =_{\text{corr.}} \overline{\text{Rec}_{\text{DIC}}}.send(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{F}_{\text{DIC}}}$
   (c) $\text{F}^{\texttt{X}}_{\texttt{2AC}}.send(\texttt{SID}, \texttt{sid}_{\texttt{2AC}})_{\text{Adv}} =_{\text{corr.}} \text{F}_{\text{DIC}}.send(\texttt{SID}, \texttt{sid}_{\text{DIC}})_{\text{Adv}}$
2. **Data Sending Session**
   (a) $\text{Init}_{\text{DIC}}.send(\texttt{Send}, \texttt{sid}^{\texttt{X}}_{\texttt{2AC}}, m)_{\text{F}^{\texttt{X}}_{\texttt{2AC}}} =_{\text{corr.}} \overline{\text{Init}_{\text{DIC}}}.send(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{F}_{\text{DIC}}}$
   (b) $\text{Rec}_{\text{DIC}}.send(\texttt{Send}, \texttt{sid}^{\texttt{X}}_{\texttt{2AC}}, m)_{\text{F}^{\texttt{X}}_{\texttt{2AC}}} =_{\text{corr.}} \overline{\text{Rec}_{\text{DIC}}}.send(\texttt{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{F}_{\text{DIC}}}$

Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$, where $\mathtt{sid}_{\mathrm{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathrm{DIC}})$.

State: *active* $\in \{\bot, \top\}$, initially $\bot$, *smes* $\in \{0,1\}^* \cup \{\bot\}$, initially $\bot$, *ntask* $\in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$
Other arbitrary variables; call "new" variables.
Transitions:
– Establish Session:
ESS1. *receive*$(\mathtt{SID}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$ Precondition: *active* and *ntask* $= \bot$, Effect:*active* $:= \top$ This task generates the parties Init and Rec in the $\mathbf{Real_{DIC}}$ system to simulate the real world. To make establish session in the simulation world, inputs *in*$(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Init}}$ and *in*$(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Rec}}$ to Init and Rec, respectively. Finally, the parties establish the two 2ACs in the simulation world.
– Data Sending Session:
DSS1. *receive*$(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{F_{DIC}}}$ Precondition: *active* $= \top$, *ntask* $= \bot$, Effect: *smes* $:=$ *m* and *ntask* $:=$ DSS2
DSS2. *simulation*$(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)$ Precondition: *mes* $:=$ *smes* and *ntask* $=$ DSS2, Effect: This task inputs *in*$(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Init}}$ to Init in the simulation world. During the simulation, if the adversary in this simulation wants to output message to the environment, this simulator outputs the message after receiving from the adversary. After simulating the real world, the simulator receives *out*$(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, r)_{\mathrm{Rec}}$ from the receiver Rec and sets *ntask* $:=$ DSS3
DSS3. *send*$(\mathtt{Response}, \mathtt{sid}_{\mathrm{DIC}}, ok)_{\mathrm{F_{DIC}}}$ Precondition: *ntask* $=$ DSS3, Effect: *ntask* $:= \bot$
– Expire Session:
EXS**1.** *receive*$(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$ Precondition: *active* $= \top$, Effect: *active* $:= \bot$
– Other tasks:
This simulator makes arbitary tasks to simulate the real world protocol system $\mathbf{Real_{DIC}}$. The tasks can be the input and output tasks with the internal tasks copied from $\mathbf{Real_{DIC}}$. Espectialy, this simulator can output the message from the adversary in the simiulating world to the environment.

Fig. 9. Code fot Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$

(c) $\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{2AC}}, mes)_{\mathrm{Adv}}$
$=_{\mathrm{corr.}} \mathrm{F_{DIC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Adv}} \cdot \mathrm{Sim_{DIC}}.simulation(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)$
(d) $\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.send(\mathtt{Receive}, \mathtt{sid}_{\mathrm{2AC}}, mes)_{\mathrm{Rec}} =_{\mathrm{corr.}} \mathrm{F_{DIC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)_{\overline{X}}$
(e) $\mathrm{Init_{DIC}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, r)_{\mathrm{Init}} =_{\mathrm{corr.}} \overline{\mathrm{Init_{DIC}}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\overline{\mathrm{Init}}}$
(f) $\mathrm{Rec_{DIC}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, r)_{\mathrm{Rec}} =_{\mathrm{corr.}} \overline{\mathrm{Rec_{DIC}}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, mes)_{\overline{\mathrm{Rec}}}$
(g) $\mathrm{Adv_{DIC}}.send(\mathtt{Response}, \mathtt{sid}^{\mathtt{x}}_{\mathrm{2AC}}, ok)_{\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}} =_{\mathrm{corr.}} \mathrm{Sim_{DIC}}.send(\mathtt{Response}, \mathtt{sid}_{\mathrm{DIC}}, ok)_{\mathrm{F_{DIC}}}$

3. **Expire Session**
(a) $\mathrm{Init_{DIC}}.send(\mathtt{Expire}_{\mathrm{2AC}}, \mathtt{sid}^{\mathtt{x}}_{\mathrm{2AC}})_{\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}} =_{\mathrm{corr.}} \overline{\mathrm{Init_{DIC}}}.send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$
(b) $\mathrm{Rec_{DIC}}.send(\mathtt{Expire}_{\mathrm{2AC}}, \mathtt{sid}^{\mathtt{x}}_{\mathrm{2AC}})_{\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}} =_{\mathrm{corr.}} \overline{\mathrm{Rec_{DIC}}}.send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$
(c) $\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.send(\mathtt{Expire}_{\mathrm{2AC}}, \mathtt{sid}_{\mathrm{2AC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F_{DIC}}.send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Adv}}$

4. All tasks of environment **Env** in $\mathbf{Real_{DIC}}$ are correspondent with the tasks of environment in $\mathbf{Ideal_{DIC}}$.

The simulation of $\mathrm{Sim_{DIC}}$ is perfectly done for establish session, data sending session and expire session with respect to the no corruption, static corruption and adaptive corruption by adversary.

1. **No corruption**
First, in the establish session, environment Env sends the establish session message *in*$(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Init}}}$ and *in*$(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Rec}}}$ to the initiator $\overline{\mathrm{Init_{DIC}}}$ and the receiver $\overline{\mathrm{Rec_{DIC}}}$, respectively. They send the establish session messages *send*$(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$ to $\mathrm{F_{DIC}}$. They send *send*$(\mathtt{SID}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Adv}}$ to

Table 1. State correspondence : Reduction of DIC to 2AC

Functionality		**Receiver**	
(a) $u.\mathrm{F}_{\mathrm{DIC}}.estcond_{\mathrm{Init}}$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.estcond_{\mathrm{Init}_i}$	(k) $u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.smes$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.smes$
(b) $u.\mathrm{F}_{\mathrm{DIC}}.estcond_{\mathrm{Rec}}$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.estcond_{\mathrm{Rec}}$	(l) $u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.rmes$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.rmes$
(c) $u.\mathrm{F}_{\mathrm{DIC}}.okcond_{\mathrm{Adv}}$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.okcond_{\mathrm{Adv}}$	(m) $u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.active$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.active$
(d) $u.\mathrm{F}_{\mathrm{DIC}}.active$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.active$	(n) $u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.ntask$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.ntask$
(e) $u.\mathrm{F}_{\mathrm{DIC}}.mes$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.mes$		
(f) $u.\mathrm{F}_{\mathrm{DIC}}.ntask$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.ntask$		
Initiator		**Adversary** and **Env**	
(g) $u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.smes$	$= s.\mathrm{Init}_{\mathrm{DIC}}.smes$	(o) $u.\mathrm{Sim}_{\mathrm{DIC}}$	$= s.\mathrm{Adv}_{\mathrm{DIC}}$
(h) $u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.rmes$	$= s.\mathrm{Init}_{\mathrm{DIC}}.rmes$	(p) $u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.*$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.*$
(i) $u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.active$	$= s.\mathrm{Init}_{\mathrm{DIC}}.active$	(q) $u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.*$	$= s.\mathrm{Init}_{\mathrm{DIC}}.*$
(j) $u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.ntask$	$= s.\mathrm{Init}_{\mathrm{DIC}}.ntask$	(r) $u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.*$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.*$
		(s) $u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Adv}_{\mathrm{DIC}}.*$	$= s.\mathrm{Adv}_{\mathrm{DIC}}.*$
		(t) $u.\mathrm{Env}$	$= s.\mathrm{Env}$

Note that $ntask \in \{\mathrm{ESS1}, \mathrm{ESS2}, \mathrm{DSS1}, \mathrm{DSS2}, \mathrm{DSS3}, \mathrm{DSS4}, \mathrm{EXS1}, \mathrm{EXS2}\}$, $i \in \{1, 2\}$ and $X \in \{I, R\}$.

the $\mathrm{Sim}_{\mathrm{DIC}}$. After $\mathrm{Sim}_{\mathrm{DIC}}$ receives the message, $\mathrm{Sim}_{\mathrm{DIC}}$ generates the parties Init and Rec in his simulation world to make the real world situation which Init and Rec exchange messages by using $\mathrm{F}_{\mathrm{2AC}}$. $\mathrm{Sim}_{\mathrm{DIC}}$ then make establish session in the simulation world. That is, he inputs two messages, $in(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Init}}$ and $in(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Rec}}$, to Init and Rec, respectively. Finally, the parties establish two 2ACs in the simulation world.

Next, in the data sending session, Env sends the message $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\overline{\mathrm{Init}}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\overline{\mathrm{Rec}}}$) to $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ (or $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$). $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ sends $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. $\mathrm{F}_{\mathrm{DIC}}$ then sends $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Adv}}$ to $\mathrm{Sim}_{\mathrm{DIC}}$. After receiving the message, $\mathrm{Sim}_{\mathrm{DIC}}$ executes $simulation(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)$ to mimic the data sending session of the real world. That is, he inputs the message $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Init}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Rec}}$) to Init and Rec in the simulation world.

Finally, in the expire session, Env sends the messages $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Init}}}$ and $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Rec}}}$ to $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$, respectively. They relay the message $send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. After receiving $send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Adv}}$ from $\mathrm{F}_{\mathrm{DIC}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ expires the session in the simulation world. That is, he inputs the messgae $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Init}}$ and $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Rec}}$ to Init and Rec in the simulation world.

2. **Static corruption**
   In this case, the advesary corrupt some parties before the protocol starts. This case also is simulated by the simulator, but the direction of message sending does not conceal to the advesary.

3. **Adaptive corruption**
   In this case, the advesary corrupt some parties when he want to do so. This case also is simulated by the simulator, but the direction of message sending does not conceal to the advesary after he corrupts some parties.

As a result, the simulation is perfectly done because $\mathrm{Sim}_{\mathrm{DIC}}$ can simulate the real world from the information message through $\mathrm{F}_{\mathrm{DIC}}$. The tasks of the real world perfectly correspond with the the tasks of ideal world. That is,

$$\mathbf{Real_{DIC}}||\mathbf{Env} \ \ \mathrm{Hyb}. \leq_0^{\mathtt{M}_{\pi_{\mathrm{DIC}}}} \mathbf{Ideal_{DIC}}||\mathbf{Env}.$$

The task sequence of the system $\mathbf{Real_{DIC}}||\mathbf{Env}$ are perfectively corresponded with the task sequence of the system $\mathbf{Ideal_{DIC}}||\mathbf{Env}$ under the schedule $M_{\pi_{\mathrm{DIC}}}$. Formally, to prove that R is simulation relation from $\mathbf{Real_{DIC}}||\mathbf{Env}$ to $\mathbf{Ideal_{DIC}}||\mathbf{Env}$, we need to show R satisfies start condition and step condition for each corresponding tasks, but we omit to mention it due to the paper limitation. See full paper that will be available soon.

4.2 Reduction of 2AC to DIC

Let $\pi_{2\mathrm{AC}}$ be a protocol of two-anonymous channel. We assume that $M_{\pi_{2\mathrm{AC}}}$, the master schedule of $\pi_{2\mathrm{AC}}$, is any schedule. Let $\mathrm{Init}_{2\mathrm{AC}}$ and $\mathrm{Rec}_{2\mathrm{AC}}$ be the initiator's code and receiver's code for a real system, respectively. Let $\overline{\mathrm{Init}_{2\mathrm{AC}}}$ and $\overline{\mathrm{Rec}_{2\mathrm{AC}}}$ be the initiator's code and receiver's code for an ideal system, respectively. Finally, let $\mathrm{Adv}_{2\mathrm{AC}}$ and $\mathrm{Sim}_{2\mathrm{AC}}$ be the adversary's code and the simulator's code, respectively. Let $\mathbf{Real_{2AC}}$ and $\mathbf{Ideal_{2AC}}$ be a two-anonymous channel protocol system and a two-anonymous channel functionality system, respectively, defined as follows:

$$\mathbf{Real_{2AC}} := \mathrm{Init}_{2\mathrm{AC}}||\mathrm{Rec}_{2\mathrm{AC}}||\mathrm{Adv}_{2\mathrm{AC}}||\mathrm{F}_{\mathrm{DIC}},$$
$$\mathbf{Ideal_{2AC}} := \overline{\mathrm{Init}_{2\mathrm{AC}}}||\overline{\mathrm{Rec}_{2\mathrm{AC}}}||\mathrm{Sim}_{2\mathrm{AC}}||\mathrm{F}_{2\mathrm{AC}}.$$

Tasks $\overline{\mathrm{Init}_{2\mathrm{AC}}}$ and $\overline{\mathrm{Rec}_{2\mathrm{AC}}}$ relay the input messages from the environment to the ideal functionality task and relay the messages received from the ideal functionality task to the environment as interface parties in the ideal system. Several codes for each tasks are omitted in this paper, see full paper version.

Theorem 2. *Two-anonymous channel protocol system* $\mathbf{Real_{2AC}}$ *perfectly hybrid implements two-anonymous channel functionality system* $\mathbf{Ideal_{2AC}}$ *with respect to adaptive adversary under any master schedule. (An anonymous channel is reducible to a direction-indeterminable channel with respect to adaptive adversary under any master schedule.)*

The proof of theorem 2 is described like theorem 1. We omit the proof in this paper, see the full paper version.

5 Equivalence between DIC and SC

In this section, we prove that the direction indeterminable channel (DIC) is equivalent to secure channel (SC) under a specific type of some schedules. That is, the task-PIOA of DIC perfectly implements task-PIOA of SC under an asynchronous schedule. To prove this, we show two reductions of SC to DIC and one of DIC to SC. Here, we consider the one bit message exchange, that is, $|m| = 1$. Informally, the reduction of SC to DIC is proven as follows: To make the channel between Init and Rec secure, the parties

exchange a random bit (as a secret shared key) by DIC. The encrypted message by the shared key is exchanged using a public channel. The communication is done not by a DIC channel but by a public channel. When the next message sending is occured, party restart from key exchange. Here, the key exchange is done under the master schedule. After the key exchange, the cipher text generated by the secret key is sent. The other reduction of DIC to SC is proven as follows: the parties Init and Rec exchange two messages by SC. The one is the message m which the sender wants to send. The other message is a dummy message to conceal the message direction. That is, sender Init sends message m and the receiver sends dummy message s under a specific type of schedules by M. We make a random message s by $\mathrm{F}_{\mathrm{SRC}}$. Note that, the adversary cannot know the direction of message because the messages are exchanged under a specific type of schedules. In this section, we need to consider the schedules (key exchange schedule and message exchange schedule) to avoid some infromation to adversary. In the UC framework, all schedule is under control of adversary. So, we use task PIOA framework.

5.1 Reduction of SC to DIC

Let n be the number of parties. Let $M_{psync}(t_1^*, \cdots, t_n^*)$ and $M_{rasync}(t_1^*, \cdots, t_n^*)$ be master schedules, respectively, where t_i^* is a task in party P_i.

Definition 10. **[$M_{psync}(t_1^*, \cdots, t_n^*)$]** *Let t_i^* be a task in party P_i. Let $ptask(t_i^*)$ be the task just before t_i^* in the local scheduler ρ_i. For example, let $\rho_i = t_{i1}, t_{i2}, t_{i3}$ for party P_i. Then $ptask(t_{i3})$ is the task t_{i2}.*

- *1. Alignment property: After the master scheduler M activates $ptask(t_i^*)$, M does not activate P_i until all of $ptask(t_1^*), \cdots, ptask(t_n^*)$ are scheduled. This situation say that M satisfies the alinment property for the specified tasks $t_1^*, \ldots, t_n^*$.*
- *2. Random executing property: The master scheduler, M, grobally executes the specified tasks, $t_1^*, \ldots, t_n^*$ in a random order. Note that the other tasks are not scheduled until all of the specified tasks, $t_1^*, \ldots, t_n^*$, finish executing.*

$M_{psync}(t_1^*, \cdots, t_n^*)$ is defined to be a master schedule such that a master scheduler M satisfies the avobe mentioned two properties for the specified tasks $t_1^*, \ldots, t_n^*$.

Definition 11. **[$M_{rasync}(t_1^*, \cdots, t_n^*, k)$]** *Let k be a integer. Let t_i^* be a task specified by ρ_i for party P_i. Let c_i be the number of times t_i^* is scheduled by M. M schedules the task acctivations of $t_1^*, \cdots, t_n^*$ so that $|c_i - c_j| \leq k$ for all i, j in a random order.*

We need to consider like chernov bound property if we treat this master schedule to use the message exchange or key exchange among party.

Let π_{SC} be a protocol of secure channel. Let $M_{\pi_{\mathrm{SC}}}$ be $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F}_{\mathrm{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F}_{\mathrm{DIC}}})$. Let $\mathrm{Init}_{\mathrm{SC}}$ and $\mathrm{Rec}_{\mathrm{SC}}$ be the initiator's code and receiver's code for a real system, respectively. Let $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$ be the initiator's code and receiver's code for an ideal system, respectively. Finally, let $\mathrm{Adv}_{\mathrm{SC}}$, $\mathrm{Sim}_{\mathrm{SC}}$ and $\mathrm{F}_{\mathrm{SRC}}$ be the adversary's code, the simulator's code and the random bit generator's

code, respectively. Let $\mathbf{Real_{SC}}$ and $\mathbf{Ideal_{SC}}$ be a secure channel protocol system and a secure channel functionality system, respectively, defined as follows:

$$\mathbf{Real_{SC}} := \mathrm{Init_{SC}}||\mathrm{Rec_{SC}}||\mathrm{Adv_{SC}}||\mathrm{F_{SRC}}||\mathrm{F_{DIC}},$$
$$\mathbf{Ideal_{SC}} := \overline{\mathrm{Init_{SC}}}||\overline{\mathrm{Rec_{SC}}}||\mathrm{Sim_{SC}}||\mathrm{F_{SC}}.$$

Tasks $\overline{\mathrm{Init_{SC}}}$ and $\overline{\mathrm{Rec_{SC}}}$ relay the input messages from the environment to the ideal functionality task and relay the receive messages from the ideal functionality task to the environment, respectively, as interface parties in the ideal system. Several codes for each tasks are omitted in this paper, see full paper version.

Theorem 3. *Secure channel protocol system* $\mathbf{Real_{SC}}$ *perfectly hybrid implements secure channel functionality system* $\mathbf{Ideal_{SC}}$ *with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, s)_{\mathrm{F_{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, t)_{\mathrm{F_{DIC}}})$. *(A secure channel is reducible to a direction-indeterminable channel with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}s)_{\mathrm{F_{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, t)_{\mathrm{F_{DIC}}})$.*)*

The proof of theorem 3 is described like theorem 1. We omit the proof in this paper, see the full paper version. The master schedule can be M_{rasync} instead of M_{psync}.

5.2 Reduction of DIC to SC

Let π'_{DIC} be a protocol of direction-indeterminable channel. Let $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}})$ be the master schedule for π'_{DIC}.

Let $\mathrm{Init}'_{\mathrm{DIC}}$ and $\mathrm{Rec}'_{\mathrm{DIC}}$ be the initiator's code and receiver's code for a real system, respectively. Let $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$ be the initiator's code and receiver's code for an ideal system, respectively. Finally, let $\mathrm{Adv}'_{\mathrm{DIC}}$ and $\mathrm{Sim}'_{\mathrm{DIC}}$ be the adversary's code and the simulator's code, respectively. Let $\mathbf{Real'_{DIC}}$ and $\mathbf{Ideal'_{DIC}}$ be a direction-indeterminable channel protocol system and a direction-indeterminable channel functionality system defined, respectively, as follows:

$$\mathbf{Real'_{DIC}} := \mathrm{Init}'_{\mathrm{DIC}}||\mathrm{Rec}'_{\mathrm{DIC}}||\mathrm{Adv}'_{\mathrm{DIC}}||\mathrm{F_{SRC}}||\mathrm{F_{SC}},$$
$$\mathbf{Ideal'_{DIC}} := \overline{\mathrm{Init}'_{\mathrm{DIC}}}||\overline{\mathrm{Rec}'_{\mathrm{DIC}}}||\mathrm{Sim}'_{\mathrm{DIC}}||\mathrm{F_{DIC}}.$$

Tasks $\overline{\mathrm{Init}'_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}'_{\mathrm{DIC}}}$ relay the input messages from the environment to the ideal functionality task and relay the receive messages from the ideal functionality task to the environment, respectively, as interface parties in the ideal system. Several codes for each tasks are omitted in this paper, see full paper version.

Theorem 4. *Direction-indeterminable channel protocol system* $\mathbf{Real'_{DIC}}$ *perfectly hybrid implements direction-indeterminable channel functionality system* $\mathbf{Ideal'_{DIC}}$ *with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}})$. *(A direction-indeterminable channel is reducible to a secure channel with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}})$.*)*

The proof of theorem 4 is described like theorem 1. We omit the proof in this paper, see the full paper version.

6 Conclusion

This paper studied the relationship of the three cryptographic channels, secure channels (SC), two-anonymous channels (2AC) and direction-indeterminable channels (DIC), by considering communication schedules and composable security. For this purpose, we adopted the universally composable (UC) framework with the task-probabilistic input/output automata (PIOA) model. We showed that the three channels are reducible to each other under some types of schedules in the UC framework with the PIOA model.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In: Proc. of STOC, pp. 1–10 (1988)
2. Canetti, R.: Universally Composable Security: A New paradigm for Cryptographic Protocols. 42nd FOCS, IACR ePrint Archive 2000/067 (2001), http://eprint.iacr.org
3. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Taskstructured probabilistic I/O automata. In: Proc. of WODES 2006 (2006)
4. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Time-bounded task-PIOAs: a framework for analyzing security protocols. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 238–253. Springer, Heidelberg (2006)
5. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Using probabilistic I/O automata to analyze an oblivious transfer protocol. Technical Report MIT-CSAIL-TR-2006-046, CSAIL, MIT, 2006. This is the revised version of Technical Reports MIT-LCS-TR-1001a and MIT-LCS-TR-1001 (2006)
6. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Using Task-Structured Probabilistic I/O Automata to Analyze an Oblivious Transfer Protocol. This is a revised version of Technical Report MIT-CSAIL-TR-2006-046, http://eprint.iacr.org
7. Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Proc. of STOC, pp. 11–19 (1988)
8. Håstad, J.: Pseudo-Random Generators under Uniform Assumptions. In: Proc. of STOC (1990)
9. Impagliazzo, R., Levin, L., Luby, M.: Pseudo-Random Number Generation from One-Way Functions. In: Proc. of STOC, pp. 12–24 (1989)
10. Naor, M.: Bit Commitment Using Pseudo-Randomness. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 128–136. Springer, Heidelberg (1990)
11. Naor, M., Yung, M.: Universal One-Way Hash Functions and Their Cryptographic Applications. In: Proc. of STOC, pp. 33–43 (1989)
12. Okamoto, T.: On the Relationship among Cryptographic Physical Assumptions. In: Ng, K.W., Balasubramanian, N.V., Raghavan, P., Chin, F.Y.L. (eds.) ISAAC 1993. LNCS, vol. 762, pp. 369–378. Springer, Heidelberg (1993)
13. Rompel, J.: One-Way Functions are Necessary and Sufficient for Secure Signature. In: Proc. of STOC, pp. 387–394 (1990)

A Universally Composable Framework for the Analysis of Browser-Based Security Protocols*

Sebastian Gajek**

Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany
sebastian.gajek@nds.rub.de

Abstract. Browser-based security protocols perform cryptographic tasks within the constraints of commodity browsers. They are the bearer protocols for many security critical applications on the Internet. Roughly speaking, they are the offspring of key exchange and secure sessions protocols. Although browser-based protocols are widely deployed, their security has not been formally proved. We provide a security model for the analysis of browser-based protocols based on the Universal Composability framework.

Keywords: Universal Composability, browser-based protocols, security model.

1 Introduction

The World Wide Web has become an important means in modern society. Among the technologies that have proliferated recently are browser-based applications. They exploit the advantages of a client application with standard interfaces in order to interact with distributed systems. Protocols realizable within the constraint of commodity browsers are called browser-based protocols. They lay the foundations for numerous Web applications (e.g. eCommerce, eVoting). Browser-based security protocols are the offspring of key exchange and secure sessions protocols. They transfer numerous cryptographic protocols to concrete applications using the design principles of the layered Internet approach. Important examples include federated identity management protocols which adopt the Needham-Schroeder-Loewe protocol, calling a trusted third party in order to receive an authentication token whereby the primitives to realize the protocols are limited to a set of network protocols (e.g. TCP, SSL/TLS) and some minor browser functionalities.

Although in many Internet settings browser-based protocols carry highly vital and sensitive information (e.g. credit card numbers, passwords, votes), their security is still unproven. The security analysis on browser-based protocols is

* Please contact the author for the full version.

** The author was supported by the European Commission (IST-2002-507932 ECRYPT).

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 283–297, 2008.

typically based on informal vulnerability analyses. Whenever flaws in browser-based schemes are reported, countermeasures are introduced leading to a costly and inconvenient revision of the broken scheme. However, it is well-known that security proofs, but not vulnerability analyses give the desired guarantee for the security. Unfortunately, the lack of precise protocol definitions and of underlying formal models hampers the systematic design and a cryptographically faithful analysis of browser-based protocols—even though elaborated security models and methods already exist (e.g. [8,2,17,14,3,20,4]).

Many obstacles must be overcome in order to transfer these techniques to browser-based protocols. Traditional analyses of cryptographic protocols presuppose the existence of protocol machines that precisely execute the protocol specification (unless the machine is corrupt), without its integration in a more comprehensive system and relation to surrounding protocols. By contrast, the browser machine operates as an auxiliary device for the user with its own behavior that influences protocol security. The browser mediates messages coming from the user and the server. It responds to a set of predefined messages, adds some information, merges messages from different protocol layers and sessions, stores long-term and short-term secrets, and most notably is controllable by the invocation of scripts. In order to assist the user, the browser enforces certain security policies. These policies are related to underlying network protocols and contribute to the security of browser-based protocols. In fact, many protocol vulnerabilities have been exposed by considering the subtleties of nested network protocols which do *not* necessarily have a cryptographic purpose [18,11,21]. Another obstacle arises from the fact that the user performs some cryptographic tasks. Browser-based protocols prefer the anonymous user model of authentication [20]. The browser does not implicitly authenticate to the server, but executes a higher-layer protocol to send some password over a server-only authenticated channel. The user (on behalf of a protocol machine) authenticates the server and thus becomes an active protocol participant. Hence, certain assumptions must be made about the user and her behavior.

Contribution. We present a security model for the analysis of browser-based protocols. Our work is inspired by the Universal Composability (UC) framework [3] which is based on the *ideal-world/real-world paradigm*. In the real world, parties execute a protocol in order to complete a cryptographic task, while in the ideal world, parties use a trusted ideal functionality for the computation of the same task. A protocol is said to be secure if any attack on a real protocol can also be carried out in the ideal model. Since the behavior in the ideal world is considered to be safe security of the real protocol is implied. The main attraction of the model is that security holds under general composition with arbitrary protocols and players. Our contribution is to transfer the theory of the widely deployed UC framework to a concrete security model for the analysis of browser-based security protocols (BBUC). Since browser-based protocols follow the design principles of the layered Internet approach and synthesize (cryptographic) tasks from multiple protocol layers, we define a "crypto API" of security-relevant ideal functionalities. They capture native browser functionalities, are useful for the composition

of browser-based security protocols, and considerably simply their analysis. In detail, our contribution includes:

1. We present a socket functionality that bootstraps higher-level protocols, a secure communication sessions functionality that captures the task of SSL/TLS, and an identity resolution functionality that mimics the task of domain name resolution protocols. We explicitly take into account browser scripting functionalities and devise a model for the aggregation of messages from potentially dishonest players. This provision ensures that native browser-based protocols can be simulated (e.g. cookie-based and form-based secure communication sessions).
2. We revise the traditional role of the client and consider the user and the browser to be two distinct players. Our definition of user behavior is generic and captures the task that the user reveals her secrets when she identifies a perceivable human authenticator (ideal user model) or a set thereof which are in some predefined domain (relaxed user model). The definitions are independent of our framework and transferable to the analysis of protocols in which the user is an active participant and her behavior has an impact on the security of the protocol.
3. We extend the meanwhile classical adversarial model for key exchange and secure sessions protocols as proposed in, e.g., [20] to the setting where the adversary is able to attack certain functionalities. Our model includes attacks against the user, who potentially discloses her credentials (e.g. phishing), attacks against the identity resolution mechanism which influences certain browser security policies and the composition of scripting functionalities (e.g. DNS spoofing), and attacks against the server, enabling injections of malicious messages into a secure communication (e.g. cross-site-scripting).

The generic specification of our BBUC model has the advantages that changes in commodity browsers due to updates and add-ons can be easily captured by reformulating the presented functionalities and augmenting our model with additional functionalities.

2 Relation to Previous Work

Groß, Pfitzmann and Sadeghi have proposed a browser model [12] that is based on the Reactive Simulatability framework due to Pfitzmann and Waidner [17]. Their work represents the first attempt to lay a firm foundation for the analysis of browser-based protocols. Basically, their work captures a state machine model of the HTTP protocol. The underlying communication channels are provided by an ideal channel machine. The authors do not demonstrate that real-world protocols, such as TCP and TLS, securely emulate such functionality. Further, the authors exclude relevant network functionalities. In particular, they do not consider identity resolution mechanisms. These are of prime interest for the enforcement of the browser-based security policies. In addition, their model neglects some very important browser functionalities. Most notably, it includes

message aggregation and script execution by the browser. Moreover, their model makes strong assumptions about user behavior, i.e. users are assumed to validate the server identity. However, this assumption clashes with recent research results which show that average Internet users are unable to deal properly with the verification process. Lastly, the paper considers a weak adversarial model. Due to the fact that relevant network and browser functionalities are not captured, many practical attacks are omitted which have led to security breaches of browser-based protocols.

Another relevant paper has been published by Gajek et. al. [10]. These authors propose a browser-based model based on the Bellare and Rogaway model for key exchange [2]. Their work includes a real-user model that is comparable with our relaxed certificate user model (see Section 3.3). Although the authors make a first attempt to capture relaxed user behavior and approach real-world assumptions, they do not consider all feasible adversaries. In particular, they exclude corruptive behavior. Further, their work does not provide any composition guarantees. However, these are required in order to decrease the analytical complexity, especially for multi-party browser-based protocols such as federated identity management protocols.

3 BBUC Model

In this section, we specify a toolkit of functionalities that serve as primitives for the composition of browser-based security protocols. We assume the familiarity with the UC framework and browser-based protocols.

3.1 Notations

Let $\mathtt{A}$ be an algorithm. By $y \leftarrow \mathtt{A}(x)$ we denote that y was obtained by running $\mathtt{A}$ on input x; $x|y$ denotes the concatenations of two elements x and y and $x \circ y$ the expansion of value x with a constant y. Let $p_i : \mathbb{N} \rightarrow \mathbb{N}$, $i \in \mathbb{N}$ be a polynomial and k the security parameter. By $x \stackrel{r}{\leftarrow} \{0,1\}^{p_i(k)}$ we denote a value by selecting a random element from $\{0,1\}^{p_i(k)}$. We consider the server S, the browser B, and the human user U as participants of a browser-based security protocol π. The players are represented by Interactive Turing Machines (ITMs) as described in [3]. Contrary to the current cryptographic literature, we explicitly model the user U as a stand-alone ITM. (Clearly, the user ITM has a limited contingent of functionalities. We discuss the issue in Section 3.3). By C we sometimes denote the client given by a pair (U, B). Further, the players may have different roles. By I we denote the initiator, i.e. the invoking player, and by R the responder, i.e. the reacting player. By $\perp$ we denote the anonymous identity, i.e. a player who does not reveal its identity. Let $P \in (I, R)$ be a set consisting of the two identities I and R. By $\bar{P}$ we denote the inverse, i.e. (R, I). Let $\mathcal{F}$ be the specification of an ideal functionality. Then, we refer to a responder-only authenticated functionality as $\mathcal{F}^1$, a mutual authenticated as $\mathcal{F}^2$, and a hybrid thereof as $\mathcal{F}^{(1,2)}$.

3.2 Network Services

While many theoretical models underline the abstraction of real-world protocols and thus differ in many concerns, the design of browser-based protocols especially for the purpose of security is constricted within the well-defined architecture of the layered Internet approach where layers provide and consume certain services and offer surrounding layers their functionalities through interfaces. Given the abstract description of the interfaces, different network and communication protocols realize the services.

Following this approach, we describe the functionalities according to the *most relevant* services we expect from the layers. Clearly, we would have wished to specify for each Internet layer an ideal functionality. However, such a treatment would have unnecessarily complicated our model without a contribution to the security of browser-based protocols. One of the reasons is that the network and lower layer protocols send messages in clear. Consequently, the adversary can freely manipulate the messages. In fact, the functionalities on these layers are captured by the adversary's ability to send, modify and delay messages in arbitrary directions. We therefor focus on the higher-layer functionalities and look about their real-world realizations in commodity browsers.

We start with the formulation of the identity resolution functionality $\mathcal{F}_{\text{IR}}$. It allows for linking dynamic to static identities and mimics the task of domain and address resolution protocols as offered by the DNS protocol. Next, we formulate a socket functionality that feeds any surrounding protocol with a globally unique session identifier. The functionality is useful to establish an active connection and captures a service which is provided among others by the 4-way TCP handshake protocol. Lastly, we present a secure communication functionality that ensures secure message transmission. It captures the service provided by the transport security layer protocols. In commodity browser, the TLS protocol emulates the transport security layer, and thus is the heart of browser-based security protocols. The presented functionalities are intrinsic browser primitives to establish a secure connection between two parties and lay the first foundations for browser-based security protocols.

Dynamic Identity Resolution. On the Internet, there exist many different mechanisms to address a distributed machine. Apparently, this is due to the fact that browsers aggregate protocols from different layers and each protocol layer may have its own type of identification mechanism to name the target machine. Prominent examples include MAC, IP addresses or domain names. Commonly, these mechanisms are dynamic, i.e. one can resolve the types of identifiers by executing a resolution protocol. Unfortunately, these concepts have a technical character and are inconvenient for average users, browsing on the Internet. Against this background, *unified resource locators* (URLs) have been introduced. They provide users with ease of use quantifiers to contact peer machines.

Definition 1. *Let* $\mathcal{URL} \supseteq \mathcal{PROTOCOL} \times (\mathcal{DOMAIN} \cup \mathcal{ADDRESS}) \times \mathcal{PATH} \times \mathcal{PORT} \times \mathcal{ATTRIBUTE}$ *denote the domain of unified resource locators. Then* $\mathcal{PROTOCOL}$ *denotes the set of protocols,* $\mathcal{DOMAIN}$ *the set of*

domain names, $\mathcal{ADDRESS}$ the set of IP addresses, $\mathcal{PATH}$ the set of directories to access a file, $\mathcal{PORT}$ the set of ports, and $\mathcal{ATTRIBUTE}$ the set of attributes for parameterizing surrounding processes.

Throughout the remainder of the paper, we will use $\texttt{URL} \in \mathcal{URL}$ to address a target machine. We do not make any restrictions on the unified resource locator except that it is unique. Conversely, different URLs may point to the same target machine. In order to resolve the ultimate target machine, we formulate a resolution functionality $\mathcal{F}_{\text{IR}}$ that captures the dynamic resolution of URLs. This process is essential for aggregation of content, as we will see in Section 3.4.

Description of the Identity Resolution Functionality. We illustrate functionality $\mathcal{F}_{\text{IR}}$ in the full version. The functionality maintains a database of all domains and addresses $\mathcal{T}^{\texttt{ir}}$, and resolves domains into addresses through the resolution function t. Note that the functionality does not provide any security guarantees except that it checks that a resolution is valid, i.e. within the universe of predefined domains and addresses. Further, the functionality enables the adversary to fix the resolved identity. Although browsers and servers implement several techniques to prevent a false DNS look-ups, it has been shown that the protection measures are vulnerable. An attacker can easily thwart the countermeasures and fake the resolution [16,15]. This is a straight consequence from the fact that the adversary controls the network and the DNS protocol provides no message authenticity. It is a simple ping-pong protocol that transports messages in clear.[1]

Theorem 1. *Protocol DNS securely realizes $\mathcal{F}_{\text{IR}}$.*

The proof including a specification of the DNS protocol appears in the full version.

Connection Establishment. When a real-world protocol wishes to set up a connection, it must specify the end point address of that entity. To provide higher-level functionalities with a standard set of primitives for connection establishment, sockets have been introduced. Sockets describe an interface for higher-level protocols such that they can be implemented on different networks without having to worry about the network configuration. Sockets enable applications to perform inter-process communication, most commonly across a computer network. All modern operating systems have some implementation of the socket interface, as it became the standard interface for connecting to the Internet. This interface implementation is implicit for transport layer protocols, and it is therefore one of the fundamental technologies underlying the Internet.

Definition 2. *Let $\mathcal{SOCKET}$ denote the socket space in the domain $\mathcal{ADDRESS} \times \mathcal{PORT} \subset \mathcal{URL}$. Then $\texttt{Socket} \in \mathcal{SOCKET}$ denotes a socket,* `Socket.Address` *the socket address, and* `Socket.Port` *the socket port.*

[1] There are some initiatives to incorporate cryptography into DNS (aka DNSSEC). However, the protocol is not widely implemented.

In order to capture the task of connection establishment, we formulate the socket functionality $\mathcal{F}_{\text{SOCKET}}$. It assures that a session identifier between two players is established so that requests by other functionalities arrive at the respective peer. Technically, it serves with some "setup information" to bootstrap the higher-level functionalities. Given a globally unique session identifier is a central point for the composition of arbitrary protocol functionalities under the security definition of universal composability. Otherwise, it could not be guaranteed that a protocol remains secure when run concurrently with arbitrary other protocols. A dishonest player could fix an identifier which has been used in the past or origins from a concurrently running session. We present a specification of the socket functionality in the full version.

Description of the Socket Functionality. The functionality securely negotiates the session identifier *sid* between two players. In order to ensure that no party reuses the session identifier, $\mathcal{F}_{\text{SOCKET}}$ maintains a repository $\mathcal{T}^{\texttt{sid}}$ of all session identifiers. Basically, the socket functionality is a relaxation of the init functionality Barak *et. al.* propose for protocol initialization in the UC framework [1]. The relaxation includes the following properties: Firstly, it is sufficient that the invocation of the socket functionality is locally unique, when the functionality is parameterized by globally unique identities. Since these identities are part of the input, the resulting value is globally unique. The negotiated channel identifier *cid* is locally unique between two players. Secondly, the output is independent of the invoking functionality. An arbitrary functionality may employ the session identifier for invocation. Hence, additional mechanisms are necessary to associate the session identifier with some functionality. (Here, the browser determines the association.) Thirdly, we add a release subroutine to update $\mathcal{T}^{\texttt{sid}}$ and erase the session identifier. This, however, is only for the purpose of completeness. Technically, one may think of a scheduler that deletes entries after the session timed-out or either player has suggested terminating the connection.

We highlight some additional properties of $\mathcal{F}_{\text{SOCKET}}$ in the following: The functionality exists in a single instance per player and is invoked with a fixed identifier and the name of the owner of that instance (we leave out from the definition for simplicity). Otherwise, it would be infeasible to manage a database $\mathcal{T}^{\texttt{sid}}$ that prevails the uniqueness of all active sessions. Further, the adversary opts for the session identifier, unless the session identifier is not used. The functionality simply captures the fact that there are no colluding sessions between the session initiator and the responder. It makes no security guarantees about the peer identity. In fact, the initiator can end up with a connection to a peer which was chosen by the adversary, and the adversary can release the connection at any time. This captures the fact that the adversary controls the network and is capable of sending, intercepting or delaying transmitted messages. The underlying transport protocols (and lower layer protocols) do not provide any cryptographic mechanisms to prevent that the adversary controls the message flow. Even though some non-cryptographic measures are implemented in browsers and servers, they do not protect against manipulation. Prevailed examples that demonstrate the security deficiencies of transport and lower layer protocols are

TCP spoofing and TCP flooding where the adversary chooses an arbitrary IP address and sends arbitrary messages on behalf of the party, respectively.

Theorem 2. *Protocol* TCP^{4-way} *securely realizes* $\mathcal{F}_{\text{SOCKET}}$.

The proof including the specification of the 4-way TCP handshake appears in the full version.

Secure Message Transfer. The transport security layer enables higher-level protocols to communicate across a distributed network in a way that endpoint authentication and transmission privacy is guaranteed in order for preventing eavesdropping, tampering, and message forgery. It is natural to establish these security requirements for the duration of a session where in each session the players exchange a number of messages. We capture the task by specifying a universal secure communication sessions functionality in the full version.

Description of the Secure Communication Sessions Functionality. The functionality guarantees that the adversary gains no information other than some leakage information of the message sent while intercepting the session. A leakage function $l(m)$ constraints the disclosure of side channel information and the transmitted plaintext m. In particular, the information leakage includes the length of message m and some message flow information that allow the adversary to determine the transmitted messages' source and destination. A pendant to the real world is that the adversary gains some network information about the channel from lower-layers protocols. Real Web browsers reveal several information encompassed in higher-level protocol headers (e.g. HTTP) which are necessarily sent while the server is informed to establish a secure protocol. Prominent examples are the type of browser or the referrer identity. Further, the functionality notifies the players, when a secure session is established. The technicality is necessary to convey surrounding processes the signal that they shall start the transmission.

The main difference to the secure communication session functionality proposed by Canetti [3] is that $\mathcal{F}_{\text{SCS}}^{(1,2)}$ is universal in the sense that it handles both, a uni- and bi-directional model of authentication. The model of authentication is determined by the invocation of the initiator with either its own or an anonymous identity. Further, the players learn the peer identity from the functionality's output. By contrast to related protocols (e.g. SSH), initiator and responder do not know the peer's public key in advance, but learn it in the TLS handshake. Consequently, the adversary can frame an impersonation attack against the initiator, if the responder only authenticates. In which case, the initiator is parameterized with an anonymous identity, i.e. $ID_I{=}\perp$. Then, the functionality grants the adversary the privilege to mount an impersonation query. It implies that the adversary is enabled to fix a message, unless the message has been delivered by an uncorrupted party. This provision ensures that the adversary cannot mimic other uncorrupted players.

Theorem 3. *Protocol TLS securely realizes* $\mathcal{F}_{\text{SCS}}^{(1,2)}$.

A sketch of the proof including the specification of the TLS protocol appears in the full version. A full proof appeared in [19].

3.3 Modeling User Behavior: A First Attempt

Some browser-based protocols involve the user into the protocol execution and certain assumptions must be made about the behavior. The necessity for taking into account the user behavior is due to the fact that the adversary may mount attacks that target the user. Among the user challenges of interest is the question of responder authentication. The attacker queries the user whether she interacts with an honest or a spoofed party ("phishing attacks"). Formulating a rigorous model that captures user behavior obviously raises various issues that are both technical and philosophical. For instance which human abilities can we model? Should we narrow down the user behavior to certain skills, say perceive certain objects and how do we model the quality of skills? Moreover, do users behave correctly in the sense that they always behave in the same way? The problem becomes even more elaborate when the behavior shall be quantified with respect to a security parameter. In this work we go for a simplistic approach. We wish to capture the task that a user outputs a secret stored in her memory when she recognizes a *human perceivable authenticator (HPA)*.[2] A human perceivable authenticator is some auditive or visual identifier, the user is able to detect. Her secrets are low-entropy credentials in the sense of passwords. We capture the task by formulating the human recognition functionality $\mathcal{F}_{\text{REC}}$. An illustration of this functionality is presented in the full version.

Description of the Human Recognition Functionality. The human recognition functionality $\mathcal{F}_{\text{REC}}$ maintains a database of the user secrets and authenticators taken from the domains $\mathcal{D}$ and $\mathcal{W}$, respectively. The functionality is parameterized by the credential disclosure function t. It maps a human perceivable authenticator to the user secret. Here, we make the assumption that the user outputs the same secret when she is invoked with an appropriate HPA. To quantify an appropriate authenticator, we define the indistinguishability of human perceivable authenticators.

Definition 3 (HPA Indistinguishability). *For any user* U *and* $w, w^* \in \mathcal{W}$, *we say* w *and* w^* *are perfectly human-indistinguishable ('='), if we have*

$$|\Pr[U(w,w)=1] - |\Pr[U(w,w^*)=1]| = 0,$$

and relaxed human-indistinguishable ('≈'), if there exists $\kappa_0 \in \mathbb{N}$ *and polynomial* $\tau, \mathbb{N} \to \mathbb{N}$, *such that for all* $\kappa > \kappa_0$ *we have*

$$|\Pr[U(w,w)=1] - |\Pr[U(w,w^*)=1]| < \tfrac{1}{\tau(\kappa)}.$$

[2] An interesting and for the present model relevant problem for future work would be to consider user skills to solve human puzzles (e.g. CAPTCHAs). First attempts to formalize and quantify that behavior have been proposed by Canetti, Halevi and Steiner [5].

By $\mathcal{W}^* \subseteq \mathcal{W}$ we denote the subset of all HPAs which are human-indistinguishable to some $w \in \mathcal{W}$.

The definition makes the formulation of an ideal and relaxed user model possible. The ideal user model describes a deterministic user who outputs a secret when the user is invoked with an authenticator which is identical to the authenticator stored in her memory. By contrast, the relaxed user model alleviates the behavior and handles "human imperfections". Consider the following example. Let w be an authenticator that consists of an image, and let w^* be the same image, but marginally compressed (whereby we assume that the probability to guess the authenticator grows with its size). Some users would be able to distinguish w^* and w whereas some would fail. To capture the fuzziness and quantify the human failure, the user reveals the secret in the relaxed model upon invocation with an authenticator in the range $\mathcal{W}^*$ of the authenticator stored in her memory.

Obviously, the goal is to opt for human perceivable authenticators that are appropriate for most of the users and one has to conduct user experiments determining the appropriateness. In context of browser-based protocols some usability studies have been recently inferred. They turned out that the average Internet user is not able to identify a server based on digital certificates. Unfortunately, digital certificates are of prime interest for server authentication. The user does not understand the meaning of cryptographic identifiers, but prefers to authenticate servers on the base of non-cryptographic human perceivable authenticators as she is used to do in the physical world where identities are provided in an easily recognizable fashion in the sense of brands and logos. See [7,22,13] for more discussions. Arguably, an ideal user behavior model where the user properly verifies digital server certificates, does not capture realistic assumptions and asks for a relaxation. Otherwise, it is questionable whether a security proof has much strength.[3] In order to incorporate these insights and lay a firm foundation for the analysis of browser-based protocols, we transfer the results to the presented definitions and devise an ideal and relaxed certificate user model.

Definition 4. *Let $\mathcal{F}_{\mathrm{REC}}$ be the recognition functionality and let $\mathcal{W}$ be parameterized by the set of valid digital certificates. We say a protocol π is secure in the "ideal certificate user model", if π is secure in the $\mathcal{F}_{\mathrm{REC}}$-hybrid model, assuming the ideal user model. We say a protocol π is secure in the "relaxed certificate user model", if π is secure in the $\mathcal{F}_{\mathrm{REC}}$-hybrid model, assuming the relaxed user model and $\mathcal{W}^*=\mathcal{W}$.*

The ideal certificate user model captures the fact that the user outputs a recognized query, whenever the server identity matches the identity provided by the digital certificate. (This model is identical to the ideal user model presented in [12].) By contrast, the relaxed certificate user model makes weaker assumptions about the user and her behavior. (This model is identical to the relaxed

[3] Carl Ellison coined at CRYPTO 2005 Rump Session the term ceremony to denote the paradigm that a provably secure cryptosystem becomes insecure when it is interfaced to a user [9].

user model from [10]). It says that the user accepts any certificate, since they are not relaxed human indistinguishable. In which case, user-aware protocols which convey different authenticators and build on relaxed assumptions have to be designed. This is an open research problem.

3.4 Modeling Browser Behavior: The Aggregation of Messages

On application layer, messages are composed from different communication sessions with arbitrary players and aggregated to a single document. Therefore, the browser machine extracts document objects[4] from the messages and merges them to a Web page. Today's browsers support dynamic aggregation of document objects due to the broad portfolio of scripting functionalities. These scripting functionalities permit not only access to the document object model, but also provide access to sophisticated browser functionalities. Most notably, it includes access to the browser's user and network interface, including privileges to cache, cookies, and site history, and the Document Object Model.[5] In order to capture these browser subtleties, we define a script as follows:

Definition 5. *Let $\mathcal{SCRIPT}$ denote the space of scripts. Then we say a script $\mathtt{scr} \in \mathcal{SCRIPT}$ is a sequence of scripting functionalities $\langle \mathcal{F}_1^{\mathtt{scr}}, \ldots, \mathcal{F}_n^{\mathtt{scr}} \rangle$, $n \in \mathbb{N}$. A scripting functionality $\mathcal{F}^{\mathtt{scr}}$ is a functionality in the standard UC sense and captures the tasks we expect from real-world scripts.*

In the remaining of the paper we do not distinguish between static or dynamic document objects. We use the term script to denote a static object, a dynamic object, or a set thereof. The execution of scripts is constricted to a sandbox model and the intercommunication of functionalities is constrained to the *same origin policy (SOP)* security policy. Otherwise, a functionality from a dishonest player could intercept an honest functionality and subvert their output. Informally, the SOP security policy ensures that there is no communication via scripting functionalities between objects that have different domains, protocols, or ports. A formal description of the SOP enforcing functionality $\mathcal{F}_{\mathrm{SOP}}$ appears in the full version.

Description of the SOP Access Policy Functionality. The functionality stores a table of security contexts $\mathcal{T}^{\mathtt{acc}}$ and verifies that an invoking scripting functionality gains access to another functionality $\mathcal{F}^{\mathtt{scr}}$ when it is in the same security context. The security context is defined by the SOP policy. That means that scripting functionalities have access to surrounding functionalities, when they have been invoked by the same protocol on the same port and from the same domain. Otherwise, the access is prevented and $\mathcal{F}_{\mathrm{SOP}}$ discards the access request. When $\mathcal{F}_{\mathrm{SOP}}$ grants access to some functionality $\mathcal{F}^{\mathtt{scr}}$, then the invoking functionality is able to instantiate $\mathcal{F}^{\mathtt{scr}}$ with arbitrary input. In addition, it can read, delay and write the output.

[4] One can think of HTML tags, JavaScript or Java Applet functions that invoke certain functionality when processed by the browser rendering engine.

[5] See `http://demo.nds.rub.de/dmitm` for more details.

3.5 Functional Corruption Model

The model of execution does not specify the behavior of the players upon corruption. However, the composition theorem applies to any behavior including the case that the adversary corrupts a player in spite of the fact that protocol π does not protect against compromise. In the presented model we restrict our attention to *static* corruptions, i.e. the identities of the corrupted players are known prior to the protocol execution. Adapting a stronger model where the adversary adaptively corrupts the players may be accomplished by reformulating the functionalities in the sense of relaxed UC security [6].

Typically, the adversary gains access to the internal state of the player upon corruption. Browser-based protocols mandate for a more fine-gained consideration. The adversary operates on different protocol layers and thus is feasible to exploit different functionalities. To this end, we present a natural extension of the standard corruption model for key exchange protocols to the browser-based setting. We classify corruptions according to the compromised functionality:

- **Functionality $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$.** When the adversary corrupts the secure communication sessions functionality, we distinguish between the cases:
  - *Slack Corruption.* The adversary has neither access to the long-term nor ephemeral secrets. It has only access to message m. Then, the adversary controls the functionalities which are invoked by the message. More precisely, whenever a party receives a message, it processes the message according to the protocol specification. This process may stipulate new processes, leading to the invocation of subroutine calls which are expressed in terms of functionalities here. Slack corruptions capture the fact that the adversary exploits flaws of higher-level protocols in order to compromise functionalities provided by these higher-level levels or their subroutines. (In terms of browser-based protocols, these attacks are coined cross-site-scripting, request forgery or SQL injection attacks.) In order to capture a finer-grained intuition of slack corruptions, we distinguish between
    - *Left (Initiator) Slack Corruption.* The adversary fixes the message sent to the initiator. Consequently, it has access to the browser's scripting functionalities the response messages invoke. Then, $\mathcal{F}_{\mathrm{SOP}}$ does not apply (because the adversary is in the same security context as the corrupted party). This adversary could learn sensitive information from the browser scripting functionalities.
    - *Right (Responder) Slack Corruption.* The adversary fixes the message received by the responder. It has access to the server's functionalities. This adversary could reveal secret information from the server functionalities. For instance, the adversary could compromise the password file of a database server.

    When the adversary has compromised a party in this model, the adversary gains access to the higher-layer functionalities. It then operates like a man-in-the-middle between the environment and the functionality $\mathcal{F}$

(or a set thereof) provided by the higher layer. This implies that the adversary is capable of sending the following queries:

  * $\mathsf{Read}(\mathcal{F}, \mathsf{in/out})$: The adversary reads input or output values of functionality $\mathcal{F}$ that are sent over $\mathcal{F}$'s input interfaces in (resp. output interface out). In other words, before the environment invokes a functionality $\mathcal{F}$ with some initial value, this query is first forwarded to the adversary. Similarly, before the functionality sends the output query to the environment, this message is first forwarded to the adversary.
  * $\mathsf{Write}(\mathcal{F}, \mathsf{in/out}, \mathsf{value})$: The adversary writes value on the input interface in (or output interface out) according to the specification of functionality $\mathcal{F}$. Before the environment feeds a functionality with some value, the adversary rewrites this value. When the functionality produces some output, the adversary overwrites this output with a value of his choice.
  * $\mathsf{Intercept}(\mathcal{F})$: The adversary intercepts the events triggered by functionality $\mathcal{F}$. First, the adversary is notified about the event. Then, the adversary may decide to delay the event or determine the moment of delivery the event is triggered to the environment.
  * $\mathsf{Invoke}(\mathcal{F}, \mathsf{value})$: The adversary invokes arbitrary other functionality parameterized with some initial value.the processing of message m would enable.

- *Weak Corruption.* The adversary has access to the long-term secrets and message m. This model captures PKI attacks where the adversary has convinced a trusted third party (e.g. a certification authority functionality) to bind its identity to an honest player.
- *Strong Corruption.* The adversary has access to the long-term, ephemeral secrets, and message m. This model reflects the strongest corruption model in which the adversary has access to the complete internal state.

Weak and strong corruption address the key exchange functionality which composes to the secure sessions functionality. The security goal of two party key exchange protocols against weak corruption attacks is to prevent that the adversary learns the session keys from prior sessions with the knowledge of the long-term secrets. The security goal of secure communication session protocols against slack corruptions is to prevent the adversary from learning crucial information being transferred through the functionality.

– **Functionality $\mathcal{F}_{\mathrm{IR}}$.** The adversary poisons the identity resolution and is able to fix a fake look-up repository. Consequently, it can lure the initiator to an arbitrary responder. Considering corruptions for a functionality that allows the adversary to determine the output of the resolution functionality seems to capture a trivial task at first glance. The reason is that we wish to analyze browser-based protocols where the adversary does not adaptively fake the identity resolution, but has compromised the functionality in the past. Consequently, we are not restricted to analyze protocols in front of passive adversaries under the condition that the identities are falsely resolved. We will refer to as the *subverted identity corruption model*.

– **Functionality $\mathcal{F}_{\text{REC}}$.** The adversary reveals secrets provided by the user. It gains no additional information. We will refer to as the *user corruption model*. It is important to note that this model is equivalent to the relaxed certificate user model and a strengthening of the weak corruption model, provided certificates are used. There is no need to use certified identities anymore. The user accepts any certified identity and reveals the secret. Obviously, in the user corruption model we consider a naive user. This user may be easily tricked to disclose her secrets. In particular, in the user corruption model we assume that the user is susceptible to social engineering attacks. The security goal of protocols in this model is to prevent the adversary from impersonating the user despite knowledge of her secrets.

4 Conclusion

Browser-based protocols are a prevailed class of security protocols in practice. Surprisingly, there has been less effort so far to prove security from a formal point of view. To diminish the gap, we proposed a model for the composable analysis of browser-based protocols. We elaborated basic functionalities for the composition of browser-based protocols. We also presented a functional corruption model that captures the basic threats of today's browser-based and Internet protocols. Our corruption model augments the meanwhile classical model for simulation-based key exchange from, e.g., [20] and considers corruptions of the user, who knows some low-entropy secret, corruptions of the identity resolution service which resolves domain names, and corruptions of the application server which allows to inject messages into a secure communication without revealing the session keys. An interesting challenge and issue of primary relevance for Internet applications is to prove protocols secure even though the adversary has corrupted some functionalities. The presented model lays a first foundation to formally carry out such proofs.

Another interesting topic for future work is to define variants of the secure communication sessions functionality for form-based and cookie-based authenticated channels and demonstrate that browsers securely emulate these functionalities. In the form-based model, the user enters her password into a Web form triggered by a protocol running on top of TLS. In the cookie-based model the browser appends instead of the password a cookie, i.e. a browser-specific long-term secret stored by the server in a prior session. These authentication models round off the mechanisms in today's browsers to establish a secure channel.

References

1. Barak, B., Lindell, Y., Rabin, T.: Protocol initialization for the framework of universal composability. Cryptology ePrint Archive, Report 2004/006 (2004), http://eprint.iacr.org/
2. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)

3. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society, Los Alamitos (2001)
4. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Analyzing Security Protocols Using Time-Bounded Task-PIOAs. Discrete Event Dynamic Systems 18(1), 111–159 (2008)
5. Canetti, R., Halevi, S., Steiner, M.: Mitigating dictionary attacks on password-protected local storage. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 160–179. Springer, Heidelberg (2006)
6. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
7. Dhamija, R., Tygar, J.D., Hearst, M.A.: Why phishing works. In: CHI, pp. 581–590. ACM, New York (2006)
8. Dolev, D., Yao, A.C.-C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–207 (1983)
9. Ellison, C.: Ceremony design and analysis. Cryptology ePrint Archive, Report 2007/399 (2007)
10. Gajek, S., Manulis, M., Sadeghi, A.-R., Schwenk, J.: Provably secure browser-based user-aware mutual authentication over TLS. In: ASIACCS, pp. 300–311. ACM Press, New York (2008)
11. Gross, T., Pfitzmann, B.: SAML artifact information flow revisited. In: IEEE Workshop on Web Services Security, Berkeley, USA (May 2006); Appeared also as IBM Research Report RZ 3643 (#99653) 01/03/06, IBM Research Division, Zurich (January 2006)
12. Groß, T., Pfitzmann, B., Sadeghi, A.-R.: Browser model for security analysis of browser-based protocols. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 489–508. Springer, Heidelberg (2005)
13. Herzberg, A.: Why Johnny can't surf, safely? (Work in Progress) (2007)
14. Herzberg, A., Yoffe, I.: Layered specifications, design and analysis of security protocols. Cryptology ePrint Archive, Report 2006/398 (2006)
15. Jackson, C., Barth, A., Bortz, A., Shao, W., Boneh, D.: Protecting browsers from dns rebinding attacks. In: CCS 2007, pp. 421–431. ACM, New York (2007)
16. Karlof, C., Shankar, U., Tygar, J.D., Wagner, D.: Dynamic pharming attacks and locked same-origin policies for web browsers. In: CCS 2007, pp. 58–71. ACM, New York (2007)
17. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: IEEE Symposium on Security and Privacy, pp. 184–200 (2001)
18. Pfitzmann, B., Waidner, M.: Analysis of liberty single-sign-on with enabled clients. IEEE Internet Computing 7(6), 38–44 (2003)
19. Sebastian Gajek, M.M., Pereira, O.: Universally composable security analysis of tls—secure sessions with handshake and record layer protocols. Cryptology ePrint Archive, Report 2008/251 (2008), http://eprint.iacr.org/
20. Shoup, V.: On formal models for secure key exchange (version 4). Technical report, IBM Research Report RZ 3120, November 15 (1999)
21. Soghoian, C., Jakobsson, M.: A deceit-augmented man in the middle attack against bank of america's sitekey service (2007)
22. Stuart Schechter, A.O., Dhamija, R., Fischer, I.: The emperor's new security indicators. In: Symposium on Security and Privacy, pp. 51–65. IEEE Computer Society, Los Alamitos (2007)

Threshold Homomorphic Encryption in the Universally Composable Cryptographic Library

Peeter Laud[1,2,⋆] and Long Ngo[1]

[1] Tartu University
[2] Cybernetica AS
peeter.laud@ut.ee, ngothanglong@yahoo.com

Abstract. The *universally composable cryptographic library* by Backes, Pfitzmann and Waidner provides Dolev-Yao-like, but cryptographically sound abstractions to common cryptographic primitives like encryptions and signatures. The library has been used to give the correctness proofs of various protocols; while the arguments in such proofs are similar to the ones done with the Dolev-Yao model that has been researched for a couple of decades already, the conclusions that such arguments provide are cryptographically sound.

Various interesting protocols, for example e-voting, make extensive use of primitives that the library currently does not provide. The library can certainly be extended, and in this paper we provide one such extension — we add threshold homomorphic encryption to the universally composable cryptographic library and demonstrate its usefulness by (re)proving the security of a well-known e-voting protocol.

1 Introduction

Cryptographic protocol verification is an error-prone task. A tractable way of doing it usually involves employing some abstraction of cryptographic operations, for example using the Dolev-Yao model [23]. In this model, messages are modeled as terms over a certain algebra, possibly with some cancellation rules, and possible operations are defined over the structure of those terms. This approach makes it simple to use formal methods to analyse the protocol, but the question of soundness of the abstraction has not been satisfactorily solved yet. On the other hand, computational methods can produce computationally sound proofs, but are complex and error-prone.

There exists a sound abstraction of cryptographic operations — the *universally composable cryptographic library* [7,11,12] — that has the abstraction level comparable to the Dolev-Yao model. The first version of this library contained signature and public-key encryption schemes. Later, the library has been extended to some more primitives common in the Dolev-Yao model, and shown that it can not unconditionally have some special primitives.

⋆ Supported by Estonian Science Foundation, grant #6944, and European Union through the European Regional Development Fund and the 6th Framework Programme project AEOLUS (FP6-IST-15964).

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 298–312, 2008.

In this paper we extend the library to have threshold homomorphic encryption. The extension involves adding some new commands to the library, while maintaining its abstraction level. We show that the extended library is still computationally sound. It is suitable for the analysis of important classes of protocols, for example electronic voting, auctions or lotteries. A separate contribution of this paper is also the actual introduction of a possible Dolev-Yao style abstraction for threshold homomorphic encryption.

2 Related Work

The model of *universal composability* alias *reactive simulatability* was proposed by Canetti [20] and by Pfitzmann et al [13,34]. The model has been used to define sound abstractions of various cryptographic primitives. Among the most celebrated abstractions is the universally composable cryptographic library [7,11,12] providing Dolev-Yao-style abstractions for common cryptographic primitives, namely symmetric and asymmetric encryption, signatures, MACs, nonces. The library has been used in the security proofs for some protocols: Needham-Schroeder-Lowe protocol (asymmetric) [6], the Otway-Rees protocol (symmetric) [2], the strengthened Yahalom protocol (real secrecy) [10], a payment system [3]. There have also been approaches in using the library to construct secure systems [1,5,30,35]. It is also known that certain primitives cannot be reasonably abstracted by such a library [8,14]. Recently, the notion of *Conditional Reactive Simulatability* [4] has been proposed, providing soundness only for a certain class of users and potentially allowing to abstract more primitives.

Threshold homomorphic encryption [22,33] is one of the most versatile cryptographic primitives, combining the distribution of trust with the ability to combine plaintexts under encryption. An overview of using this primitive in e-voting is given in [32]. Such e-voting protocols have been proved to be universally composable [27]. In these proofs, the security requirement put on the threshold homomorphic encryption primitive has basically been security under chosen plaintext attacks (IND-TCPA) [24,25]. We are not aware of any attempts to abstract this primitive in the Dolev-Yao-style.

3 UC Cryptographic Library

In the treatment of reactive simulatability [34] the *systems* are modeled as sets of *structures*. A structure *Str* is a collection of probabilistic interactive Turing machines. Each of the machines has a number of input and output *ports*; an input and output port with the same name (which must not repeat) form a secure communication channel between corresponding machines. Authentic or insecure channels are modeled using secure channels. A port in the structure may also be unconnected; a certain subset S of such ports are called the *free ports*. The structure provides the intended service over those ports. The rest of unconnected ports represent the possible weaknesses of the structure; the adversary will connect to those ports. A *configuration* of (Str, S) is a *closed collection* (i.e. no

unconnected ports), consisting of the machines in Str, a machine H representing the users of the service (called the *environment* in [20]) and connecting only to the free ports of the structure, and an adversarial machine A. There may also be connection(s) between H and A. The *view* of the user H in some configuration C, denoted $view_C(\mathsf{H})$, is the distribution of the sequence of messages on the ports of H.

Given two structures Str and Str' with the same set S of free ports, we say that Str is *at least as secure as* Str' if for all H and A there exists an adversary S, such that $view_{Str\|\mathsf{H}\|\mathsf{A}}(\mathsf{H}) \approx view_{Str'\|\mathsf{H}\|\mathsf{S}}(\mathsf{H})$, where $\approx$ denotes computational indistinguishability [26]. The simulatability is *black-box* if there exists a single machine Sim, called the *simulator*, such that $\mathsf{Sim}\|\mathsf{A}$ is a suitable choice for S, for all H and A. The *at least as secure as*-relation is lifted to systems in the natural way. The central result of the theory of UC is the *composition theorem*. It states that if we replace a substructure of some structure with something that is at least as secure, then the entire resulting structure is also at least as secure as the original one.

When modeling and analysing systems, one speaks about real and ideal structures. The real structure reflects the distribution of components in the real world, with each participant typically having one or several machines implementing the cryptographic operations and protocol logic. A typical ideal structure consists of a single machine that "obviously" satisfies the security requirements we have put on the system. A well-designed ideal structure also has a simple internal state and does not use hard-to-analyse operations (e.g. random number generation). One has to show that the real structure is at least as secure as the ideal structure. While analysing a system, one may locate the real structures it is using, replace them with the corresponding ideal structures (using the composition theorem) and analyse the resulting ideal system instead.

The *simulatable cryptographic library* [12] is such a (set of) pair(s) of real and ideal structures. The ideal structure quite precisely imitates the Dolev-Yao terms used to abstract the cryptographic messages. The main part of the ideal machine $\mathcal{TH}_n$ for n participants is the database of terms. For each term that has ever been created by one of the participants or the adversary, it records its outermost constructor and immediate subterms. The library also records which parties know which term; if a party knows a term then it has a *handle* to it. These handles are generated as needed and are themselves devoid of information (they are just consecutive integers). Hence all message transmissions have to happen through the library, which has to translate the handles. For a term t, let $t^{\mathsf{hnd}_\mathsf{u}}$ be its handle for the user u; u may be omitted if it is clear from the context. All parties can store "raw" bit-strings in the database (called the payloads) and $\mathsf{retrieve}$ their contents, construct new terms and decompose them. The rules for the possibility of composing and decomposing terms are very similar to the rules in the Dolev-Yao model. The adversary has some extra commands in its disposal, for example, creating garbage terms or invalid ciphertexts. The machine $\mathcal{TH}_n$ has the input port $\mathsf{in}_{\mathsf{u}_i}?$ and the output port $\mathsf{out}_{\mathsf{u}_i}!$ for communicating with the i-th user and the ports $\mathsf{in}_\mathsf{a}?$ and $\mathsf{out}_\mathsf{a}!$ for communicating with the adversary. In

the real structure, the users still access the terms through the handles (because the real and the ideal interfaces must be the same), but there is a machine M_i for each participant P_i. Bit-strings are used to represent cryptographic messages; the machines use them to communicate with each other (and with the adversary). The cryptographic operations are implemented using conventional primitives. Each secure channel between two parties is modeled by one, each authentic or insecure channel by two (from the transmitter to the adversary, and from the transmitter or the adversary to the receiver) communication channels.

In the model of asynchronous relative simulatability [34], the machines themselves are in charge of scheduling. The scheduling is channel-based, each channel is scheduled by a certain machine. Whenever a machine finishes its step and stores the newly generated messages in the buffers of channels it has output ports for, it may also *clock* at most one of the channels it schedules. The first message in the buffer of that channel is then delivered to its recipient and this machine is the next to run. If this buffer is empty or if no channel was clocked then the control passes to a designated machine (usually the adversary) called the *master scheduler*. Such clocking mechanism is very versatile and allows one to model both network delays (channel is clocked by the adversary) and API calls (there is a channel in each direction between two machines, clocked by their transmitters and scheduled each time they are written to). The commands from the user H to $\mathcal{TH}_n$ / M_i are made through API calls. The machine $\mathcal{TH}_n$ also communicates with the adversary using API calls. In the real structure, the machine M_i clocks the channels from itself to the adversary, while the channels from the adversary or between two machines are clocked by the adversary.

More details on the asynchronous relative simulatability and the UC cryptographic library can be found in [34,12], as well as in the full version of this paper [31].

4 Adding Threshold Homomorphic Encryption

4.1 The Cryptographic Primitives

A (t, w)-*threshold* $(\boxdot, \boxplus)$-*homomorphic encryption primitive* is a tuple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{Z}, \mathcal{V}, \mathcal{C})$ where the *key generation* algorithm $\mathcal{K}$ returns a new public key pk, secret keys $sk_1, \ldots, sk_w$ and (public) verification keys $vk_1, \ldots, vk_w$ at each invocation; the encryption $\mathcal{E}_{pk}(m, r)$ returns the encryption of the message m under the public key pk with the random coins r; the decryption $\mathcal{D}_{sk_i}(c)$ returns the *i-th decryption share* ds_i of the ciphertext c, the correctness of decryption can be verified by invoking $\mathcal{V}_{vk_i}(ds_i, p_i, c)$ where $p_i = \mathcal{Z}_{sk_i}(c)$; the *share combination* algorithm $\mathcal{C}$ takes any t decryption shares $ds_{i_1}, \ldots, ds_{i_t}$ and combines them into the plaintext m. For any $(pk, sk_1, \ldots, sk_w, vk_1, \ldots, vk_w)$ possibly returned by $\mathcal{K}$, the algorithms must satisfy the following conditions [27]. The cryptosystem of [22] can be used here.

- Correctness: If $c = \mathcal{E}_{pk}(m, r)$ and $ds_i = \mathcal{D}_{sk_i}(c)$ then $\mathcal{C}(ds_{i_1}, \ldots, ds_{i_t}) = m$.
- Homomorphism: Let $c_i = \mathcal{E}_{pk}(m_i, r_i)$. Then $c_1 \boxdot c_2$ is a valid ciphertext corresponding to the plaintext $m_1 \boxplus m_2$.

- Correct decryption: Let $c = \mathcal{E}_{pk}(m, r)$, $ds_i = \mathcal{D}_{sk_i}(c)$ and $p_i = \mathcal{Z}_{sk_i}(c)$. Then $\mathcal{V}_{vk_i}(ds_i, p_i, c) = \mathsf{true}$.
- Simulatability: there exists and algorithm $\mathcal{S}$ taking as inputs any m, c, and $ds_{i_1}, \ldots, ds_{i_{t'-1}}$ ($t' \leq t$) and returning the (simulated) decryption shares for the rest of the authorities, such that any t of the shares will be combined to m and the simulated shares are indistinguishable from the real shares even to someone with the knowledge of $sk_1, \ldots, sk_{i_{t'-1}}$.
- IND-CPA-security, even if the adversary has learned up to $t-1$ secret keys.

To ease the presentation, we will in the following assume that $vk_i = (pk, i)$. I.e. the public key includes the verification keys.

A *non-interactive zero-knowledge (NIZK) proof* is a message, constructed by a party (the prover) that convinces any other party that the prover knows the witness for the membership of a certain bit-string in a certain language, without leaking any other information.

4.2 Ideal Library

We extend the machine $\mathcal{TH}_n$ to accommodate the new primitive. The extension involves introducing new message constructors for the kinds of data created by the new primitive, as well as commands for generating keys, encrypting, and decrypting messages, verifying and combining shares and performing the homomorphic operations. Foreseeing the application of the extended library in the analyses of various protocols, where the participants must show that the plaintext in the ciphertexts they have produced comes from a restricted set, we parameterize the library with a predicate $\mathbf{V}$ over bit-strings, giving their *validity*. The library allows one to encrypt *only the payloads*, because it is far from clear what the $\boxplus$-combination of non-payload terms should be. The extension adds several new commands and term constructors to $\mathcal{TH}_n$.

To initiate the generation of a new key, a party u (or the adversary; all commands available to a party are also available to the adversary) invokes the command $\mathsf{gen\_enc\_thres\_keylist}(a_1, \ldots, a_w)$, where a_i indicates who receives the i-th share of the secret key (its either a user or the adversary with the adversary receiving at most $t-1$ shares). Upon receiving that command, $\mathcal{TH}_n$ adds to the database a new public key pk (constructor thpk, no arguments) and secret key shares sk_i (constructor thsk, arguments pk, i, a_i) for $1 \leq i \leq w$. It sends *to the adversary* the command $\mathsf{keylist\_notify}(pk^{\mathsf{hnd}}, \mathsf{u}, a_1, \ldots, a_w, sk_{i_1}^{\mathsf{hnd}}, \ldots, sk_{i_k}^{\mathsf{hnd}})$ where $sk_{i_1}, \ldots, sk_{i_k}$ are the secret key shares intended for the adversary. As this command abstracts a certain multiparty computation protocol, the adversary controls when a user learns the key shares intended for it. The adversary may later send a command $\mathsf{adv\_learn\_share}(pk^{\mathsf{hnd}}, j)$, causing $\mathcal{TH}_n$ to send $\mathsf{learn\_share}(pk^{\mathsf{hnd}}, a_1, \ldots, a_w, sk_{i_1}^{\mathsf{hnd}}, \ldots, sk_{i_k}^{\mathsf{hnd}})$ the user u_j, where $sk_{i_1}, \ldots, sk_{i_k}$ are intended for it. The adversary can also generate *invalid keys* by invoking the command $\mathsf{adv\_gen\_key}()$. This causes $\mathcal{TH}_n$ to generate just a single new term for a public key (constructor thpk) and return its handle to the adversary.

The encryption is straightforward: a command $\mathsf{encth}(pk^{\mathsf{hnd}}, m^{\mathsf{hnd}})$ causes $\mathcal{TH}_n$ to create a new term c with the constructor thciph and the arguments pk and

m. But $\mathcal{TH}_n$ verifies before that pk is a public key, m is a payload, and $\mathbf{V}(m)$ holds. If the verification is unsuccessful, an error is returned. In addition to c, $\mathcal{TH}_n$ also creates a term $p = \mathsf{nizkv}(c)$ embodying the NIZK proof of correctness of the validity of m. The handles of both c and p are returned. The adversary can also generate an invalid ciphertext or proof — the commands $\mathsf{adv\_invenc}(pk^{\mathsf{hnd}}, l)$ and $\mathsf{adv\_invproof}(pk^{\mathsf{hnd}}, l)$ return handles to terms c and p, respectively, where $c = \mathsf{thciph}(pk, l)$ and $p = \mathsf{nizkv}(l)$. Here l is the intended length of the plaintext. The plaintext itself does not have to be present. Similarly to [12], it is possible to find the public key from a ciphertext using the command keyofth, and it is impossible to use the secret key shares for anything else than decryption.

The decryption command is more complex — it is $\mathsf{decth}(sk^{\mathsf{hnd}}, c_1^{\mathsf{hnd}}, p_1^{\mathsf{hnd}}, \ldots, c_k^{\mathsf{hnd}}, p_k^{\mathsf{hnd}})$, where $c_1, \ldots, c_k$ are ciphertexts and $p_1, \ldots, p_k$ are NIZK proofs. $\mathcal{TH}_n$ verifies that all ciphertexts are created with the public key pk, where sk is the term $\mathsf{thsk}(pk, j, \_)$. $\mathcal{TH}_n$ also verifies that $p_i = \mathsf{nizkv}(c_i)$ (for all i). If some p_i was an invalid proof (had only a length argument) then $\mathcal{TH}_n$ sends $\mathsf{adv\_findwit}(c_i^{\mathsf{hnd_a}}, p_i^{\mathsf{hnd_a}})$ to the adversary and expects to receive $\mathsf{adv\_foundwit}(c_i^{\mathsf{hnd_a}}, p_i^{\mathsf{hnd_a}}, m_i^{\mathsf{hnd_a}})$. If m_i is the plaintext of c_i then $\mathcal{TH}_n$ changes p_i into $\mathsf{nizkv}(c_i)$ and accepts it. While constructing the $\mathsf{adv\_foundwit}(\ldots)$-answer, the adversary is allowed to parse the terms and store new payloads, but not communicate with H. If the checks succeed then $\mathcal{TH}_n$ creates a new payload term d whose payload is the $\boxplus$-combination of the plaintexts of $c_1, \ldots, c_k$. It also creates new terms ds (plaintext share; constructor thshare, arguments $d, j, c_1, \ldots, c_k$) and dp (proof of correctness of decryption, constructor sharepr, argument ds) and returns the handles to the last two terms. The adversary can also use the command adv_decth to decrypt; it takes the same arguments, except for NIZK proofs of plaintext validity, and returns a plaintext share and proof of correctness of decryption. The adversary can also construct an *invalid share* by invoking $\mathsf{adv\_invshare}(l, j, c_1^{\mathsf{hnd}}, \ldots, c_k^{\mathsf{hnd}})$, where l is the length of the plaintext, j is the position of the plaintext share and $c_1, \ldots, c_k$ are the ciphertexts from whose combination the plaintext share has been apparently obtained. This command verifies that $c_1, \ldots, c_k$ have the same public key, adds a single new term (constructor thshare, arguments $\perp, j, c_1, \ldots, c_k$) to the database and returns the handle to it. An *invalid proof of correctness of decryption* can also be created by the adversary by invoking $\mathsf{adv\_invdp}()$; it creates a new sharepr-term without arguments. A valid proof can be *transformed* by invoking $\mathsf{adv\_transdp}(dp^{\mathsf{hnd}})$; it creates a copy of the term dp and returns a handle to it.

Finally, there is the command to combine plaintext shares: when receiving $\mathsf{combine}(ds_1^{\mathsf{hnd}}, dp_1^{\mathsf{hnd}}, \ldots, ds_t^{\mathsf{hnd}}, dp_t^{\mathsf{hnd}}, pk^{\mathsf{hnd}})$, $\mathcal{TH}_n$ checks that the decryption shares correspond to the same set of ciphertexts, that they are different, and that the public key is the one that was used to create the ciphertexts. If these checks pass then there are two options. If pk was created by the command gen_enc_thres_keylist then $\mathcal{TH}_n$ checks that all proofs of correctness of decryption point to their respective decryption shares. If all checks pass, then a handle to the payload d referenced by all ds_i is returned. If pk was created by the command adv_gen_key then $\mathcal{TH}_n$ forwards the combine-command to the adversary

(translating the handles in the process) and forwards its answer (which must be a handle to a payload, or $\perp$) back to the user.

The adversary can also invoke a command $\mathsf{adv\_parse}(t^{\mathsf{hnd}})$ for any term for which it has a handle. In most cases, $\mathcal{TH}_n$ answers with the type of t, as well as with t's arguments (the subterms are translated into handles). Only if t is a ciphertext and the adversary does not know enough plaintext and key shares to decrypt, is the adversary given no handle to the plaintext, but is given only the length of the plaintext. Similarly, if t is a plaintext share (its arguments are the plaintext and the ciphertexts whose combination is decrypted) and the adversary is unable to find the plaintext (for the same reasons as above), is the plaintext omitted from the answer of $\mathcal{TH}_n$.

Remark. We assume that while the ideal adversary processes an $\mathsf{adv\_findwit}$-command, its behavior is somehow constrained. Such assumptions on the ideal-process adversary are relatively wide-spread, but little-researched. They appeared already in the original report introducing universal composability [19], where the ideal signature functionality $\mathcal{F}_{\mathrm{SIG}}$ assumed the adversary to return a bit-string representing the signature when asked so. The rationale of putting such restrictions on the ideal adversary are twofold. First, they make the ideal functionality more secure, and second, this restricted class of ideal adversaries is large enough to take into account all possible real adversaries — the composition of the simulator and the real adversary will always belong to this restricted class. We will see more examples of such constraints in Sec. 4.3.

4.3 Real (or Hybrid) Library

The real library for n participants consists mainly of n machines $M_1, \ldots, M_n$ where M_i, having the ports $\mathsf{in}_{\mathsf{u}_i}?$ and $\mathsf{out}_{\mathsf{u}_i}!$ handles the cryptographic tasks for the i-th participant. The machines M_i work as in [12], but they also have to be extended to cope with the new commands. Additionally, we will use certain ideal functionalities for some tasks. These functionalities also have universally composable implementations, with the help of the composition theorem we will get the entire implementation of the real library (in the *common reference string* model [21]).

We make use of the NIZK functionality $\mathcal{F}^R_{\mathrm{NIZK}}$ [28], where R is the *witness relation*. It works as follows. On input $\mathsf{prove}(x, w)$ from some party (including the adversary) it first verifies whether $(x, w) \in R$. If not then it ignores the input. Otherwise it sends $\mathsf{proof}(x)$ to the adversary and expects it to return some bit-string π. $\mathcal{F}^R_{\mathrm{NIZK}}$ stores (x, π) and returns π (representing the proof) to the querying party. To verify a proof, a party submits $\mathsf{verify}?(x, \pi)$ to $\mathcal{F}^R_{\mathrm{NIZK}}$. If (x, π) has been stored, it returns "yes". Otherwise $\mathcal{F}^R_{\mathrm{NIZK}}$ sends $\mathsf{witness}(x, \pi)$ to the adversary and expects it to return some witness w. If $(x, w) \in R$ then $\mathcal{F}^R_{\mathrm{NIZK}}$ stores (x, π) and returns "yes", otherwise it returns "no". While answering to the queries from $\mathcal{F}^R_{\mathrm{NIZK}}$, the adversary *is not allowed to change its state or communicate with other machines*. In other words, the adversary will not remember that it has answered those queries. The simulator given in [28] satisfies this property.

Our real structure contains two machines realizing $\mathcal{F}_{\text{NIZK}}$, with different witness relations. $\mathcal{F}^1_{\text{NIZK}}$ is used to give validity proofs of ciphertexts; its witnessing relation is $R_1 = \{((pk, c), (m, r)) \mid c = \mathcal{E}_{\text{pk}}(m, r) \wedge \mathbf{V}(m)\}$. The machine $\mathcal{F}^2_{\text{NIZK}}$ is used to construct the correctness proofs for decryption; its witnessing relation is $R_2 = \{((ds, c, pk, j), p) \mid \mathcal{V}_{(pk,j)}(ds, p, c) = \mathsf{true}\}$. Those machines have connections to and from the machines $M_1, \ldots, M_n$, as well as the adversary. All communication over those connections is through API calls (subroutine-style), i.e. the sender on a channel also clocks that channel.

Our real structure also contains a machine $\mathcal{F}_{\text{KEY}}$ serving as the ideal functionality for distributed key generation. It also has connections to and from the machines $M_1, \ldots, M_n$ and the adversary. The connections from M_i and between $\mathcal{F}_{\text{KEY}}$ and the adversary are clocked subroutine-style. However, as $\mathcal{F}_{\text{KEY}}$ represents a distributed protocol, the connections from it to machines M_i are clocked by the adversary. $\mathcal{F}_{\text{KEY}}$ accepts a single command $\mathsf{keygen}(a_1, \ldots, a_w)$ from one of the machines $M_1, \ldots, M_n$ or the adversary. Here $a_1, \ldots, a_w$ have the same meaning as by $\mathsf{gen\_enc\_thres\_keylist}$. It responds by generating a set of keys pk, $sk_1, \ldots, sk_w$ and sending to the adversary and all parties mentioned among $a_1, \ldots, a_w$ the public key and all secret key shares intended for this party. Protocols implementing $\mathcal{F}_{\text{KEY}}$ are given e.g. in [37].

Recall that the state of the machines M_i mainly consisted of a dictionary that mapped handles of messages to bit-strings; we assume that the type of each message can be uniquely determined from the bit-string representing it. Let us now describe how M_i processes commands from H. The key-generation command $\mathsf{gen\_enc\_thres\_keylist}(a_1, \ldots, a_w)$ is forwarded to $\mathcal{F}_{\text{KEY}}$ as $\mathsf{keygen}(a_1, \ldots, a_w)$. If some answer is received from $\mathcal{F}_{\text{KEY}}$ (recall that this answer is scheduled by the adversary) then the received public key and secret key shares are stored together with new handles generated for them (we assume that each secret key share includes its position). The handles are also sent to the user as arguments of the command $\mathsf{learn\_share}$.

The command $\mathsf{encth}(pk^{\mathsf{hnd}}, m^{\mathsf{hnd}})$ is realized by performing the same checks as the ideal library, generating random coins r, calling $c^* \leftarrow \mathcal{E}_{pk}(m)$, submitting (pk, c^*) together with the witness (m, r) to $\mathcal{F}^1_{\text{NIZK}}$, getting back p^*, generating new handles c^{hnd} and p^{hnd}, and storing $c^{\mathsf{hnd}} \mapsto (c^*, pk)$ and $p^{\mathsf{hnd}} \mapsto (p^*, c^*)$. Finally, M_i returns c^{hnd} and p^{hnd}. Note that the NIZK proof includes the ciphertext. The command $\mathsf{keyofth}$ is straightforward to implement.

Decryption $\mathsf{decth}(sk^{\mathsf{hnd}}, c_1^{\mathsf{hnd}}, p_1^{\mathsf{hnd}}, \ldots, c_k^{\mathsf{hnd}}, p_k^{\mathsf{hnd}})$ is done by checking all the proofs p_i with the help of $\mathcal{F}^1_{\text{NIZK}}$, combining the ciphertexts as $c = c_1 \boxdot \cdots \boxdot c_k$, decrypting c as $ds^* = \mathcal{D}_{sk}(c)$, finding the proof of correctness by $dp_\circ = \mathcal{Z}_{sk}(c)$, turning it into a NIZK proof of correctness by submitting $(ds^*, c, (pk, j))$ with the witness $dp_\circ$ to $\mathcal{F}^2_{\text{NIZK}}$ and getting back dp^* (here j is the position of sk among the secret key shares; M_i has stored it alongside sk), generating new handles ds^{hnd} and dp^{hnd}, and storing $ds^{\mathsf{hnd}} \mapsto (ds^*, j, c_1, \ldots, c_k)$ and $dp^{\mathsf{hnd}} \mapsto (dp^*, ds^*, j, c_1, \ldots, c_k)$. Here j is the position of the secret key share sk (stored together with it). The newly generated handles are returned. Again note that the proof contains its subject plaintext share which in turn contains

the ciphertexts it was generated from. Finally combine (whose argument was a list of handles to plaintext shares, correctness proofs of decryption, and the public key) is implemented by verifying all the proofs with the help of $\mathcal{F}^2_{\mathrm{NIZK}}$ and combining the shares using the algorithm $\mathcal{C}$.

Theorem 1. *The real structure consisting of machines $M_1, \ldots, M_n$, $\mathcal{F}^1_{\mathrm{NIZK}}$, $\mathcal{F}^2_{\mathrm{NIZK}}$, $\mathcal{F}_{\mathrm{KEY}}$ is at least as secure (in the black-box sense) as the ideal structure consisting of the machine $\mathcal{TH}_n$.*

5 The Simulator

Theorem 1 is proved by constructing a suitable simulator *Sim*. The main task of the simulator is to translate between the views of the real and the ideal adversary. Whenever a message is received from $\mathcal{TH}_n$, the simulator has to assign a bit-string to it and forward it to the real adversary. Whenever a message is received from the real adversary, the simulator has to parse that message and enter it into $\mathcal{TH}_n$, receiving a handle for it in the process. Additionally, the scheduling decisions have to be translated. On the one hand, the simulator *Sim* has the ports $\mathsf{in_a}!$ and $\mathsf{out_a}?$ to communicate with $\mathcal{TH}_n$ (the simulator also clocks the channel $\mathsf{in_a}$). On the other hand, it has all the ports for the real adversary, such that it can play the machines $M_1, \ldots, M_n$, $\mathcal{F}^i_{\mathrm{NIZK}}$ and $\mathcal{F}_{\mathrm{KEY}}$ to it. The simulator can be thought of as containing the copies of those machines, although it is possible to intervene with their normal operation. In principle, all channels between those machines also exist, even though both their input and output ports belong to *Sim*. If the channel is also clocked by *Sim*, then one does not have to consider this channel. But there are also some channels from *Sim* to *Sim* (originally from $\mathcal{F}_{\mathrm{KEY}}$ to M_i) that are scheduled by the adversary.

The full description of the simulator, as well as its correctness proof is given in [31]. Here we will only describe some more interesting aspects of its work. The main part of the state of the simulator is a database, similar to the machines M_i. It stores the handles of the messages (coinciding with the handles assigned to terms by $\mathcal{TH}_n$) together with the bit-string representation of those messages. There may be some additional arguments associated with each entry. The state of the simulator also includes the states of the "embedded" machines $\mathcal{F}^i_{\mathrm{NIZK}}$.

To translate the handles received from $\mathcal{TH}_n$ to bit-strings given to the real adversary, parse the term corresponding to the received handle, generate new keys, ciphertexts, etc. for all terms that the simulator has not seen before, use the saved bit-string representations for terms already seen, and combine everything together using the cryptographic operation corresponding to the constructor of the term. The simulator may have difficulties if $\mathcal{TH}_n$ does not allow it to parse a certain term. If this term was a ciphertext (and the simulator does not have access to sufficiently many secret key shares to decrypt it) then translate it by generating a random ciphertext and let the embedded $\mathcal{F}^1_{\mathrm{NIZK}}$ give a validity proof for it. If the untranslatable term was a decryption share then construct a bit-string corresponding to it by invoking the share simulation algorithm $\mathcal{S}$ (if the simulator can obtain a handle to the plaintext term) or by generating

a random bit-string (otherwise). The matching proof of validity is given by the *embedded* $\mathcal{F}^2_{\text{NIZK}}$, whose operation is modified to *not require a witness*.

The translation of bit-strings received from the real adversary to terms entered into $\mathcal{TH}_n$ is similar — parse the bit-string as much as possible, using the information already available to the simulator, enter the subterms into $\mathcal{TH}_n$ and finally use a message constructor operation to create the term corresponding to the entire bit-string (or some other command available to honest users to obtain a handle to an already existing term). If the simulator cannot fully parse the received bit-string, then one of the adversarial commands of $\mathcal{TH}_n$ has to be used to construct a suitable term; the set of adversarial commands given in Sec. 4.2 is sufficient for all cases. A bit-string representing a public key that the simulator has not yet seen is entered into $\mathcal{TH}_n$ by using the command adv_gen_key. A ciphertext encrypted with such a key is entered with the help of the command adv_invenc. This command also returns the handle for a suitable proof of plaintext validity. If another proof for the same ciphertext is received from the real adversary then adv_invproof is used to create a handle corresponding to it. The received decryption shares are treated similarly — if the adversary does not have a handle to the necessary secret key share then the commands for creating invalid shares and/or validity proofs are used. Note that the choice between transforming and existing proof (command adv_transdp) and generating an invalid proof (command adv_invdp) depends on whether the corresponding decryption share already has a validity proof in the database of $\mathcal{TH}_n$.

Note that by containing a copy of $\mathcal{F}_{\text{KEY}}$, the simulator knows the secret key shares for all key generations initiated by honest participants (through $\mathcal{TH}_n$), as well as by the real adversary, if it chose to use the functionality $\mathcal{F}_{\text{KEY}}$ for it. The commands adv_gen_key and adv_invproof are only necessary if the real adversary has generated the keys without any help from the simulator.

6 Example: A Simple e-Voting System

To demonstrate the usefulness of our extension, we construct a simple e-voting system based on it and prove that it satisfies certain security properties. The system runs with n voters and w authorities. The functionality of the i-th voter is implemented by the machine $\mathsf{M^V}_i$ and the functionality of the j-th authority by the machine $\mathsf{M^{AU}}_j$. These machines can be seen as parts of the honest user H of $\mathcal{TH}_{n+w}$. They receive commands (to vote in a particular way, to start tallying) from the rest of the honest user, implement the voting protocol, and use our library to implement the cryptography and networking. In other words, they are the *protocol machines* of [30]. Any number of machines $\mathsf{M^V}_i$ and at most $(t-1)$ of the machines $\mathsf{M^{AU}}_j$ may be under adversarial control (only static corruptions are allowed).

Later we will recall a number of security requirements for voting systems and show that this system (using the ideal library) meets these requirements. By the composition theorem, the security of the e-voting system is still preserved when we replace the ideal library with the real one.

$vote^{\mathsf{hnd}} \leftarrow \mathsf{store}(v)$
$(c^{\mathsf{hnd}}, p^{\mathsf{hnd}}) \leftarrow \mathsf{encth}(pk^{\mathsf{hnd}}, vote^{\mathsf{hnd}})$
$s^{\mathsf{hnd}} \leftarrow \mathsf{sign}(k_i^{\mathsf{s,hnd}}, c^{\mathsf{hnd}})$
$l^{\mathsf{hnd}} \leftarrow \mathsf{list}(s^{\mathsf{hnd}}, p^{\mathsf{hnd}})$
for all $i \in \{1, ..., w\}$ **do**
  $\mathsf{send\_i}(\mathsf{AU}_i, l^{\mathsf{hnd}})$
end for

$s^{\mathsf{hnd}} \leftarrow \mathsf{list\_proj}(l^{\mathsf{hnd}}, 1)$
$p^{\mathsf{hnd}} \leftarrow \mathsf{list\_proj}(l^{\mathsf{hnd}}, 2)$
$c^{\mathsf{hnd}} \leftarrow \mathsf{msg\_of\_sig}(s^{\mathsf{hnd}})$
if $\mathsf{verify}(s^{\mathsf{hnd}}, k_i^{\mathsf{v,hnd}}, c^{\mathsf{hnd}})$
  and $(c^{\mathsf{hnd}}, p^{\mathsf{hnd}}) \notin image(S)$ **then**
  $S \leftarrow S \cup \{i \mapsto (c^{\mathsf{hnd}}, p^{\mathsf{hnd}})\}$
    /* Initially, S is empty */
end if

Fig. 1. Algorithms for sending and receiving a vote

We put an additional condition on the adversary: when interfering with the communication from voters to authorities, it treats all authorities equally. I.e., it may block a voter transmitting its vote to the authorities, but it may not allow the vote to reach some authorities and not reach the others. If the adversary changes the message sent from a voter to the authorities, all adversaries will still receive the same message. This restriction models the bulletin board that is typically used for the voters to publish their (encrypted) votes [27]. We also assume that the user(s) of $\mathsf{M^V}_i$ and $\mathsf{M^{AU}}_j$ make sure that different phases of voting (key distribution, voting, tallying) start and end at the same time for different machines.

Initialization. Each machine $\mathsf{M^V}_i$ generates a signing and verification key pair $(k_i^{\mathsf{s}}, k_i^{\mathsf{v}})$ using the command $\mathsf{gen\_sig\_keypair}$ [12] and sends it to all other parties *over the authentic channel*. Some party (or the adversary) invokes $\mathsf{gen\_enc\_thres\_keylist}(\mathsf{AU}_1, \ldots, \mathsf{AU}_w)$. The authorities will learn their respective secret key shares $sk_1, \ldots, sk_w$ and the encryption key pk. The public key is also transmitted to the voters in an authentic manner. This might be realized by having each authority send pk to each voter over an authentic channel and let the voter accept if it has received the same pk at least t times.

Voting. Fig. 1 (left) describes the actions of $\mathsf{M^V}_i$ upon receiving the command $\mathsf{vote}(v)$ from the user for the first time (each subsequent time, the command is ignored). For sending messages to multiple receivers, one has to handle the scheduling [34], but we will omit the details here. Fig. 1 (right) describes the actions of $\mathsf{M^{AU}}_j$ upon receiving a vote l^{hnd}, apparently from the voter $\mathsf{M^V}_i$. After $\mathsf{M^{AU}}_j$ has successfully received the vote of $\mathsf{M^V}_i$, it ignores the subsequent attempts to send it.

Tallying. Fig. 2 (left) describes the actions of $\mathsf{M^{AU}}_j$ after receiving the command to count the votes and publish a share of the final result. As usual, the final result is presumed to be the $\boxplus$-combination of the votes. Fig. 2 (right) describes how the votes are combined in any machine.

We see that the system we have thus defined can only be used for a single voting. It would be straightforward to modify it for several elections, by adding a *session identifier* to each command. This session identifier must then be bound to the messages the parties send to each other, requiring the authorities to also

$(ds^{\mathsf{hnd}}, dp^{\mathsf{hnd}}) \leftarrow$
  $\mathsf{decth}(sk^{\mathsf{hnd}}, S(1), \ldots, S(n))$
$l^{\mathsf{hnd}} \leftarrow \mathsf{list}(ds^{\mathsf{hnd}}, dp^{\mathsf{hnd}})$
for all $i \in \{1, ..., w\}$ **do**
  $\mathsf{send\_i}(\mathsf{AU}_i, l^{\mathsf{hnd}})$
end for
for all $i \in \{1, ..., n\}$ **do**
  $\mathsf{send\_i}(\mathsf{V}_i, l^{\mathsf{hnd}})$
end for

$num\_shares \leftarrow num\_shares + 1$
  /* Initially, $num\_shares = 0$ */
$C \leftarrow C \cup \{num\_shares \mapsto (ds^{\mathsf{hnd}}, dp^{\mathsf{hnd}})\}$
if $num\_shares \geq t$ **then**
  for all $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots num\_shares\}$ **do**
    $res^{\mathsf{hnd}} \leftarrow \mathsf{combine}(C(i_1), \ldots, C(i_t), pk^{\mathsf{hnd}})$
    if $res^{\mathsf{hnd}} \neq \perp$ **then**
      $res \leftarrow \mathsf{retrieve}(res^{\mathsf{hnd}})$
      Output res to the user and **stop**
    end if
  end for
end if

Fig. 2. Algorithms for tallying and for combining the results

sign their messages (plaintext shares). The same key can be used for several elections, in contrast to [27]. Such possibility is given by the functionality $\mathcal{F}_{\text{NIZK}}$, which can be implemented so, that the simulator has a trapdoor for extracting witnesses, and does not have to resort to rewinding the user H.

6.1 Security of the e-Voting System

Several security properties have been defined for e-voting protocols. Some properties can only be satisfied by policies or voting procedures. We only mention here the security properties in terms of cryptography.

An e-voting protocol should have the following properties [32].

Correctness. The voting results must be computed from only legitimate votes.
Privacy. Voter's preferences are private.
Coercion-freeness. A voter can not later prove that he/she voted in a particular way (Then he can not be forced to vote for something he does not prefer).
Independence. A voter should know his vote.

We claim that the e-voting system described above is secure in the above sense. Section 6.2 shows the proof. When the e-voting system uses the real library instead of the abstract one, the properties automatically preserved.

6.2 Proof for the Ideal Setting

The arguments below are quite similar to those in the Dolev-Yao model.

Correctness. The votes that the authority receives from the voters are signed. Hence the adversary could not change their content. Each authority accepts a vote from some $\mathsf{M^V}_i$ only once, hence there are no possibilities for replaying the votes. Because of the adversary simulating the bulletin board, the same set of votes reaches each authority. When combining the plaintext shares, the sets of votes must be the same and at least one of the shares must originate from an honest authority. The plaintext shares cannot be interfered by the adversary, as this would invalidate the correctness proofs of decryption.

Privacy. The vote privacy can be defined as the secrecy of payloads [9]. To achieve it, the inability of the adversary to get handles to the actual votes is sufficient. But the terms representing the votes are only ever used in the ciphertexts the voters send to the authorities and possibly also in plaintext shares if the adversary chooses to decrypt a vote. But the adversary has corrupted at most $(t-1)$ authorities, hence it cannot combine the shares of a decrypted vote.

After the tallying phase, the adversary learns only the result because all of authorities just give the decryption shares for the correct result.

Coercion-freeness. After the voting protocol, a voter has handles to his/her vote, the encrypted vote and the proof of validity of the vote. The only way that the ideal library allows one to verify that the ciphertext represents the vote, is to decrypt the ciphertext. The adversary does not have sufficiently many shares of the decryption key for that.

Independence. The library does not offer any means for a party to change the plaintext of a ciphertext without decrypting that ciphertext. Hence an adversarial voter cannot change the vote of an honest voter and present it as its own. An adversarial voter cannot even copy the vote of another voter because algorithm 1 does not allow repetitions.

7 Discussion and Conclusions

We have extended the UC cryptographic library with threshold homomorphic encryption. While extending it, we have made some design choices, the optimality of which can only be decided by using the library in the design and analysis of protocols. In particular, we have chosen to verify the proofs of validity and decryption correctness at the time where the ciphertexts and plaintext shares are actually used. One could imagine the existence of special commands to verify those proofs, and a condition on the user of the library (leading to *conditional reactive simulatability* [4]) to always verify the proofs before decrypting or combining. With the current choice we have avoided introducing the conditions, thereby making the presentation of the library simpler.

As threshold homomorphic encryption is widely applied, this extended library can be used in analysing various protocols. It enables us to achieve computationally sound proofs for a larger class of protocols, including e-voting, in an easier way (by tools or even by hand). As an example, we specify and analyse a simple e-voting protocol in Appendix 6.

Conditional reactive simulatability puts conditions on the *user* of some cryptographic primitive with UC-secure abstraction. In this paper we have shown that it may be equally important to consider restrictions on the possible behavior of the *adversary* trying to attack the *ideal* system. We have seen that the application of the composition theorem may combine those conditions in various ways, sometimes leading to their disappearance. This phenomenon certainly warrants a more thorough investigation.

UC cryptographic library, when combined with conditions put on the user and on the adversary, is one approach possibly leading to machine-assisted verification of security of cryptographic protocols and larger systems. A rather different approach is the *sequence-of-games* approach [29,16,17,18,36] that has seemingly received more attention recently. We believe both approaches have their unique merits and both deserve attention.

References

1. Adão, P., Fournet, C.: Cryptographically sound implementations for communicating processes. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 83–94. Springer, Heidelberg (2006)
2. Backes, M.: A Cryptographically Sound Dolev-Yao Style Security Proof of the Otway-Rees Protocol. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 89–108. Springer, Heidelberg (2004)
3. Backes, M., Dürmuth, M.: A cryptographically sound Dolev-Yao style security proof of an electronic payment system. In: CSFW 2005, pp. 78–93 (2005)
4. Backes, M., Dürmuth, M., Hofheinz, D., Küsters, R.: Conditional reactive simulatability. Int. J. Inf. Sec. 7(2), 155–169 (2008)
5. Backes, M., Laud, P.: Computationally sound secrecy proofs by mechanized flow analysis. In: ACM CCS 2006, pp. 370–379 (2006)
6. Backes, M., Pfitzmann, B.: A Cryptographically Sound Security Proof of the Needham-Schroeder-Lowe Public-Key Protocol. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 1–12. Springer, Heidelberg (2003)
7. Backes, M., Pfitzmann, B.: Symmetric Encryption in a Simulatable Dolev-Yao Style Cryptographic Library. In: CSFW 2004, pp. 204–218 (2004)
8. Backes, M., Pfitzmann, B.: Limits of the cryptographic realization of Dolev-Yao-style XOR. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 178–196. Springer, Heidelberg (2005)
9. Backes, M., Pfitzmann, B.: Relating Symbolic and Cryptographic Secrecy. In: IEEE S&P 2005, pp. 171–182 (2005)
10. Backes, M., Pfitzmann, B.: On the cryptographic key secrecy of the strengthened Yahalom protocol. In: SEC 2006 (IFIP 201), pp. 233–245 (2006)
11. Backes, M., Pfitzmann, B., Waidner, M.: Symmetric authentication within a simulatable cryptographic library. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 271–290. Springer, Heidelberg (2003)
12. Backes, M., Pfitzmann, B., Waidner, M.: A Universally Composable Cryptographic Library. In: ACM CCS 2003, pp. 220–230 (2003)
13. Backes, M., Pfitzmann, B., Waidner, M.: A General Composition Theorem for Secure Reactive Systems. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 336–354. Springer, Heidelberg (2004)
14. Backes, M., Pfitzmann, B., Waidner, M.: Limits of the BRSIM/UC soundness of Dolev-Yao models with hashes. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 404–423. Springer, Heidelberg (2006)
15. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS 1997, pp. 394–403 (1997)
16. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)

17. Blanchet, B.: A computationally sound mechanized prover for security protocols. In: IEEE S&P 2006, pp. 140–154 (2006)
18. Blanchet, B.: Computationally sound mechanized proofs of correspondence assertions. In: CSF 2007, pp. 97–111 (2007)
19. Canetti, R.: A unified framework for analyzing security of protocols. In: ECCC, vol. 8(16) (2001)
20. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS 2001, pp. 136–145 (2001)
21. Damgård, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 424–436. Springer, Heidelberg (2000)
22. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
23. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Transactions on Information Theory IT-29(12), 198–208 (1983)
24. Fouque, P.-A., Pointcheval, D.: Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 351–368. Springer, Heidelberg (2001)
25. Fouque, P.-A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
26. Goldreich, O.: Foundations of Cryptography. Volume 1 - Basic Tools. Cambridge University Press, Cambridge (2001)
27. Groth, J.: Evaluating security of voting schemes in the universal composability framework. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 46–60. Springer, Heidelberg (2004)
28. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
29. Laud, P.: Symmetric encryption in automatic analyses for confidentiality against active adversaries. In: IEEE S&P 2004, pp. 71–85 (2004)
30. Laud, P.: Secrecy Types for a Simulatable Cryptographic Library. In: ACM CCS 2005, pp. 26–35 (2005)
31. Laud, P., Ngo, L.: Threshold Homomorphic Encryption in the Universally Composable Cryptographic Library. Cryptology ePrint Archive, Report 2008/367 (2008)
32. Lipmaa, H.: Secure electronic voting protocols. In: The Handbook of Information Security. John Wiley & Sons, Chichester (2006)
33. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
34. Pfitzmann, B., Waidner, M.: A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In: IEEE S&P 2001, pp. 184–200 (2001)
35. Sprenger, C., Backes, M., Basin, D.A., Pfitzmann, B., Waidner, M.: Cryptographically sound theorem proving. In: CSFW 2006, pp. 153–166 (2006)
36. Tšahhirov, I., Laud, P.: Application of dependency graphs to security protocol analysis. In: Barthe, G., Fournet, C. (eds.) TGC 2007. LNCS, vol. 4912, pp. 294–311. Springer, Heidelberg (2008)
37. Wikström, D.: Universally composable DKG with linear number of exponentiations. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 263–277. Springer, Heidelberg (2005)

Universally Composable Security Analysis of TLS*

Sebastian Gajek[1], Mark Manulis[2], Olivier Pereira[2], Ahmad-Reza Sadeghi[1], and Jörg Schwenk[1]

[1] Ruhr University Bochum, Germany
{sebastian.gajek,joerg.schwenk}@nds.rub.de
ahmad.sadeghi@trust.rub.de
[2] Université Catholique de Louvain, Belgium
{mark.manulis,olivier.pereira}@uclouvain.be

Abstract. We present a security analysis of the complete TLS protocol in the Universal Composable security framework. This analysis evaluates the composition of key exchange functionalities realized by the TLS handshake with the message transmission of the TLS record layer to emulate secure communication sessions and is based on the adaption of the secure channel model from Canetti and Krawczyk to the setting where peer identities are not necessarily known prior the protocol invocation and may remain undisclosed. Our analysis shows that TLS, including the Diffie-Hellman and key transport suites in the uni-directional and bi-directional models of authentication, securely emulates secure communication sessions.

Keywords: Universal Composability, TLS/SSL, key exchange, secure sessions.

1 Introduction

The protocol framework of *Transport Layer Security (TLS)* [1] serves as fundamental primitive for WWW security and has fostered to the most valuable cryptographic protocol family in practice. The TLS protocol suites enable applications to communicate across a distributed network in a way that endpoint authentication and transmission privacy is guaranteed. The main goal of this paper is to provide a rigorous and generic analysis of TLS's cryptographically relevant parts of the protocol framework, namely the handshake and record-layer protocols. Given the wide deployment of TLS and the fact that it has been designed as contemporary cryptography started to explore provable security, it is natural that this analysis is of high, practical interest. Since TLS has already been investigated with respect to certain cryptographic primitives and protocol abstractions (see below), a general belief is that the framework is secure. Yet, there is no security proof of the entire TLS protocol and a careful observation

* A full version of this paper is available at http://eprint.iacr.org/2008/251

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 313–327, 2008.

of TLS's subtleties in the various modes provided by the different cipher suites. However, such a proof would significantly contribute to the analysis of complex protocols executed on top of TLS.

Our analysis is carried out in the meanwhile classical model of Universally Composable (UC) security [2] which guarantees protocol security under general composition with arbitrary other protocols. This valuable property stimulated the search for universal protocol design techniques and their realizations [3,4,5,6,7,8]. On the other hand, there are important impossibility results [3,9] so that a security proof of TLS in this model is neither obvious nor trivial. Our work particularly continues the way of Canetti's and Krawczyk's consideration of the Σ-protocol underlying the signature based modes in IPSec [10] and their model to build up secure channels [11] in the UC model with the exception that instead of proving single modes, we utilize UC as technique to prove the complete protocol secure in a single proof. Applied to the analysis of TLS, it includes Diffie-Hellman and encrypted key transport in the uni- or bi-directional model of authentication which are part of the TLS handshake, and their emulation to build secure communication protocols realized by the additional TLS record layer.

The most relevant question is how to reduce the complexity of the proof. Is it possible to unitize TLS in meaningful protocol fragments such that the composition theorem allows for an efficient protocol reformulation in the hybrid model? That means, can we define ideal functionalities that capture the cryptographic task of some of its fragments and simply reuse these functionalities with the next fragment? Otherwise, a composite analysis would not make sense so that we could switch to stand-alone protocol proofs. Fortunately, we answer the questions in the positive. To this end, we introduce two ideal functionalities, dubbed the *universal key exchange* and *universal secure communication sessions*. The functionalities are "universal" in the sense that they emulate different key establishment methods and modes of authentication in a self-contained definition. We show that the TLS framework including the different modes securely emulates the universal secure sessions functionality in the presence of *non-adaptive adversaries*. Our result can significantly simplify security proofs of higher-layer protocols by employing the composition theorem. We are not aware of any prior work that evaluates the essential composability property of TLS.

Related Work. Because of its eminent role the TLS framework has been repeatedly peer-reviewed. Schneier and Wagner [12] gave the first informal analysis in the core specification. Bleichenbacher [13] found some weaknesses in the PKCS#1 standard for RSA encryption as used with some SSL 3.0 handshake protocols.[1] Jonsson and Kaliski [14] showed that the encryption in the revised PKCS#1.5 standard is secure against chosen cipher attacks in the Random Oracle Model. Krawczyk [15] analyzed the composition of symmetric authentication and encryption to establish a secure communication channel with TLS record

[1] Note that the attack exploited weaknesses of the PKCS#1 standard and not the TLS protocol.

layer protocols and found some problems in the case of general composition. However, these do not apply to the standard cipher suites.

Apart from the analysis of some cryptographic primitives, a line of research addressed the analysis of *dedicated* TLS protocols on the basis of cryptographic abstractions to allow automated proof techniques. Paulson [16] gave an inductive analysis of a simplified version of TLS, using the theorem proving tool Isabelle. Mitchell, Shmatikov, and Stern [17] checked TLS, using the finite-state enumeration tool named Murphϕ. Ogata and Futatsugi [18] used the interactive theorem prover OTS/CafeObj to check a simplified version of the key transport handshake protocol through equational reasoning. He *et al.* [19] provided a proof of correctness of TLS in conjunction with the IEEE 802.11i wireless networking protocol, using the Protocol Composition Logic. The drawback these tool-supported approaches currently share is that the proofs are considerably simplified. They follow the *Dolev-Yao model* [20] which represents cryptography as term algebras and abstracts away the comprehensiveness of the adversary such that the proofs are not known to be cryptographically sound.

Very recently, Morrissey *et al.* [21] analyzed in an independent work the modularity of a *TLS-related* handshake protocol in a game-based style. The handshake is not exactly conform with the core TLS specification [1] and considers not all protocol variants. Their work focuses on a generic proof of the iterated session key constructions. By contrast, our work is of independent interest and practical relevance. We investigate TLS's intrinsic compositional property which is to provide higher-layer protocols with some secure communication functionality. Furthermore, our work addresses the native handshake protocols and additionally the record layer protocols in different authentication models under the stronger security notion of universally composable security.

Organization. The remaining sections are structured as follows. Section 2 clarifies notation and cryptographic building blocks. Section 3 shortly introduces the TLS protocol family and describes the compositional proof idea. Section 4 is devoted to the TLS handshake subroutines we use throughout the analysis. Section 5 proves the full framework and Section 6 concludes.

2 Preliminaries

2.1 Notations

The protocols run between two players: a client and a server. A player P may act as *initiator* I or *responder* R. If P is acting as I then by $\bar{P}$ we denote a player acting as R and viceversa. An anonymous player, i.e. a party whose identity is not known is denoted by $\perp$. We refer to the handshake protocol structure as π and the composition with the record-layer protocols as ρ. Additionally, we use different indices to capture the modes of authentication in ideal functionalities. We refer to a responder-only authenticated functionality as $\mathcal{F}^1$, i.e. a functionality where the responder authenticates to the initiator, but the initiator's identity remains unknown. Further, we denote an ideal functionality, where both players authenticate by $\mathcal{F}^2$, and a hybrid functionality of $\mathcal{F}^1$ and $\mathcal{F}^2$ by $\mathcal{F}^{(1,2)}$.

2.2 Cryptographic Building Blocks and Their Constructions

The specification of TLS [1] uses several cryptographic primitives and mandates or recommends certain instantiations of them as described in the following:

An ASYMMETRIC ENCRYPTION SCHEME $(\mathtt{ENC}_{pk_R}(), \mathtt{DEC}_{sk_R}())$ for transporting the encrypted premaster secret which must be instantiated with the RSA-OAEP construction (known to provide indistinguishability under adaptive chosen ciphertext attacks [14] in the Random Oracle Model). In TLS handshake a private key sk_R is known to the responder R and its public key pk_R is signed by a Certification Authority (CA).

A DIGITAL SIGNATURE SCHEME $(\mathtt{SIG}_{sk}(), \mathtt{VER}_{vk}())$ for entity authentication which can be instantiated with DSA and RSA-PSS (the latter is known to provide weak existential unforgeability under chosen message attacks in the Random Oracle Model [22]). Each player owns a signing key sk and the respective verification vk is certified by a CA.

A MESSAGE AUTHENTICATION CODE function $\mathtt{HMAC}_k()$ from [23] and a SYMMETRIC ENCRYPTION SCHEME $(\mathtt{E}_k(), \mathtt{D}_k())$ which is recommended to be DES or 3DES in different modes and with different key lengths. The construction of symmetric authentication *with* encryption is known to provide weak unforgeability under chosen message attacks and indistinguishability under chosen plaintext attacks [15,24].

A PSEUDO-RANDOM FUNCTION for the key derivation and confirmation, denote here by $\mathtt{PRF}_k()$. It is evaluated with seed k on an input string l_i, $i \in [1, 4]$ which is labeled with different publicly known space delimiters and two independently chosen random values, i.e. the nonces exchanged in the first protocol, or a function thereof. The specification defines a special construction based on HMAC combiners which has been recently proven to be a good randomness extractor [25].

3 Transport Layer Security

3.1 TLS in a Nutshell

The standard TLS specification [1] comprises handshake, alert, change cipher spec, and record layer (sub)protocols. The *handshake protocol* is used to negotiate key material and cryptographic algorithms and the record layer protocol can then be applied to secure transmitted application data. The *change cipher spec protocol* consisting of one message triggers a change in the cryptographic parameters used by the record layer, while the *alert protocol* communicates error messages, whenever a failure during the handshake or message protection occurs. Thus, the essential cryptographic building blocks for TLS and target to the presented analysis are the handshake and record layer protocols.

Handshake and Record Layer. The TLS handshake aims at the negotiation of a common secret called the *master secret* k_m which is in turn derived from the the previously established *premaster secret* k_p. The modularity of the handshake

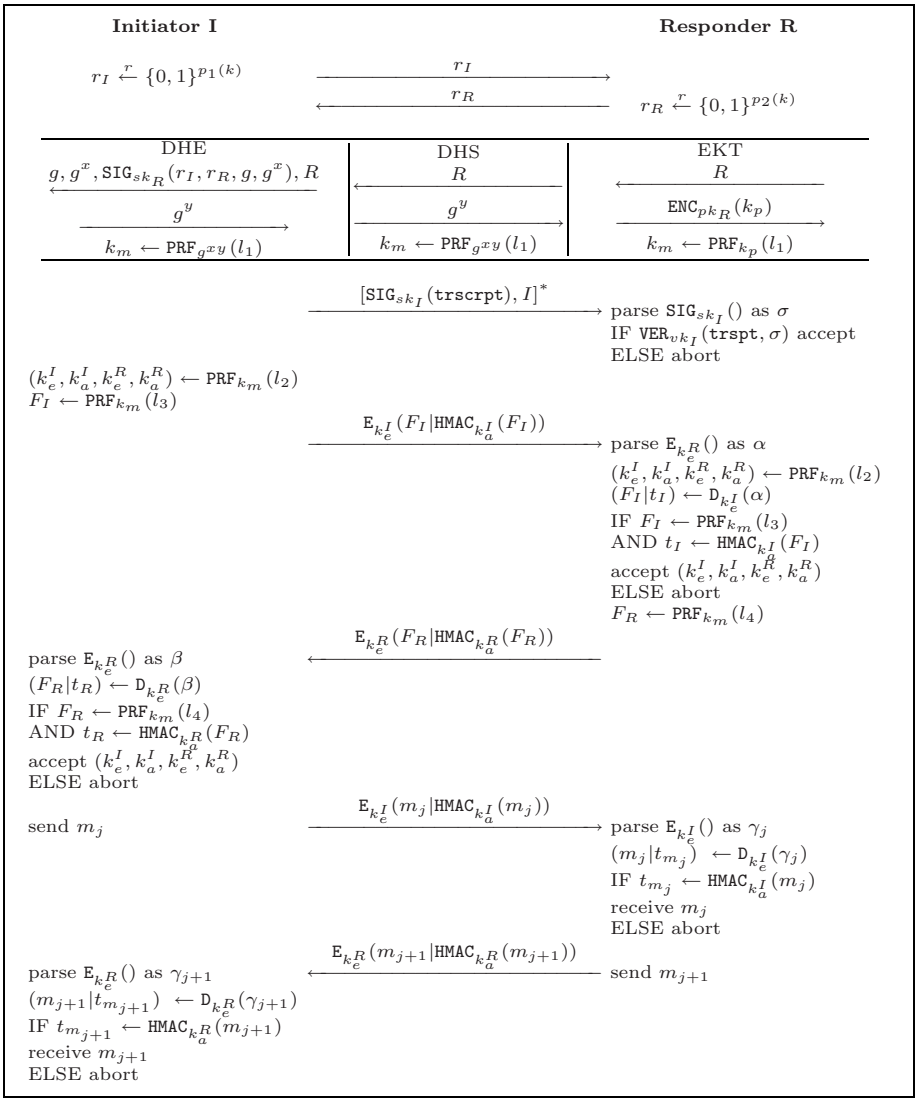

Fig. 1. The TLS protocol including the different subroutines DHE, DHS, and EKT to establish the master secret k_m. $[\cdot]^*$ marks the optional client authentication message. Event 'abort' invokes the alert protocol with the respective error message; events 'send' and 'receive' trigger interfaces to the application layer.

protocol is captured by the fact that different subroutines are applied to establish the premaster secret and derive the master secret while the remaining structure of the handshake is unchanged (see Fig. 1). TLS distinguishes among the following subroutines: encryption of the premaster secret using the server's public key (EKT); static (DHS) or ephemeral signed (DHE) Diffie-Hellman key

exchange. Optionally, TLS allows for the client authentication via a signature over all received values `trscrpt` which can be verified using the public key with the client certificate. The master secret k_m is then used to derive up to four cryptographic keys for the record layer: two symmetric encryption keys k_e^P (including an initialization vector for the block-cipher based encryption), and two authentication keys k_a^P, where $P \in \{I, R\}$. Finally, client and server confirm the negotiated security parameters by exchanging their finished messages which are derived from k_m and protected via *authenticated encryption* by the record layer (i.e. MAC of the plaintext is used as input to the symmetric encryption). The same protection is then applied to the subsequent application data.

Remark 1. Note that an application message may be fragmented and compressed when processed by the record layer. Therefore, the record layer encodes sequence numbers into the fragments and maintains a counter in order to prevent disorder. Note also that a key feature of TLS is session resumption in order to reduce server-sided performance penalties. The client names an earlier session that it intends to continue; if the server agrees, the previous master secret is used with the new nonces to generate new key material for the record layer. Though not explicitly treated in our paper, it is easy to see that the security of the abbreviated handshake follows from our analysis of the full handshake.

3.2 Roadmap for the Modular Analysis of TLS

The structure of the TLS framework advocates its modular analysis. Intuitively, the handshake protocol captures the cryptographic task of key exchange and the composition with the record layer protocol emulates secure transfer of application messages. However, the straightforward idea to model the complete handshake protocol as ideal key exchange functionality in order to negotiate the session keys and compose it with the record layer protocol in order to realize a secure communication sessions functionality fails in general. The handshake protocol does *not* securely realize the ideal key exchange functionality since it uses the derived session keys to encrypt and authenticate finished messages. Thus, the environment can test the keys using the finished messages and tell the two worlds apart.

In our analysis we avoid this obstacle by devising a functionality $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ that emulates the handshake's subroutines to negotiate the master secret k_m (instead of a straight-line computation of the session keys). $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ captures the fact that two players receive a random key unless either player is corrupted. Next, we demonstrate that the subroutines DHE, DHS, and EKT securely realize $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ (Section 4). Our analysis is focused on responder-only and mutual authenticated communication which are the authentication modes supported by TLS (apart from anonymous Diffie-Hellman suites). Since TLS operates in a setting where a Certificate Authority (CA) is required, we formalize the global setup assumption by formulating the real-world protocols in $\mathcal{F}$-hybrid models, utilizing the *certification functionality* $\mathcal{F}_{\mathrm{CERT}}$, *certified public key encryption functionality* $\mathcal{F}_{\mathrm{CPKE}}$, and *certificate authority functionality* $\mathcal{F}_{\mathrm{CA}}$, as presented in [26,27].

The composition with these functionalities to a subroutine protocol is preserved by the *universal composition with joint state (JUC) theorem*, proposed in [28]. This operation is similar to universal composition with the exception that multiple instances of a protocol can have a joint state. It is useful in the case of key exchange when multiple subroutine protocol sessions have access to the same instance of functionalities $\mathcal{F}_{\text{CERT}}$, $\mathcal{F}_{\text{CPKE}}$, and $\mathcal{F}_{\text{CA}}$ that use the same key for authenticating multiple messages (i.e. the signature, encryption, and deposited key is the joint state, respectively). Finally, we make use of the composition theorem and specify the TLS protocol in the $\mathcal{F}_{\text{KE}}^{(1,2)}$-hybrid model. We show that the reformulated TLS protocol securely realizes the ideal functionality for the secure communication sessions (Section 5).

4 Analysis of TLS Subroutines

We proceed with the specification of an ideal-world functionality which we henceforth call *universal key exchange* $\mathcal{F}_{\text{KE}}^{(1,2)}$ that captures the requirements of the subroutines DHE, DHS, and EKT. The key exchange functionality $\mathcal{F}_{\text{KE}}^{(1,2)}$ is illustrated in Fig. 2. It mimics the cryptographic task that the players I and R agree upon a shared secret μ which is indistinguishable from an independently chosen value of the same length as long as a party is uncorrupted. There is a large body of literature that covers ideal key exchange functionalities (e.g. [2,11,27]). $\mathcal{F}_{\text{KE}}^{(1,2)}$ is similar to these functionalities except for:

Functionality $\mathcal{F}_{\text{KE}}^{(1,2)}$

$\mathcal{F}_{\text{KE}}^{(1,2)}$ proceeds as follows when parameterized with security parameter k.

- Upon receiving an input ("establish-key", SID, ID_I) from some party, where $ID_I \in (\perp, I)$, record ID_I as initiator, and send a message ("establish-session", SID, ID_I) to the adversary. Upon receiving input ("establish-key", SID, R) from some other party, record R as responder, and send the message ("establish-key", SID, R) to the adversary.
- Upon receiving an answer ("impersonate", SID, $\tilde{\mu}$) from the adversary, do: If $ID_I = \perp$, record the adversary as initiator and send message ("Key", SID, $\perp$, $\tilde{\mu}$) to the responder. Else, ignore the message.
- Upon receiving an answer ("Key", SID, P, $\tilde{\mu}$) from the adversary, where P is either the initiator or responder, do: If neither initiator nor responder is corrupted, and there is no recorded key, fix μ uniformly from $\{0,1\}^k$. If either initiator or responder is corrupted, and there is no recorded key, record $\mu \leftarrow \tilde{\mu}$ as the adversary. Send message ("Key", SID, $\bar{P}$, μ) to P.

Fig. 2. The Universal Key Exchange Functionality. $\mathcal{F}_{\text{KE}}^{2}$ is identical to $\mathcal{F}_{\text{KE}}^{1}$ except that it excludes the impersonation query.

First, the players authenticate in a post-specified fashion, i.e. the environment invokes players with the session identifier SID and optionally their own identity. A player learns its peer identity while executing the TLS protocol (captured by the fact that peer identities are given by the functionality and not in the setup). This is an essential difference of TLS to related protocols (e.g. SSH) where the players have already negotiated their public keys before the protocol start.

Second, the functionality defines a hybrid notion of authenticated key exchange. When the initiator is parameterized with an identity, i.e. $ID_I{=}I$, the functionality assures mutual authentication between the initiator and server. Then, the functionality randomly fixes the (master) key unless a party is corrupt. On the other hand, when the initiator is invoked with an anonymous identity, i.e. $ID_I{=}\perp$, the functionality guarantees a matching conversation between the responder and some party whose identity is unknown. Consequently, the adversary can impersonate the initiator and fix the master key.[2] The corresponding case in the real world is that the environment instructs the adversary to replay the key exchange protocol with the exception that it contributes to the pre-master key. The initiator is unable to terminate the session while the responder accepts the session. Technically, the functionality deploys the session identifier SID to determine the anonymous player. Such technicality is only feasible for a two party functionality. Recall that the SIDs of all Turing machines in a protocol instance must be identical in the UC framework. Any player participating in the same session who is not a responder must be a potential initiator.

Third, the functionality is defined for non-adaptively corrupting adversaries and therefore excludes (perfect) forward secrecy. In fact, this exclusion is precisely what makes it possible to define a single universal key exchange functionality which covers both, key transport and Diffie-Hellman key agreement.

Theorem 1. *Protocol* EKT *in the* $\mathcal{F}_{\text{CPKE}}$, DHE *in the* $\mathcal{F}_{\text{CERT}}$, *and* DHS *in the* $\mathcal{F}_{\text{CA}}$*-hybrid model securely realize* $\mathcal{F}_{\text{KE}}^{1}$. *Protocol* EKT *in the* $(\mathcal{F}_{\text{CPKE}},\mathcal{F}_{\text{CERT}})$, DHE *in the* $(\mathcal{F}_{\text{CERT}},\mathcal{F}_{\text{CERT}})$, *and* DHS *in the* $(\mathcal{F}_{\text{CA}},\mathcal{F}_{\text{CERT}})$*-hybrid model securely realize* $\mathcal{F}_{\text{KE}}^{2}$.

The proof appears in the full version.

5 TLS UC-Realizes Secure Communication Sessions

The natural abstraction of TLS is to allow secure communication between players in a single protocol instance. While the handshake protocol aims at securely sharing uniformly distributed session keys, the record layer protocol provides authenticated encryption of session messages.

[2] Note that in case of Diffie-Hellman the key exchange functionality does not consider key control issues (see [5]). However, this has no impact on the security of secure communication sessions because the impersonator learns the master key and thus derives the session keys for the protection of the messages.

5.1 Universal Secure Communication Sessions

Secure communication sessions have been discussed in [2,11] for the general case in which all players are authenticated. We refine the functionality and relax the requirements to the universal model of authentication in the post-specified setting, where a player learns the identity of its peer during the execution of the protocol and must cope with impersonation attacks against the initiator, provided the environment keeps the initiator's identity secret. In which case, we have to expect a real-world adversary that plays the role of the initiator by intercepting the first two protocol rounds, choosing own premaster secret, and completing the protocol in the normal way. The initiator will be unable to terminate the session. Nevertheless, the responder accepts the session and answers to the adversary, mimicking arbitrary party. We capture the requirements by formulating a universal secure communication sessions functionality $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$ in Fig. 3. Let us highlight some characteristics of $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$ in the following:

Functionality $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$

$\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$ proceeds as follows, when parameterized by a leakage function $l : \{0,1\}^* \to \{0,1\}^*$.

- Upon receiving an input ("establish-session", `SID`, ID_I) from some party, where $ID_I \in (\perp, I)$, record ID_I as initiator, and send the message to the adversary. Upon receiving input ("establish-session", `SID`, R) from some party, record R as responder, and forward the message to the adversary.
- Upon receiving a value ("impersonate", `SID`) from the adversary, do:If (ID_I=$\perp$), check that no `ready` entry exists, and record the adversary as initiator. Else ignore the message.
- Upon receiving a value ("send", `SID`, m, $\bar{P}$) from party P, which is either initiator or responder, check that a record (`SID`, P, $\bar{P}$) exists, record `ready` (if there is no such entry) and send ("sent", `SID`, $l(m)$) to the adversary and a private delayed value ("receive", `SID`, m, P) to $\bar{P}$. Else ignore the message. If the sender is corrupted, then disclose m to the adversary. Next, if the adversary provides m' and no output has been written to the receiver, then send ("send", `SID`, m', P') to the receiver unless P' is an identity of an uncorrupted party.

Fig. 3. The Universal Secure Communication Sessions Functionality

First, the functionality handles a uni- and bi-directional model of authentication (as in the universal key exchange functionality). The latter is accomplished by invoking the players with their own identity. The first is realized by invoking the initiator with an empty identity value $\perp$ allowing the adversary to mount an impersonation attack. The functionality proceeds in the usual way except that a secure session is established between the adversary and the responder.

Second, the functionality guarantees that the adversary gains no information other than some side channel information about the transmitted plaintext m, expressed via a leakage function $l(m)$, when the adversary has neither impersonated nor corrupted a player. In particular, the information leakage includes the length and sequence number of m and some information concerning the transmitted messages' source and destination; thus, modeling network information about the TLS-protected channel from lower-layer protocols and higher-layer protocols prior to their processing by the record layer. (We remark that the environment may provide additional leakage information such as the domain name and the name of the client application. This leakage information may be important upon composition with a dedicated higher-layer protocol).

Third, the session identifier `SID` assures that the functionality may address the initiator even though its identity is undisclosed (because it knows the responder's identity and the underlying system model permits a party, i.e. the initiator, to interact with the functionality with an identical session identifier). This is so because TLS runs above transport-layer protocols which provide the players with a globally unique address (e.g. IP address). Furthermore, these protocols ensure that the channel is locally fresh by exchanging a pair of nonces.

Last, the functionality manages an internal ready state. This technicality ensures that in the responder-only model of authentication the adversary cannot impersonate the initiator after the responder agreed upon the session keys and switched into the pending state waiting for the transmission.

5.2 Protocol ρ Realizes $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$

In Fig. 4 we apply Theorem 1 and reformulate protocol ρ in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid model. The general Universal Composability theorem guarantees that no probabilistic polynomial time-bounded environment distinguishes between the case that it observes an instance of TLS executing the subroutines DHE, DHS and EKT and the case that it interacts with a TLS instance where the subroutines are replaced by the ideal key exchange functionality. We are now ready to state our main theorem.

Theorem 2. *Protocol ρ in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid model securely realizes $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$.*

Proof. Let $\mathcal{A}$ be a real-world adversary that operates against ρ. We construct an ideal-world adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between the case that it interacts with $\mathcal{A}$ and parties running ρ in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid model or with $\mathcal{S}$ in the ideal world for $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$. $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$ and mimics an interaction with players executing ρ. It tries to make the internal protocol simulation consistent with the real protocol execution and the limitation that it has no information about the transmitted message m other than its length $l(m)$. The simulator allows the adversary $\mathcal{A}$ to attack the simulated protocol execution in arbitrary way throughout the simulation. $\mathcal{S}$ emulates the protocol execution in such a way that $\mathcal{A}$ thinks that it intercepts a real-world execution of ρ, and

Protocol ρ

1. Upon activation with query ("establish-session", SID, ID_I) by $\mathcal{Z}$, where $ID_I \in (\perp, I)$, the initiator sends the init message (r_I) where $r_I \xleftarrow{r} \{0,1\}^{p_1(k)}$ is a nonce. Upon activation with query ("establish-key", SID, R) by $\mathcal{Z}$, the responder waits for the receipt of the init message. It responds with own nonce $r_R \xleftarrow{r} \{0,1\}^{p_2(k)}$ and initializes a copy of $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ with session identifier $\mathtt{SID}_{\mathrm{KE}}=(r_I|r_R)$ by sending query ("establish-key", $\mathtt{SID}_{\mathrm{KE}}$, R) to $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$.
2. Upon receiving the response message, the initiator calls $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ with session identifier $\mathtt{SID}_{\mathrm{KE}}=(r_I|r_R)$ on query ("establish-key", $\mathtt{SID}_{\mathrm{KE}}$, ID_I) and waits for the delivery of output ("Key", $\mathtt{SID}_{\mathrm{KE}}$, R, μ). It then computes the session keys $(k_e^I, k_a^I, k_e^R, k_a^R) \leftarrow \mathtt{PRF}_\mu(l_2)$ and the finished value $F_I \leftarrow \mathtt{PRF}_\mu(l_3)$. Additionally, the initiator sends the final initiator message $(\mathtt{E}_{k_e^I}(F_I|\mathtt{HMAC}_{k_a^I}(F_I)))$.
3. When the responder receives the final initiator message (α), it first waits for the delivery of ("Key", $\mathtt{SID}_{\mathrm{KE}}$, ID_I, μ) from $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$. Then, the responder computes in the same way the session keys $(k_e^I, k_a^I, k_e^R, k_a^R) \leftarrow \mathtt{PRF}_\mu(l_2)$ for the players. It decrypts the final initiator message $(F_I|t_I) \leftarrow \mathtt{D}_{k_e^I}(\alpha)$ and verifies that $F_I \leftarrow \mathtt{PRF}_\mu(l_3)$ and $t_I \leftarrow \mathtt{HMAC}_{k_a^I}(F_I)$. If the verification fails, it aborts. Otherwise, it computes the finished value $F_R \leftarrow \mathtt{PRF}_\mu(l_4)$ and sends the final responder message $(\mathtt{E}_{k_e^R}(F_R|\mathtt{HMAC}_{k_a^R}(F_R)))$.
4. Upon delivery of the final responder message (β), the initiator decrypts the message $(F_R|t_R) \leftarrow \mathtt{D}_{k_e^R}(\beta)$. Then, it verifies that $F_R \leftarrow \mathtt{PRF}_\mu(l_4)$ and $t_R \leftarrow \mathtt{HMAC}_{k_a^R}(F_R)$. If the verification fails, it aborts.
5. Once the session keys are agreed upon, the sender $P \in (I, R)$ waits for the transmission notification ("send", SID, m, $\bar{P}$) from $\mathcal{Z}$. It then sends $\mathtt{E}_{k_e^P}(m|t_m)$ whereby message m is authenticated through the tag $t_m \leftarrow \mathtt{HMAC}_{k_a^P}(m)$. Upon receiving the message γ, the receiver $\bar{P}$ decrypts the message $(m|t_m) \leftarrow \mathtt{D}_{k_e^P}(\gamma)$ and verifies that $t_m \leftarrow \mathtt{HMAC}_{k_a^P}(m)$. If the verification fails, it aborts. Otherwise, the receiver accepts the message and makes the local output ("receive", SID, m, P) to $\mathcal{Z}$.

Fig. 4. The full TLS Framework Structure, in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid Model

such that its interaction with $\mathcal{Z}$ is distributed computationally indistinguishable from that observed by the environment in the real-world execution.

In detail, the simulator proceeds in the following way:

1. **Simulating invocation of I.** Upon receiving ("establish-session", SID, ID_I) from $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$, $\mathcal{S}$ feeds $\mathcal{A}$ with the init message (r_I) where $r_I \xleftarrow{r} \{0,1\}^{p_1(k)}$.
2. **Simulating invocation of R.** Upon receiving ("establish-session", SID, R) from $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$, $\mathcal{S}$ waits for receipt of an init message (r_I') from $\mathcal{A}$. Then, it chooses a nonce $r_R \xleftarrow{r} \{0,1\}^{p_2(k)}$ and feeds $\mathcal{A}$ with the response message (r_R, R). Finally, it calls $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ on query ("establish-key", $\mathtt{SID}'_{\mathrm{KE}}$, R), where $\mathtt{SID}'_{\mathrm{KE}}=(\mathtt{SID} \circ \mathtt{r}'_{\mathtt{I}}|\mathtt{r}_{\mathtt{R}})$.
3. **Simulating receipt of a response message by I.** Upon $\mathcal{A}$ delivers the message (r'_R, P') to I, $\mathcal{S}$ proceeds as follows:

(a) $\mathcal{S}$ verifies that I has previously sent the init message (r_I).
(b) $\mathcal{S}$ checks that P'=R. Otherwise, it aborts the simulation.
(c) $\mathcal{S}$ mimics on behalf of I the master key generation by invoking a copy of $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$. The master key is obtained by handing $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ the message ("establish-key", $\mathtt{SID}_{\mathrm{KE}}$, ID_I), where $\mathtt{SID}_{\mathrm{KE}}$=($\mathtt{SID} \circ \mathtt{r_I}|\mathtt{r'_R}$) and waiting for the delivery of the response message ("Key", $\mathtt{SID}_{\mathrm{KE}}$, R, μ). Otherwise, $\mathcal{S}$ terminates with an internal error message (because there was no matching activation of the same instance of $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ in form of a query ("establish-key", $\mathtt{SID}_{\mathrm{KE}}$, R) by the simulator on behalf of the responder).
(d) $\mathcal{S}$ defines the master key μ, the session keys $(k_e^I, k_a^I, k_e^R, k_a^R)$, and the finished value F_I to be random values Δ_{k_m}, $(\Delta_{k_e^I}, \Delta_{k_a^I}, \Delta_{k_e^R}, \Delta_{k_a^R})$, and Δ_{F_I} chosen from the appropriate spaces, respectively.
(e) $\mathcal{S}$ feeds $\mathcal{A}$ with the final initiator message $(\mathtt{E}_{\Delta_{k_e^I}}(\Delta_{F_I}|t_I))$, where $t_I \leftarrow \mathtt{HMAC}_{\Delta_{k_a^I}}(\Delta_{F_I})$.

4. **Simulating receipt of a final initiator message by** R**.** When $\mathcal{A}$ delivers the message (α) to R, $\mathcal{S}$ proceeds as follows:
(a) $\mathcal{S}$ verifies that it has previously received an init message (r'_I) and sent a response message (r_R, R).
(b) $\mathcal{S}$ waits for the master key by mimicking the key establishment process of $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$. Now we distinguish between the following two distinct cases.
Case 1 (no impersonation): If $\mathcal{S}$ receives an answer ("Key", $\mathtt{SID}_{\mathrm{KE}}$, ID_I, μ) from $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$, then no impersonation attack has occurred. In this case $\mathcal{S}$ uses for k_m, $(k_e^I, k_a^I, k_e^R, k_a^R)$, and F_I exactly the same values Δ_{k_m}, $(\Delta_{k_e^I}, \Delta_{k_a^I}, \Delta_{k_e^R}, \Delta_{k_a^R})$, and Δ_{F_I} that it has chosen on behalf of the initiator before. Then, it waits for the delivery of the final initiator message and applies the session keys to decrypt $(F'_I|t'_I) \leftarrow \mathtt{D}_{\Delta_{k_e^I}}(\alpha)$. $\mathcal{S}$ compares whether $F'_I = \Delta_{F_I}$ and $t'_I = t_I$. If the verification fails, it aborts the simulation. Otherwise, it chooses F_R to be a random value Δ_{F_R} from the same space and feeds $\mathcal{A}$ with the final responder message $(\mathtt{E}_{\Delta_{k_e^R}}(\Delta_{F_R}|t_R))$ where $t_R \leftarrow \mathtt{HMAC}_{\Delta_{k_a^R}}(\Delta_{F_R})$. Then, $\mathcal{S}$ prepares for the secure message exchange on behalf of R.
Case 2 (impersonation): If $\mathcal{S}$ receives an answer ("Key", $\mathtt{SID}'_{\mathrm{KE}}$, P', $\tilde{\mu}$), then the original master key has been modified by the adversary implying the impersonation attack framing the initiator. In this case $\mathcal{S}$ computes $(k_e^I, k_a^I, k_e^R, k_a^R)$, and F_I as specified in the protocol, i.e. $(k_e^I, k_a^I, k_e^R, k_a^R) \leftarrow \mathtt{PRF}_{\tilde{\mu}}(l_2)$, and $F_I \leftarrow \mathtt{PRF}_{\tilde{\mu}}(l_3)$. Then, it waits for the delivery of the final initiator message and decrypts $(F'_I|t'_I) \leftarrow \mathtt{D}_{k_e^I}(\alpha)$. $\mathcal{S}$ compares whether $F'_I = F_I$ and $t'_I = \mathtt{HMAC}_{k_a^I}(F_I)$. If the verification fails, it aborts the simulation. Otherwise, $\mathcal{S}$ computes $F_R \leftarrow \mathtt{PRF}_{\tilde{\mu}}(l_4)$, and feeds $\mathcal{A}$ with the final responder message $(\mathtt{E}_{k_e^R}(F_R|\mathtt{HMAC}_{k_a^R}(F_R)))$. Finally, $\mathcal{S}$ sends ("impersonate", $\mathtt{SID}$) to $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$. This is exactly the point in the simulation where the adversary has impersonated the unauthenticated party. Then, $\mathcal{S}$ continues the simulation with the exception that the interaction proceeds with $\mathcal{A}$ and I aborts the protocol.

Note that in all subsequent simulation steps, $\mathcal{S}$ uses session keys (k_e^P, k_a^P) for $P \in (I, R)$ and finished values F_I and F_R obtained from one of the above two cases.

5. **Simulating receipt of a final responder message by I.** When $\mathcal{A}$ delivers the message (β) to an uncorrupted I, $\mathcal{S}$ proceeds as follows:
   (a) $\mathcal{S}$ verifies that it has previously sent an init message (r_I), received a response message (r'_R, P'), and sent a final initiator message $(\mathtt{E}_{\Delta_{k_e^I}}(\Delta_{F_I}|t_I))$.
   (b) $\mathcal{S}$ uses its own session keys $(\Delta_{k_e^R}, \Delta_{k_a^R})$ to decrypt β obtaining $F'_R|t'_R$. Since no responder impersonation attacks may occur it aborts the simulation if $F'_R \neq \Delta_{F_R}$ or $t'_R \neq t_R$ whereby Δ_{F_R} and t_R are the values used by $\mathcal{S}$ on behalf of R in the previous simulation step 4b (case 1). If the simulation does not abort then $\mathcal{S}$ prepares for the secure message exchange on behalf of I.
6. **Simulating Message Transmission.** Upon receiving ("sent", SID, $l(m)$) from $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$, $\mathcal{S}$ extracts from $l(m)$ the sender and receiver identities. It then chooses a random message $\Delta_m \xleftarrow{r} \{0,1\}^{l(m)}$ and feeds $\mathcal{A}$ with message $\mathtt{E}_{k_e^P}(\Delta_m|t_{\Delta_m})$ where $t_{\Delta_m} \leftarrow \mathtt{HMAC}_{k_a^P}(\Delta_m)$.
7. **Simulating Message Reception.** Upon receiving the message (γ), the receiver decrypts the message $(\Delta'_{m'}, t'_{\Delta_{m'}}) \leftarrow \mathtt{D}_{k_e^P}(\gamma)$ and then verifies that $t'_{\Delta_{m'}} \leftarrow \mathtt{HMAC}_{k_a^P}(\Delta_m)$ using its own keys. If the verification fails, it aborts. Otherwise, $\mathcal{S}$ signals $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$ to send the message.
8. **Simulating Static Corruption.** If one of the parties gets corrupted, then $\mathcal{S}$ proceeds by emulating a ρ protocol session, just as a honest party would play it. In particular, $\mathcal{S}$ uses the message m transmitted by $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$ in the emulation of the last protocol round.

The proof of indistinguishability to demonstrate the validity of $\mathcal{S}$ appears in the full version.

6 Conclusion

We have analyzed the TLS protocol family in the framework of Universal Composition. We have shown that the complete TLS protocol framework securely realizes secure communication sessions. Thus, future analysis of composite protocols can be considerably simplified by calling the secure communication functionality in the hybrid-model reformulation. The composition theorem preserves that security holds under general composition with arbitrary players. Our analysis is performed under the consideration of static corruptions, since this setting is suitable for the combined treatment of key transport *and* Diffie-Hellman protocol suites specified within the TLS standard. A future work may include consideration of adaptive corruptions, and thus modeling of (perfect) forward secrecy, which seems to be achievable by the TLS protocol suites based on Diffie-Hellman but not key transport.

Acknowledgment

We would like to thank Dennis Hofheinz, Aggelos Kiayias, Ralf Küsters, and Ivan Visconti for fruitful discussions and their valuable feedback. The authors were supported by the European Commission (IST-2002-507932 ECRYPT).

References

1. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol, Version 1.1. RFC 4346, IETF (2006); Proposed Standard
2. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)
3. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
4. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
5. Hofheinz, D., Müller-Quade, J., Steinwandt, R.: Initiator-Resilient Universally Composable Key Exchange. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 61–84. Springer, Heidelberg (2003)
6. Katz, J.: Universally Composable Multi-Party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
7. Canetti, R., Krawczyk, H., Nielsen, J.: Relaxing Chosen-Ciphertext Security. Cryptology ePrint Archive, Report 2003/174 (2003)
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally Composable Two-Party and Multi-Party Secure Computation. In: STOC 2002, pp. 494–503. ACM, New York (2002)
9. Kidron, D., Lindell, Y.: Impossibility Results for Universal Composability in Public-Key Models and with Fixed Inputs. Cryptology ePrint Archive, Report 2007/478 (2007)
10. Canetti, R., Krawczyk, H.: Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002)
11. Canetti, R., Krawczyk, H.: Universally Composable Notions of Key Exchange and Secure Channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
12. Schneier, B., Wagner, D.: Analysis of the SSL 3.0 Protocol. In: Proceedings of the 2nd USENIX Workshop on Electronic Commerce (1996)
13. Bleichenbacher, D.: Chosen Ciphertext Attacks against Protocols based on the RSA Encryption Standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
14. Jonsson, J., Kaliski, B.: On the Security of RSA Encryption in TLS. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 127–142. Springer, Heidelberg (2002)
15. Krawczyk, H.: The Order of Encryption and Authentication for Protecting Communications (or: How Secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)

16. Paulson, L.C.: Inductive Analysis of the Internet Protocol TLS. ACM Transactions on Computer and System Security 2(3), 332–351 (1999)
17. Mitchell, J.C., Shmatikov, V., Stern, U.: Finite-State Analysis of SSL 3.0. In: Proceedings of the 7th Conference on USENIX Security Symposium, p. 16 (1998)
18. Ogata, K., Futatsugi, K.: Equational Approach to Formal Analysis of TLS. In: ICDCS 2005, pp. 795–804. IEEE Computer Society Press, Los Alamitos (2005)
19. He, C., Sundararajan, M., Datta, A., Derek, A., Mitchell, J.C.: A Modular Correctness Proof of IEEE 802.11i and TLS. In: ACM Conference on Computer and Communications Security CCS 2005, pp. 2–15. ACM, New York (2005)
20. Dolev, D., Yao, A.C.C.: On the Security of Public Key Protocols. IEEE Transactions on Information Theory 29(2), 198–207 (1983)
21. Morrissey, P., Smart, N.P., Warinschi, B.: A Modular Security Analysis of the TLS Handshake Protocol. Cryptology ePrint Archive, Report 2008/236 (2008)
22. Jonsson, J.: Security Proofs for the RSA-PSS Signature Scheme and Its Variants. Cryptology ePrint Archive, Report 2001/053 (2001)
23. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
24. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
25. Fouque, P.A., Pointcheval, D., Zimmer, S.: HMAC is a Randomness Extractor and Applications to TLS. In: AsiaCCS 2008, pp. 21–32. ACM Press, New York (2008)
26. Canetti, R.: Universally Composable Signature, Certification, and Authentication. In: CSFW 2004, pp. 219–233. IEEE CS, Los Alamitos (2004), http://eprint.iacr.org/2003/239
27. Canetti, R., Herzog, J.: Universally Composable Symbolic Analysis of Mutual Authentication and Key-Exchange Protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 380–403. Springer, Heidelberg (2006)
28. Canetti, R., Rabin, T.: Universal Composition with Joint State. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
29. Hansen, S., Skriver, J., Nielson, H.: Using Static Analysis to Validate the SAML Single Sign-On Protocol. In: Proceedings of the 2005 Workshop on Issues in the Theory of Security (2005)
30. Groß, T., Pfitzmann, B., Sadeghi, A.R.: Browser Model for Security Analysis of Browser-Based Protocols. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 489–508. Springer, Heidelberg (2005)
31. Groß, T., Pfitzmann, B., Sadeghi, A.R.: Proving a WS-Federation Passive Requestor Profile with a Browser Model. In: Workshop on Secure Web Services. ACM Press, New York (2005)

Round Optimal Universally Composable Oblivious Transfer Protocols

Huafei Zhu

C&S Department, I^{2}R, A-STAR, Singapore
huafei@i2r.a-star.edu.sg

Abstract. In this paper, a round optimal oblivious transfer protocol is proposed and analyzed. Our protocol is built upon the top of an oblivious double-trapdoor encryption scheme (the double-trapdoor information consisting of a master key and a local key). The idea behind our construction is that the master key is used to extract the exact input messages of a corrupted sender (as a result, a simulator designated for the corrupted sender is constructed) while the local key is used to extract the exact input message of a corrupted receiver (as a result, a simulator designated for the corrupted receiver is defined). We show that our protocol is universally composable in the common reference string model assuming that the decisional Diffie-Hellman problem over a squared composite modulus of the form $N = pq$ is hard.

Keywords: Double trap-door cryptosystems, oblivious transfer, simulator, universally composable.

1 Introduction

The oblivious transfer introduced by Rabin [15], and extended by Even, Goldreich and Lempel [8] and Brassard, Crépeau and Robert [2] is one of the most basic and widely used protocol primitives in cryptography. An oblivious transfer protocol allows one party called receiver to get exactly one of the two (or more) values from another party called the sender. The receiver is oblivious to the other values while the sender is oblivious to which value was received. The concept of oblivious transfer protocol stands at the center of the fundamental results on secure two-party and multi-party computation showing that any efficient functionality can be securely computed ([17] and [10]). Due to its general importance, the task of constructing efficient oblivious transfer protocols has attracted much interest.

Naor and Pinkas [12] first constructed efficient oblivious transfer protocols based on the decisional Diffie-Hellman assumption. Tauman [16] generalized Naor and Pinkas' results based on a variety of concrete assumptions building on the top of projective hash framework of Cramer and Shoup [7]. The primary drawback of these constructions is that their security is only proven according to a semi-simulation definition of security (i.e., a receiver security is defined by requiring that a sender's view of the protocol when the receiver chooses index

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 328–334, 2008.

σ_0 is indistinguishable from a view of protocol when the receiver chooses index σ_1. The sender security follows the real/ideal world paradigm and guarantees that any malicious receiver in the real world can be mapped to a receiver in an idealized game in which the oblivious transfer protocol is implemented by a trusted third party.)

Very recently, Camenisch, Neven and Shelat [3] proposed practical oblivious transfer protocols that are provable secure according to a full-simulation definition (the security employs the real/ideal world paradigm for both receiver and the sender. The difficulty in obtaining secure oblivious transfer protocols in this model is the strict security requirement of simulation based definition). Subsequently, Green and Hohenberger [9] proposed simulation secure oblivious string transfer protocols based on a weaker set of static assumptions on bilinear groups.

Lindell [11] presented the first efficient implementation of fully-simulatable oblivious bit transfer protocols under the decisional Diffie-Hellman problem, the Nth residuosity and quadratic residuosity assumptions as well as the assumption of that homomorphic encryption exists. All protocols are nice since they are provably secure in the presence of malicious adversaries under the real/ideal model model simulation paradigm without using general zero-knowledge proofs under standard complexity assumptions. The idea behind Lindell's construction is that it makes use of the cut-and-choose technique so that each party is not required to prove in zero-knowledge and allows a simulator to rewind the malicious party so that an expected polynomial time simulator under the standard cryptographic primitives can be defined.

Unfortunately, all these schemes mentioned above are NOT known to be secure when composed in parallel and concurrent. For example, the proof technique presented in [11] explicitly use the rewinding technique that is forbidden for proof a protocol is universally composable (the environment $\mathcal{Z}$ canNOT be rewound at all). At Crypto'08, a practical implementation of universally composable secure oblivious transfer protocol is proposed by Peikert, Vaikuntanathan and Waters [14]. Their protocols are based on a new abstraction called a dual-mode cryptosystem. Such a system starts with a setup phase that produces a common reference string which is made available to all parties. The cryptosystem is set up in one of two modes: extraction mode and decryption mode. A crucial of the dual-mode cryptosystem is that no adversary can distinguish, given the common reference string between two modes. To prove the sender's security (secure against a malicious sender), a simulator must run a trap-door information extractable algorithm that given a trap-door t, outputs (pk, sk_0, sk_1), where pk is a public encryption key and sk_0 and sk_1 are corresponding secret keys for index 0 and 1 respectively. To prove the receiver's security (secure against a malicious receiver), a simulator must run a find-lossy algorithm (i.e., an index information extractable algorithm) that given a trap-door t and pk, outputs an index corresponding to the message-lossy index of pk.

1.1 This Paper

In this paper, we propose a new construction for round optimal oblivious transfer protocols and show that:

Theorem: Our protocol (described in Section 4) is universally composable in the common reference string model assuming that the decisional Diffie-Hellman problem over a squared composite modulus of the form $N = pq$ is hard.

Using Yao-style garble circuit [17], we can use the proposed oblivious transfer protocol to obtain round optimal universally composable two-party computation for non-reactive functionalities. That is,

Corollary [14]: There exists a 2-round (respectively, 3-round) protocol that securely realizes any non-reactive functionality $\mathcal{F}$ for which only one party receives output (respectively both parties) in the $\mathcal{F}_{\mathsf{crs}}$-hybrid model.

Road-map: The rest of this paper is organized as follows: in Section 2, universally composable framework is sketched. A concrete implementation of oblivious double decryption cryptosystem is proposed in Section 3; In section 4, a round optimal universally composable oblivious transfer protocol is presented and analyzed. We conclude this work in Section 5.

2 Universally Composable Model

We work in the standard universally composable framework of Canetti [4]. The universally composable framework defines a probabilistic polynomial time (PPT) environment machine $\mathcal{Z}$. $\mathcal{Z}$ oversees the execution of a protocol π in the real world involving PPT parties and a real world adversary $\mathcal{A}$. $\mathcal{Z}$ also oversees the execution of a protocol in the ideal world involving dummy parties and an ideal world adversary $\mathcal{S}$ (a simulator). We refer to [4] for a detailed description of the executions, and definitions of $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ and $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$.

Definition 1. *Let $\mathcal{F}$ be a functionality. A protocol π is said to universally composable realize $\mathcal{F}$ if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, the ensemble $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable with the ensemble $REAL_{\pi,\mathcal{A},\mathcal{Z}}$.*

The common reference string model $\mathcal{F}_{\mathsf{crs}}$ produces a string with a distribution that can be sampled by a PPT algorithm $\mathcal{D}$.

Functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$ (due to [6])

$\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$ runs with parties $P_1, \cdots, P_n$ and is parametrized by an algorithm $\mathcal{D}$.

- when receiving a message (sid, P_i, P_j) from P_i, let $\mathsf{crs} \leftarrow \mathcal{D}(1^n)$ and send (sid, crs) to P_i and send (crs, P_i, P_j) to the adversary. Next when receiving (sid, P_i, P_j) from P_j (and only from P_j), send (sid, crs) to p_j and to the adversary, and halt.

The functionality of an oblivious transfer involves a sender S with input (x_0, x_1) and a receiver R with input $\sigma \in \{0,1\}$. R learns x_σ and S learns nothing at all. These requirements are captured by the specification of the oblivious transfer functionality $\mathcal{F}_{OT}$ from [5]

Functionality $\mathcal{F}_{OT}$ (due to [5])

- Upon receiving a message (sid, sender, x_0, x_1) from S, where each $x_i \in \{0,1\}^l$, store (x_0, x_1);
- Upon receiving a message (sid, receiver, σ) from R, check if (sid, sender, $\cdots$) message was previously sent. If yes, send (sid, x_σ) to R and sid to the adversary and halt. If not, send nothing to R.

3 Oblivious Double-Trapdoor Cryptosystem

Oblivious double trap-door cryptosystem described below constructed from [1]. The proposed oblivious double trap-door cryptosystem is initialized in a trusted setup phase, which produces a common reference string crs known to all participants along with some trap-door information t.

- Setup algorithm Setup: on input a security parameter n, Setup produces composite modulus of the form $N = pq$ that is a product of two safe primes p and q. Setupd also outputs two random elements g_0 and g_1 of order $\lambda(N)$ in $Z^*_{N^2}$, where $\lambda(\cdot)$ is Carmichael function. Setup randomly chooses $x_0, x_1 \in [0, N^2/2]$ and sets $h_i = g_i^{x_i} \bmod N^2$ $(i = 0, 1)$. The common reference string crs is defined as (g_0, h_0, g_1, h_1) and N. The auxiliary trap-door information is defined as $t = (p, q, x_0, x_1)$. The auxiliary string (p, q) is called the master key and (x_0, x_1) is called a local key.
- Key generation algorithm KeyGen: on input crs and $\sigma \in \{0,1\}$, KeyGen randomly chooses $r \in [0, N/4]$, and computes the cipher-text (g, h) where $g = g_\sigma^r \bmod N^2$ and $h = h_\sigma^r \bmod N^2$. Let $pk = (g, h)$ and $sk = r$;
- Encryption algorithm Enc, on input crs, a bit $b \in \{0,1\}$ and a message $m \in Z_N$, Enc performs the following computations:
  1) Randomly choosing $s, t \in [0, N/4]$;
  2) Computing $u = g_b^s h_b^t$, $v = g^s h^t (1+N)^m$;
  The output of Enc is the cipher-text c $(=(u, v))$ of the message m.
- Decryption algorithm Dec: On input sk and c, Dec recovers the message m from the equation $v/u^r \bmod N^2 = 1 + mN$

We stress that if $b = \sigma$, then the above encryption scheme in essence, is Bresson, Catalano and Pointcheval's cryptosystem. Thus, assuming that the decisional Diffie-Hellman problem defined over a squared composite modulus of the form $N = pq$ is hard, the scheme described above is semantically secure. In case that $b \neq \sigma$, then (g_b, h_b, g, h) is not a Diffie-Hellman quadruple and the output of the decryption algorithm is a random message (hence the name of oblivious double-trapdoor encryption scheme).

4 Oblivious Transfer Protocol

In this section, we describe a round optimal oblivious transfer protocol. We then show that our round optimal oblivious transfer protocol is universally composable.

4.1 Description of Protocol

Our protocol based on oblivious double trap-door cryptosystem

Inputs:

- The input of a sender S: $(sid, ssid, m_0, m_1)$, where $m_0, m_1 \in Z_N$;
- The input of a receiver R: $(sid, ssid, \sigma)$, where $\sigma \in \{0, 1\}$;

Initial Step: S is first activated by sending $(sid, ssid, S, R)$ to $\mathcal{F}_{\mathsf{crs}}$. S gets back crs and sid (by running the setup algorithm Setup). R then is activated, and sends (sid, S, R) to $\mathcal{F}_{\mathsf{crs}}$, and gets back crs and sid.
Step 1: R generates $(pk, sk) \leftarrow$ the key generation algorithm KeyGen described above. R sends pk to S and stores $(sid, ssid, sk)$;
Step 2: S gets $(sid, ssid, pk)$ from R, and then computes $c_b = \mathsf{Enc}(pk, b, m_b)$ for each $b \in \{0, 1\}$, and sends $(sid, ssid, c_0, c_1)$ to R;
Step 3: R gets $(sid, ssid, c_0, c_1)$ from S and outputs $(sid, ssid, \mathsf{Dec}(sk, m_\sigma))$.

Notice that our protocol is only two rounds, it follows that our protocol is round optimal.

4.2 The Proof of Security

We claim that

Theorem 1. *The oblivious transfer protocol described above is universally composable in the common reference string model assuming that the decisional Diffie-Hellman problem over a squared composite modulus of the form $N = pq$ is hard.*

Proof. We consider the following two cases:

Case 1: Suppose the sender S is corrupted by a statical adversary $\mathcal{A}$, we now define a simulator sim_S below (the task of sim_S is to extract the input messages of $\mathcal{A}$):

- sim_S runs the setup algorithm Setup and obtains crs and the corresponding trapdoor information t, where $\mathsf{crs} = (N, g_0, h_0, g_1, h_1)$, $t = (p, q)$ such that $N = pq$. We remark that in case that the sender S is corrupted, sim_S needs not to know the trap-door information (x_0, x_1) such that $h_i = g_i^{x_i}$ $(i = 0, 1)$. We also remark that in case that both the sender S and the receiver R are corrupted, sim needs to know the trap-door information (p, q) and (x_0, x_1) such that $N = pq$ and $h_i = g_i^{x_i}$ $(i = 0, 1)$.
- when parties query the ideal functionality $\mathcal{F}_{\mathsf{crs}}$, return (sid, crs) to them;

- when a dummy party R is activated on (sid, $ssid$, crs), sim_S computes (pk, sk) as that in the real protocol described above and sends pk to $\mathcal{A}$ as it comes from R;
- when sim_S obtains $(sid, ssid, c_0, c_1)$ from $\mathcal{A}$, sim_S decrypts (c_0, c_1) using the auxiliary string (p, q) and gets $\mathcal{A}$'s two input messages (m_0, m_1);
- sim_S now forwards (m_0, m_1) to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$.

Since the underlying double encryption scheme is semantically secure, it follows that the ensemble $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable with the ensemble $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$.

Case 2: Suppose the receiver R is corrupted by a statical adversary $\mathcal{A}$, we now define a simulator sim_R below (the task of sim_R is now to extract the input index σ of $\mathcal{A}$):

- sim_R runs the setup algorithm Setup and obtains crs and the corresponding trapdoor information t, where $\mathsf{crs} = (N, g_0, h_0, g_1, h_1)$, $t = (x_0, x_1)$ such that $h_i = g_i^{x_i}$ $(i = 0, 1)$. We remark that in case that the receiver R is corrupted, sim_R needs not to know the trap-door information (p, q) such that $N = pq$. To enable the simulator sim_R extract input σ of the malicious receiver R, we allow the simulator sim_R knows the trapdoor information (x_0, x_1) such that $h_i = g_i^{x_i}$ $(i = 0, 1)$. Same remark as above if both the sender S and the receiver R are corrupted, sim needs to know the trap-door information (p, q) and (x_0, x_1) such that $N = pq$ and $h_i = g_i^{x_i}$ $(i = 0, 1)$.
- when parties query the ideal functionality $\mathcal{F}_{\mathsf{crs}}$, return ($sid$, crs) to them;
- when sim_R obtains pk $(= g, h)$, sim_R tests the validity of the equation $h = g^{x_0}$. If the equation is valid, then sim_R outputs $\sigma = 0$, otherwise $h = g^{x_1} \neq g^{x_0}$, in this case, sim_R outputs $\sigma = 1$;
- sim_R now forwards σ to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$ and receives the output (sid, ssid, m_σ).
- when the dummy S is activated for sub-session (sid, ssid), S looks up the corresponding σ and m_σ, and computes $c_\sigma \leftarrow \mathsf{Enc}(pk, \sigma, m_\sigma)$ and $c_{1-\sigma} \leftarrow \mathsf{Enc}(pk, 1 - \sigma, 0^l)$

It is straightforward to verify that ensemble $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable with the ensemble $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ assuming that the underlying encryption is semantically secure.

5 Conclusion

We have proposed round optimal oblivious transfer protocols based on the notions of oblivious encryption schemes. We have shown that our protocol is universally composable in the common reference string model assuming that the decisional Diffie-Hellman problem over a squared composite modulus of the form $N = pq$ is hard.

References

1. Bresson, E., Catalano, D., Pointcheval, D.: A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003)
2. Brassard, G., Crépeau, C., Robert, J.-M.: All-or-Nothing Disclosure of Secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
3. Camenisch, J., Neven, G., Shelat, A.: Simulatable Adaptive Oblivious Transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
4. R. Canetti: a new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)
5. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC 2002, pp. 494–503 (2002)
6. Canetti, R., Rabin, T.: Universal Composition with Joint State. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
7. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
8. Even, S., Goldreich, O., Lempel, A.: A Randomized Protocol for Signing Contracts. Commun. ACM 28(6), 637–647 (1985)
9. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
10. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In: STOC 1987, pp. 218–229 (1987)
11. Lindell, Y.: Efficient Fully-Simulatable Oblivious Transfer. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 52–70. Springer, Heidelberg (2008)
12. Naor, M., Pinkas, B.: Computationally Secure Oblivious Transfer. J. Cryptology 18(1), 1–35 (2005)
13. Paillie, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
14. Peikert, C., Vaikuntanathan, V., Waters, B.: A Framework for Efficient and Composable Oblivious Transfer. Crypto (2008)
15. Michael, O.: Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University (1981)
16. Kalai, Y.T.: Smooth Projective Hashing and Two-Message Oblivious Transfer. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 78–95. Springer, Heidelberg (2005)
17. Yao, A.C.-C.: Protocols for Secure Computations (Extended Abstract). In: FOCS 1982, pp. 160–164 (1982)

A Tamper-Evident Voting Machine Resistant to Covert Channels*

Wei Han[1], Tao Hao[1], Dong Zheng[1], Kefei Chen[1], and Xiaofeng Chen[2]

[1] Shanghai Jiaotong University, Shanghai, 200240, P.R.China
[2] Sun Yat-sen University, Guangzhou 510275, P.R.China

Abstract. To provide a high level of security guarantee cryptography is introduced into the design of the voting machine. The voting machine based on cryptography is vulnerable to attacks through covert channels. An adversary may inject malicious codes into the voting machine and make it leak vote information unnoticeably by exploiting the randomness used in encryptions and zero-knowledge proofs. In this paper a voting machine resistant to covert channels is designed. It has the following properties: Firstly, it is tamper-evident. The randomness used by the voting machine is generated by the election authority. The inconsistent use of the randomness can be detected by the voter from examining a destroyable verification code. Even if malicious codes are run in the voting machine attacks through subliminal channels are thwarted. Next, it is voter-verifiable. The voter has the ability to verify if the ballot cast by the machine is consistent with her intent without doing complicated cryptographic computation. Finally, the voting system is receipt-free. Vote-buying and coercion are prevented.

Keywords: electronic voting, covert channel, tamper-evident, receipt-free.

1 Introduction

Electronic voting will change the way we vote in the near future. Direct Recording Electronic (DRE) voting machines have been widely used in many political elections in recent years. The inner mechanism of the DRE is opaque to voters, which results in the debate on the trustworthiness of the machine. One of the most challenging issues is to design voting systems that can be trusted by human voters even if the election computers are running malicious codes. There have been some voting schemes in which a voter can get direct verification that her vote is correctly recorded by the DRE, such as Neff's MarkPledge scheme [1] and Moran-Naor's scheme [2]. The two schemes are based on cryptography and they are vulnerable to attacks through covert channels. A corrupt programmer may

* This work is supported by 973 Program (2007CB311201), National Natural Science Foundation of China (No. 60503006), and NSFC-KOSEF Joint Research Project (No. 60611140543).

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 335–344, 2008.

write malicious codes to leak voters' choices by encoding the randomness used in encryptions or zero-knowledge proofs in a secret and determinant way. When she has access to the bulletin board she can deduce the content of an encrypted ballot by reading publicly available information. We take Moran-Noar's voting scheme as an example. A voter operates the DRE in the voting booth as follows:

1. The voter chooses one candidate whom she supports.
2. The voter inputs challenges for other candidates.
3. The voter waits till the DRE prints a commitment to her choice on the receipt.
4. The voter inputs the challenge for the candidate she chooses.
5. The voter waits till the DRE prints the rest of the receipt. She compares the challenges in her mind with the challenges printed on the receipt. If they are consistent she takes the receipt and leaves the voting booth, or else she complains to the voting authority.

In such a scheme attacks can be mounted through covert channels. We divide the adversaries into two types: weak adversary and strong adversary. The weak adversary does not interact with the voter. The weak adversary only injects malicious codes into the DRE. When the DRE uses randomness it encodes the voter's choice into the random string in a private way which is known to the adversary. A strong adversary may be able to both inject malicious programs into the DRE and coerce voters. A strong adversary can mount attacks in the following cases:

Case 1. The adversary can require a coerced voter to use a special challenge which is recognized by the DRE. Once it knows a voter is coerced, the DRE can change the vote as it wishes. The coerced voter will not complain even if she detects the inconsistence.

Case 2. The adversary can control the order of voters entering the voting booth. The voting order is stored into the DRE. So the DRE can distinguish the coerced voters and change their ballots.

Case 3. The adversary can make use of timing channels. The time interval of striking keys on the keyboard of the DRE can be used to identify a coerced voter.

In this paper, we make the assumption that the arrival of voters is random so we can ignore case 2. We leave case 3 as an open problem. We only consider how to defend against the weak adversary and the strong adversary of case 1.

The attacks through covert channels are cumbersome. Public code audit may mitigate the effect to some extent, however, it is difficult to ascertain that the audited code is the actual code that gets loaded into the DRE. In order to escape detection a corrupt software developer may instruct the DRE to switch its behavior to a correct mode in the course of code test.

To prevent covert channels we can require the randomness used by the DRE is generated by the voting authority. The voting authority may be composed

of several election trustees who generate the randomness via multi-party computation. In such a way trust is distributed. The construction of witness of correct randomness use is difficult since it must be done without exposing the randomness used, and it must not introduce new covert channels. Choi, Golle and Jakobsson presented the design of tamper-evident mix networks with auditable privacy [3]. Inspired by their ideas, a tamper-evident DRE voting machine is designed in this paper. The voter acts as a verifier who checks that the randomness used by the DRE is correctly generated by the voting authority.

The rest of the paper is organized as follows: the voting system model is outlined in section 2. Some building blocks are presented in section 3. The voting scheme is detailed in section 4. Its security is analyzed in section 5 and the conclusion is drawn in section 6.

2 System Model

In this paper we consider a 1-out-of-L voting scheme in which a voter chooses a candidate out of L candidates. Voters cast their ballots in a private voting booth. The following entities are involved: the voter, the DRE, the voting authority, the adversary and the assistant verifier.

The voter enters the voting booth and casts a ballot on the DRE. She audits that all the randomness used by the DRE is generated by the voting authority and that the DRE casts a ballot as intended.

The DRE generates encrypted ballots and posts them to a public bulletin board. The DRE prints a cryptographic receipt for the voter.

The voting authority maintains a list of all eligible voters. The authority is responsible to generate the randomness used by the DRE in encryptions. The authority sends each voter a verification code out of band, which is used by the voter to check if the given randomness is used by the DRE.

The assistant verifier is able to execute cryptographic computation. It can be any third party the voter trusts. When the voter leaves the booth, she gives her receipt to the verifier. The assistant verifier checks if the receipt is generated in accordance with some pre-defined rules.

Next we consider communication channels in the voting scheme: The bulletin board model is widely employed in electronic voting schemes. It is a public broadcast channel. Only eligible users can append messages on it. Messages posted cannot be tampered with. In our voting scheme the bulletin board is used for the authority and the DRE to publish information related to voting. The voting booth is modeled as an untappable channel between the voter and the DRE. The adversary can communicates with the DRE before it is deployed at the polling station. From this moment on, the adversary has no access to the DRE, and except the bulletin board the DRE has no communication media to transmit information to the adversary.

3 Preliminaries

Security requirements. A voting protocol should satisfy the security requirements below:

- Eligibility: Only eligible voters can participate in the election, and each eligible voter can cast a single vote.
- Privacy: The content of an individual ballot is kept secret. Only the final tally result is published.
- Verifiability: The validity of the individual ballot and the tally process can be verified. When any passive third party can also verify the ballot cast and the tally, this property is called universal verifiability.
- Robustness: The voting protocol can tolerate corrupt voters and dishonest authorities to some extent.
- Fairness: The partial results of the tally should not be exposed prior to the end of the voting phase.
- Receipt-freeness: The voter cannot provide a receipt to convince others how she casts a ballot. If a voter is given a cryptographic receipt for verification and the receipt can not be used to convince others that the voter chooses a particular candidate, the voting protocol still satisfies the receipt-freeness.

Encoding votes. We denote by M a strict upper bound on the number of voters. We represent L candidates with numbers $0, \cdots, L-1$ and encode a vote on candidate i as M^i . Tallying such encoded votes gives us an M-addic representation of the result $\sum_{i=0}^{L-1} v_i M^i$,where v_i is the number of votes on candidate i.

Homomorphic encryption. In this paper we make use of a semantically secure homomorphic threshold cryptosystem. A probabilistic public-key encryption function $E : P \times R \to C$ is homomorphic if for all $x_1, x_2 \in P, r_1, r_2 \in R$, it holds that $E(x_1; r_1) \otimes E(x_2; r_2) = E(x_1 + x_2; r_1 \oplus r_2)$, where P is a group which is the plaintext space, R is a group which is the randomness space, and C is a group which is the ciphertext space. The group operations in P, in R and in C are denoted by "$+$", "$\oplus$" and "$\otimes$" respectively. Examples of homomorphic cryptosystems are the additive version [4] of ElGamal [5] and Paillier [6]. They both are semantically secure and have threshold variants ([7] [8] [9]).

A cut-and-choose zero-knowledge proof protocol. Suppose E is a homomorphic encryption function. A prover P presents a verifier V with a ciphertext $c = E(m; r)$ and claims that it encrypts the plaintext m . By the homomorphic property P can convince V without disclosing r by the protocol below:

Common input: c

Private input for P: r such that $c = E(m; r)$

1. P chooses $r_0, r_1 \in R$ at random such that $r = r_0 \oplus r_1$. P computes $c_0 = E(m/2; r_0)$, $c_1 = c \oslash c_0$. "$\oslash$" is the inverse operation of "$\otimes$" . P sends c_0, c_1 to V.

2. V chooses $b \in \{0,1\}$ randomly and sends b to P.

3. P sends r_0 to V if $b = 0$, sends r_1 to V if $b = 1$. V checks that $c = c_0 \otimes c_1$ and $c_b = E(m/2; r_b)$.

The protocol is a typical construction with soundness $1/2$. It can be repeated k times to make the error probability achieve $1/2^k$.

4 The Proposed Voting Scheme

In this section, we present our voting scheme, which consists of the following steps:

Step 1. Voting system initialization

The voting authority posts the list of eligible voters on the bulletin board. It generates the key pairs of a homomorphic encryption cryptosystem. The public key and a cryptographic hash function H are published on the bulletin board and loaded into the firmware of the DRE. The secret key may be distributed in a threshold shared manner. The voting authority deploys DREs into polling stations.

The voting authority generates ballots for each voter as follows:

1. The authority firstly picks $r_1, r_2, ..., r_L \in R$ randomly and then generates a ballot set $BS = E(A_1; r_1), E(A_2; r_2), ..., E(A_L; r_L)$ including encrypted valid votes. We denote by A_i the coding of candidate $i (i = 1, 2, ..., L)$ for notation convenience. $E(A_i; r_i)$ is a valid encrypted vote for candidate i.
2. The authority generates a unique id for the ballot set BS.
3. For each encrypted ballot $E(A_i; r_i)$, the authority picks k pairs of random numbers $(r_{i,0}^{(1)}, r_{i,1}^{(1)})$, $(r_{i,0}^{(2)}, r_{i,1}^{(2)})$,..., $(r_{i,0}^{(k)}, r_{i,1}^{(k)})$ such that $r_{i,0}^{(s)} + r_{i,1}^{(s)} = r_i$, where k is a security parameter and $1 \le s \le k$. The authority then computes the row vector

$$[(E(A_i/2; r_{i,0}^{(1)}), E(A_i/2; r_{i,1}^{(1)})), (E(A_i/2; r_{i,0}^{(2)}), E(A_i/2; r_{i,1}^{(2)})), ..., (E(A_i/2; r_{i,0}^{(k)}), E(A_i/2; r_{i,1}^{(k)}))].$$

Note that each element in the row vector is a pair of encryptions on $A_i/2$ and the equation $E(A_i/2; r_{i,0}^{(s)}) \otimes E(A_i/2; r_{i,1}^{(s)}) = E(A_i; r_i)$ holds. All of the row vectors forms a ballot set matrix

$$\begin{bmatrix} (E(A_1/2; r_{1,0}^{(1)}), E(A_1/2; r_{1,1}^{(1)})) & (E(A_1/2; r_{1,0}^{(2)}), E(A_1/2; r_{1,1}^{(2)})) & ... & (E(A_1/2; r_{1,0}^{(k)}), E(A_1/2; r_{1,1}^{(k)})) \\ (E(A_2/2; r_{2,0}^{(1)}), E(A_2/2; r_{2,1}^{(1)})) & (E(A_2/2; r_{2,0}^{(2)}), E(A_2/2; r_{2,1}^{(2)})) & ... & (E(A_2/2; r_{2,0}^{(k)}), E(A_2/2; r_{2,1}^{(k)})) \\ ... & ... & ... & ... \\ (E(A_L/2; r_{L,0}^{(1)}), E(A_L/2; r_{L,1}^{(1)})) & (E(A_L/2; r_{L,0}^{(2)}), E(A_L/2; r_{L,1}^{(2)})) & ... & (E(A_L/2; r_{L,0}^{(k)}), E(A_L/2; r_{L,1}^{(k)})) \end{bmatrix}$$

4. The authority computes the hash values of the matrix BSM and presents the hash values in the form of a matrix

$$HBSM = \begin{bmatrix} (H(E(A_1/2; r_{1,0}^{(1)})), H(E(A_1/2; r_{1,1}^{(1)}))) & \cdots & (H(E(A_1/2; r_{1,0}^{(k)})), H(E(A_1/2; r_{1,1}^{(k)}))) \\ \cdots & \cdots & \cdots \\ (H(E(A_L/2; r_{L,0}^{(1)})), H(E(A_L/2; r_{L,1}^{(1)}))) & \cdots & (H(E(A_L/2; r_{L,0}^{(k)})), H(E(A_L/2; r_{L,1}^{(k)}))) \end{bmatrix}$$

The matrix $HBSM$ is the verification code for the DRE. Each row corresponds to the encrypted vote for a candidate. The authority prints the verification code on a sheet of paper and inserts the sheet into an envelope. The envelope is sealed and the ballot id number is printed on the surface of the envelope.

Step 2. Casting a ballot

On the election day the authority loads the ballot information including the id of the ballot set and all of the randomness used in the ballot generation into the DRE. The sealed envelopes are transported to each polling station.

After a voter's identity is verified, she is given a sealed envelope. The verification code is inside the envelope. The voter generates an k-bit string in a random manner such as flipping a coin under the inspection of the polling workers. The voter publishes the string as her challenge which will be used in the voting booth.

The voter enters the voting booth and starts the DRE. The id number shown on the screen should be equal to the id printed on the envelope. Assume that the voter chooses the candidate $t(1 \leq t \leq L)$ on the DRE. The DRE chooses the encryption $E(A_t; r_t)$ from the given ballot set BS. The DRE prints $E(A_t; r_t)$ on the receipt and posts it on the bulletin board. After the encryption is printed, the voter inputs the k-bit challenge string into the DRE. For the l-th bit $b_l (l = 1, 2, ...k)$, if $b_l = 0$, the DRE reveals $r_{i,0}^{(l)}$ in the l-th column of the matrix BSM $(i = 1, 2, ..., L)$; if $b_l = 1$, the DRE reveals $r_{i,1}^{(l)}$ in the l-th column of the matrix BSM. The revealed randomness is printed on the receipt. The DRE computes a matrix BSM' as follows: first, let $BSM = BSM'$. Next, some elements in BSM' will be modified. The t-th row in matrix BSM' is not changed. For the i-th row $(i \neq t)$, in each encryption pair $(E(A_i/2; r_{i,0}^{(s)}), E(A_i/2; r_{i,1}^{(s)}))$ $(s = 1, 2, ..., k)$ the revealed encryptions $E(A_i/2; r_{i,b_s}^{(s)})$ are maintained. The unrevealed encryptions $E(A_i/2; r_{i,\bar{b_s}}^{(s)})$ are modified as $E(A_t; r_t) \oslash E(A_i/2; r_{i,b_s}^{(s)})$. The DRE computes the hash value of each element in matrix BSM' to obtain a matrix $HBSM'$. The challenge and the matrix $HBSM'$ are printed on the receipt.

Step 3. Verification inside the voting booth

The voter opens the envelope and takes out the verification code. She compares the matrix $HBSM$ with the matrix $HBSM'$. The t-th row in $HBSM$ should be the same as that in $HBSM'$. In other rows, in each pair of hash values if $b_l = 0$ the two left components should both be equal to $H(E(A_i/2; r_{i,0}^{(l)}))$, otherwise the two right components should both be equal to $H(E(A_i/2; r_{i,1}^{(l)}))$. If the verification fails the voter must immediately complain to the polling workers.

Step 4. Verification outside the voting booth

The voter takes the receipt and goes out of the booth. She destroys the verification code under the inspection of the polling workers. Polling workers check that the challenge string printed on the receipt is the same as the one the voter committed to before entering the booth. If the two challenge strings are inconsistent, the ballot cast by the voter is cancelled and the cancellation is announced on the bulletin board at once. The voter submits her receipt to the assistant verifier. The verifier checks: 1. The encrypted ballot $E(A_t; r_t)$ on the receipt has been published on the bulletin board. 2. The matrix $HBSM'$ is correctly computed by the hash function. Note that the verifier can use the revealed randomness $r_{i,b_l}^{(l)}$ to reconstruct the matrix BSM'. 3. In the matrix BSM' $E(A_t; r_t)$ equals the product of two components in each pair. If any inconsistence is detected, the assistant verifier will raise an alert and complain to the legal authority on behalf of the voter.

Step 5. Tallying

When the voting phase ends, the voting authority can aggregate the ciphertexts of the ballots by the homomorphic property of the cryptosystem. The aggregated ciphertext may be threshold decrypted by multiple authorities. In this case a zero-knowledge proof is posted on the bulletin board to convince the public that the decryption is correctly performed. It is straightforward to extract the voting result from the plaintext. Using tamper-evident mix networks [3] to tally is an alternative choice.

Step 6. Verifying the tally

Any passive third party interested in the election can re-compute the aggregated ciphertext and verify the zero-knowledge proof of correct decryption.

5 Security Analysis and Discussions

Firstly, we can prove that the proposed voting scheme satisfies the requirements of electronic voting:

- **Eligibility.** The list of eligible voters is published on the bulletin board. The voter's identity is verified before she enters the voting booth.
- **Privacy.** The ballot posted on the bulletin board is encrypted. The verification code is destroyed when the voter leaves the booth. The receipt does not reveal the content of the encrypted ballot.
- **Verifiability.** The tally can be re-computed and verified by any third party by using information published on the bulletin board.
- **Robustness.** The encryptions of ballots are generated by the voting authority. If the DRE fails to encrypt a ballot, the voter can detect the mistake by comparing the verification code with the receipt.
- **Fairness.** No single encrypted ballot is decrypted. A voter makes her choice at her disposal.

– **Receipt-freeness.** The encrypted ballot is not generated by the voter. If the verification code and the receipt can be seen at the same time, the voter's choice can be deduced. But the verification code is destroyed. Just by reading the receipt we can see the probability for each candidate is equal. So the scheme is receipt-free.

Next we will show the voting scheme is secure against covert channels. The encrypted ballot set is generated by the authority. All the computation performed by the DRE is deterministic. There is no way for the DRE to encode secret information into the randomness. The voter is forced to generate her challenge string in a random manner just before she enters the booth. The DRE cannot distinguish the coerced voters from average voters by recognizing the challenges.

The voting scheme is voter verifiable. The voter can check that the DRE casts a ballot as intended and that the voting authority generates encrypted ballots correctly. If the encrypted ballot is inconsistent with the voter's choice, the DRE will fail to open the randomness in the matrix BSM with the probability $1-1/2^k$. It is easy for the voter to detect the mistake.

In real life thousands of DREs may be deployed at polling stations in a political election, and DREs may be manufactured by different vendors. It is hard to examine all the voting machines thoroughly. In the proposed voting scheme the generation of the encrypted ballots is centralized so the process is ease to monitor. The DRE is inspected by the voter. The overhead of malicious code test is decreased.

6 Conclusion

In this paper an electronic voting scheme is presented. Especially, the DRE is designed to achieve tamper-evidence and the attacks from covert channels are prevented. The voter verifies that the DRE uses pre-defined randomness by examining a destroyable verification code. Although the voting scheme is based on cryptography, it is unnecessary for the voter to do cryptographic computation. Only the operation of string comparison is performed by the voter. The complicated cryptographic computation can be carried out by any third party outside the booth. Although a voter is given a receipt to verify if the ballot is cast as intended, the receipt cannot be used to convince others which candidate the voter chooses. The vote-buying and coercion are thwarted.

Acknowledgement

We are grateful to the anonymous referees for their invaluable suggestions.

References

1. Neff, C.A.: Practical high certainty intent verification for encrypted votes (2004), http://votehere.com/old/vhti/documentation/vsv-2.0.3638.pdf

2. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
3. Choi, J.Y., Golle, P., Jakobsson, M.: Auditable privacy: on tamper-evident mix networks. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 126–141. Springer, Heidelberg (2006)
4. Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authourity Election Scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
5. ElGamal, T.: A Public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
6. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Class. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–239. Springer, Heidelberg (1999)
7. Pedersen, T.P.: A threshold cryptosystem without a trusted third party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
8. Fouque, P.A., Poupard, G., Stern, J.: Sharing Decryption in the Context of Voting or Lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
9. Damgard, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)

Appendix A: A Concrete Example

To explain the voting scheme in detail, we give a simple example here. Assume that there are two candidate and $k = 2$. The candidates are encoded as A_1, A_2. The voting authority works as follows:

1. The voting authority generates the encrypted ballot set $BS = E(A_1; r_1)$, $E(A_2; r_2)$, $r_1, r_2 \in R$.
2. The voting authority creates an id for the set BS.
3. The voting authority computes the matrix

$$BSM = \begin{bmatrix} (E(A_1/2; r_{1,0}^{(1)}), E(A_1/2; r_{1,1}^{(1)})) & (E(A_1/2; r_{1,0}^{(2)}), E(A_1/2; r_{1,1}^{(2)})) \\ (E(A_2/2; r_{2,0}^{(1)}), E(A_2/2; r_{2,1}^{(1)})) & (E(A_2/2; r_{2,0}^{(2)}), E(A_2/2; r_{2,1}^{(2)})) \end{bmatrix}$$

4. The voting authority computes the matrix

$$HBSM = \begin{bmatrix} (H(E(A_1/2; r_{1,0}^{(1)})), H(E(A_1/2; r_{1,1}^{(1)}))) & (H(E(A_1/2; r_{1,0}^{(2)})), H(E(A_1/2; r_{1,1}^{(2)}))) \\ (H(E(A_2/2; r_{2,0}^{(1)})), H(E(A_2/2; r_{2,1}^{(1)}))) & (H(E(A_2/2; r_{2,0}^{(2)})), H(E(A_2/2; r_{2,1}^{(2)}))) \end{bmatrix}$$

The matrix $HBSM$ is printed on the receipt as the verification code.

Assume the voter chooses the candidate A_2. The DRE prints $E(A_2; r_2)$ on the receipt. If the voter uses the challenge bit string "10", the randomness revealed is $\begin{bmatrix} r_{1,1}^{(1)} & r_{1,0}^{(2)} \\ r_{2,1}^{(1)} & r_{2,0}^{(2)} \end{bmatrix}$. The DRE can compute the matrix

$$BSM = \begin{bmatrix} (E(A_1;r_1) \oslash E(A_1/2;r_{1,1}^{(1)}), E(A_1/2;r_{1,1}^{(1)})) & (E(A_1/2;r_{1,0}^{(2)}), E(A_1;r_1) \oslash E(A_1/2;r_{1,0}^{(2)})) \\ (E(A_2;r_2) \oslash E(A_2/2;r_{2,1}^{(1)}), E(A_2/2;r_{2,1}^{(1)})) & (E(A_2/2;r_{2,0}^{(2)}), E(A_2;r_2) \oslash E(A_2/2;r_{2,0}^{(2)})) \end{bmatrix}$$

The DRE modifies some values in BSM to obtain the matrix

$$BSM' = \begin{bmatrix} (E(A_2;r_2) \oslash E(A_1/2;r_{1,1}^{(1)}), E(A_1/2;r_{1,1}^{(1)})) & (E(A_1/2;r_{1,0}^{(2)}), E(A_2;r_2) \oslash E(A_1/2;r_{1,0}^{(2)})) \\ (E(A_2;r_2) \oslash E(A_2/2;r_{2,1}^{(1)}), E(A_2/2;r_{2,1}^{(1)})) & (E(A_2/2;r_{2,0}^{(2)}), E(A_2;r_2) \oslash E(A_2/2;r_{2,0}^{(2)})) \end{bmatrix}$$

The DRE computes the matrix

$$HBSM' = \begin{bmatrix} (H(E(A_2;r_2) \oslash E(A_1/2;r_{1,1}^{(1)})), H(E(A_1/2;r_{1,1}^{(1)}))) & (H(E(A_1/2;r_{1,0}^{(2)})), H(E(A_2;r_2) \oslash E(A_1/2;r_{1,0}^{(2)}))) \\ (H(E(A_2;r_2) \oslash E(A_2/2;r_{2,1}^{(1)})), H(E(A_2/2;r_{2,1}^{(1)}))) & (H(E(A_2/2;r_{2,0}^{(2)})), H(E(A_2;r_2) \oslash E(A_2/2;r_{2,0}^{(2)}))) \end{bmatrix}$$

The voter compares the matrix $HBSM$ with the matrix $HBSM'$:

$$HBSM = \begin{bmatrix} (H(E(A_1/2;r_{1,0}^{(1)})), \underline{H(E(A_1/2;r_{1,1}^{(1)}))} & \underline{(H(E(A_1/2;r_{1,0}^{(2)}))}, H(E(A_1/2;r_{1,1}^{(2)}))) \\ \underline{(H(E(A_2/2;r_{2,0}^{(1)})), H(E(A_2/2;r_{2,1}^{(1)})))} & \underline{(H(E(A_2/2;r_{2,0}^{(2)})), H(E(A_2/2;r_{2,1}^{(2)})))} \end{bmatrix}$$

$$HBSM' = \begin{bmatrix} (H(E(A_2;r_2) \oslash E(A_1/2;r_{1,1}^{(1)})), \underline{H(E(A_1/2;r_{1,1}^{(1)}))} & \underline{(H(E(A_1/2;r_{1,0}^{(2)}))}, H(E(A_2;r_2) \oslash E(A_1/2;r_{1,0}^{(2)}))) \\ \underline{(H(E(A_2;r_2) \oslash E(A_2/2;r_{2,1}^{(1)})), H(E(A_2/2;r_{2,1}^{(1)})))} & \underline{(H(E(A_2/2;r_{2,0}^{(2)})), H(E(A_2;r_2) \oslash E(A_2/2;r_{2,0}^{(2)})))} \end{bmatrix}$$

The challenge string is:"10"

Look at the elements underlined in the two matrixes:

The second row of $HBSM$ is the same as that of $HBSM'$.

In the first row, the second element in the first pair and the first element in the second pair are the same.

If this is the case, the voter is convinced that the DRE uses the randomness given by the voting authority. There is no randomness generated by the DRE. If the assistant verifier finds that the matrix $HBSM'$ is correctly computed from the encryption $E(A_2;r_2)$ and the randomness $\begin{bmatrix} r_{1,1}^{(1)} & r_{1,0}^{(2)} \\ r_{2,1}^{(1)} & r_{2,0}^{(2)} \end{bmatrix}$, the voter is convinced that the DRE submits a vote on the candidate A_2 as her intent with probability 3/4.

Self-healing Key Distribution with Revocation and Resistance to the Collusion Attack in Wireless Sensor Networks

Wei Du and Mingxing He

School of Mathematics and Computer Engineering,
Xihua University, 610039, Chengdu, Sichuan, P.R.China
vivian_dv@163.com
he_mingxing64@yahoo.com.cn

Abstract. In this paper, we analyze an existing constant storage self-healing key distribution scheme in wireless sensor networks. Then, we show two attacks and propose a modified scheme to overcome the two flaws. At last, we propose a new self-healing key distribution scheme to improve the modified scheme. The most prominent properties of the new proposed scheme are as follows: achieving forward and backward secrecy and resisting to a collusion attack. So that a revoked user with the assistance of the newly joined users cannot get any information of group session keys which it is not entitled to get.

Keywords: Key distribution, self-healing, collusion attack, security, WSNs.

1 Introduction

Wireless sensor networks (WSN) consists of a large number sensor nodes with limited power, computation, storage and communication capabilities. Sensor nodes can be deployed in many different fields such as military, environment, health, home and other commercial areas etc. Security in WSN has six challenges [1]: (i) wireless nature of communication, (ii) resource limitation on sensor nodes, (iii) very large and dense WSN, (iv) lacking of fixed infrastructure, (v) unknown network topology prior to deployment, (vi) high risk of physical attacks to unattended sensors. Moreover, in some deployment scenarios sensor nodes need to operate under adversarial condition. Security solutions for such applications depend on existence of strong and efficient key distribution mechanisms.

With the widely development of wireless sensor networks, how to distribute group session keys for secure communication to large and dynamic groups over an unreliable, lossy networks becomes a increasing serious issue. Since the huge mobility of the users and the frequent loss of packets, it is important to guarantee the reliable transmission of updating group session keys to the authority users. A popular approach using self-healing mechanism enables a dynamic group of users to establish a group session key over an unreliable network and allow the non-revoked users who missed some previous group session keys using their personal

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 345–359, 2008.

key and the received broadcast message can still recover the missed group session keys without requesting additional transmission from the group manager. This approach reduces the communication overhead and lightens the heavy work of the group manager.

Our contributions in this paper are as follows. First, we analyze one existing scheme [10] and present two attacks that can be applied to [10]. Second, we modify the scheme to achieve the secrecy. Finally, we improve the modified scheme to propose a new scheme. The new scheme not only achieves forward and backward secrecy, but also resists to a collusion attack. The property of resistance to collusion attack is significant in the wireless sensor network, but few existing self-healing key distribution schemes with revocation in wireless sensor networks published have this property.

2 Related Work

Self-healing key distribution with revocation capability was firstly proposed by Staddon et al in [2]. It uses secret sharing [11] blinding the user's ability to recover from packet loss of the user's membership [10]. But the protocol given in [2] suffers from high overhead in both memory storage and communication complexity. Later on several other schemes have been proposed in [3], [4], [8], [9], [15] based on [2]. These protocols have been improved in terms of both memory storage and communication complexity. Liu et al. [3] applied a novel personal key distribution approach to the basic scheme of [2], made it more efficient. C.Blundo [8] first analysis the basic definition in [2] and [3], then argues that it is impossible for any scheme to achieve the security requirements proposed in [2] and [3], then proposes a new definition of self-healing key distribution and a different mechanism [5] for implementing the self-healing approach. C.Blundo in [7] also shows an attack on [2]'s first scheme. Hong et al.in [5] proposed self-healing key distribution schemes having less storage and communication complexity.

Recently, more and more schemes [10], [12], [13], [14] of self-healing key distribution scheme with revocation in wireless sensor networks are proposed. Most of them adopt the one-way hash functions to generate the group session keys, which significantly reduce the communication and storage overhead. R.Dutta in [10] using one way hash key chain function to proposed a constant storage self-healing key scheme. R.Dutta in [13] largely reduced the communication overhead by constructing two hash key chain functions. Unfortunately, however, none of them can resist to the collusion attack. So that a revoked user with the assistance of the newly joined users can get the information of keys which it is not entitled to get. The property of resistance to collusion attack is significant in the wireless sensor network. Because the sensor nodes are lack of the physical protection and are prone to be compromised by adversaries. It is still a challenge to construct a self-healing key distribution scheme with the property of resistance to collusion attack and having tolerant communication and storage overhead.

3 A Original Scheme and Two Attacks

In this section, we firstly recall the scheme given in [10]. Then we will show that the original scheme [10] has two flaws. One is that any user once becomes a member can recover the personal secret polynomial after receiving a single broadcast message. Therefore it can recover all the group session keys no matter whether it is a non-revoked user or not. The other one is that scheme [10] doesn't achieve the backward secrecy [10], so that a newly joined user can compute the previous group session keys with the knowledge of the future group session keys and his secret information.

3.1 Review of R.Dutta's Scheme

Assume that GM is a group manager, $U = \{U_1, U_2, ..., U_n\}$ is a set of n users of the network, $E_k(\cdot)$ and $D_k(\cdot)$ respectively, denotes an encryption and corresponding decryption function, which may be regarded as keyed permutations over F_q under $k \in F_q$(F_q denotes a finite field with order q and q is a large prime number($q > n$)), and f is a random one way permutation over F_q such that $f^i(u) \neq f^j(u)$ for all positive integers $i, j, i \neq j$ and $u \in F_q$,

Setup. First, the group manager (GM) randomly chooses a t-degree polynomial $\psi(x) = a_0 + a_1x + a_2x^2 + ... + a_tx^t \in F_q[x]$ and also chooses a random initial session identifier $sid_0 \in F_q$;

Then GM sends in a private way to $U_i(1 \leq i \leq n)$, sid_0 and $\psi(i)$ as U_i's secret;

At last, GM selects randomly a prime key $K_0 \in F_q$ that is kept secret to himself.

Broadcast in session $j(j > 1)$. GM computes the j-th session identifier $sid_j = f(sid_{j-1})$, chooses a random number $\beta_j \in F_q$ and computes the j-th group session key $K_j = E_{\beta_j}(K_{j-1})$;

let $R_j = \{U_{r_1}, ..., U_{r_{w_j}}\} \subseteq U$, $\mid R_j \mid = w_j \leq t$ be the set of all revoked users in session j and before session j and $r_1, ..., r_{w_j}$ be the ID of the revoked users, such that $\mid R_j \mid = w_j \leq t$. GM broadcasts the following broadcast message B_j:

$$B_j = R_j \cup \{\phi_j(x) = A_j(x)K_j + sid_j \cdot \psi(x)\} \cup \{E_{K_j}(\beta_1), ..., E_{K_j}(\beta_j)\} \quad (1)$$

Where $A_j(x) = (x - r_1)(x - r_2)...(x - r_{w_j})$. Here the polynomial $A_j(x)$ is called revocation polynomial and $\psi(x)$ performs the role of masking polynomial. Notice that each user $U_i \in U$ knows a single point, namely $\psi(i)$ on the polynomial $\psi(x)$.

Once a non-revoked user U_i recovers the current group session key K_j, it can recover the self-healing keys $\beta_1, ..., \beta_j$.

Group Session Key and Self-Healing Key Recovery in Session j. If a non-revoked user $U_i(1 \leq i \leq n)$ receives the broadcast message B_j, U_i first computes the session identifier $sid_j = f(sid_{j-1})$ and replaces the previous session

identifier sid_{j-1} by the current sid_j for $j > 1$(in case j=1, sid_1 is stored). Then U_i evaluates $\phi_j(x), \psi(x), A_j(x)$ at point i from Equation (1) to compute the group session key $K_j = (\phi_j(i) - sid_j \cdot \psi(i))/A_j(i)$, where $A_j(i) \neq 0$;

A revoked user who receives the broadcast message can recover neither the current group session key nor the self-healing keys. Because for any $U_i \in R_j$, $A_j(i) = 0$.

Add Group Members. When GM wants to add a new user starting from session j, GM chooses a unique identity $v \in \{1, 2, ..., n\}$ for U_v, which is never used before. Then GM gives the personal key $\psi(v)$ and sid_j to U_v through the secure communication channel between them.

3.2 Attack 1: Launched by Any User

Suppose $U_i \in U$ joins the group in session j so that U_i can get its personal secrets $\psi(i)$ and the session identifier sid_j from the group manager via the secure channel.

After receiving the j-th broadcast message B_j, U_i can compute the j-th group session key K_j by using Equation (1) at point i. For $A_j(i) \neq 0$, U_i can get $K_j = (\phi_j(i) - sid_j \cdot \psi(i))/A_j(i)$;

Then, U_i recovers the personal secret polynomial $\psi(x)$ easily by evaluating $\psi(x)$ from Equation (1) and $\psi(x) = (\phi_j(x) - A_j(x) \cdot K_j)/sid_j$. Because U_i knows K_j, sid_j and $A_j(x)$ can be determined by using the revoked users'IDs from the broadcast message B_j and $A_j(x) = (x - r_1)(x - r_2)...(x - r_{w_j})$.

After knowing the structure of personal secret polynomial $\psi(x)$, U_i can get all the group session keys no matter whether U_i is revoked or not. If U_i is revoked, it can inject false or forged user's ID (not really a user's ID in the networks) to get all the past and future group session keys easily without being detected. Because given $\phi_j(x), A_j(x)$ and $\psi(x)$, K_j is a unique value.

So it is dangerous for the group to have the personal secret polynomial $\psi(x)$ revealed, and it guarantees none secrecy at all and any user can compute any group session key K_j easily.

C.Blundo in [8] claims that it is impossible for any protocol to achieve the security requirement that stated in [2] and [3] and this is the root of the flaw existing in scheme [10]. C.Blundo in [8] also points out that there is none protocol satisfying the security requirements in [2] and [3] unless the entropy of the group session key K_j is 0. That is to say, K_j is already known to any user, no matter what the broadcast message or the user's personal secret is.

This attack, however, does not work effectively in [2] and [3]. Because unlike [10], [2] and [3] use different secret polynomials in different sessions, and the used secret polynomials will not be reused any longer. Therefore, being revealing one or more used secret polynomial to users has no effect on the future unused secret polynomials, so it has no effect on the future group session keys.

3.3 Attack 2: Launched by a Newly Joined User

Suppose $U_v \in U$ joins the group in session $j(j > 1)$. Now we will show that U_v can compute the group session key in session $(j-1)$ with the broadcast message B_j.

After receiving the j-th group session key distribution broadcast message B_j, U_v uses Equation (1) to evaluate $\psi(v), \phi_j(v)$ and $A_j(x)$. Thus U_v can computes the current group session key K_j, since $K_j = (\phi_j(v) - sid_j \cdot \psi(v))/A_j(v)$ and $A_j(v) \neq 0$;

Then U_v decrypts $\{E_{K_j}(\beta_1), ..., E_{K_j}(\beta_j)\}$ using K_j respectively to get all the self-healing keys $\beta_1, ..., \beta_j$;

As $E_K(\cdot)$ and $D_K(\cdot)$ respectively denotes an encryption and corresponding decryption function with $D_K(E_K(M)) = M$ and $K_j = E_{\beta_j}(K_{j-1})$, U_v can decrypt $E_{\beta_j}(K_{j-1})$ by using β_j and K_j to get K_{j-1} which U_v is not entitled to get. More seriously, U_v can deduce all the previous group session keys, even the prime key K_0 as follows: $K_{j-1} = D_{\beta_j}(K_j) = D_{\beta_j}(E_{\beta_j}(K_{j-1}))$,..., $K_0 = D_{\beta_1}(K_1) = D_{\beta_1}(E_{\beta_1}(K_0))$ and K_0 is chosen by GM and should not be known to anyone except GM.

This attack sponsored by any newly joined user points out that the scheme above doesn't achieve the t-wise backward secrecy [10] which it claims to have.

3.4 A Modified Scheme

We assume that m denotes the maximum of the session numbers. First, we use m t-degree polynomials, say $f_1(x), ..., f_m(x) \in F_q[x]$ instead of one t-degree polynomial $\psi(x)$ in [10]. So the secret of any user U_i is $s_i = \{f_1(i), ..., f_m(i)\}$. After doing such modification, in session j, all the users in the group have no knowledge of the future secret polynomials $f_i(x)(j < i \leq m)$ until they recover the corresponding group session key K_j.

Second, we use one-way hash function to generate the new group session key instead of using encryption function. More precisely, let $H_1(\cdot)$ denote the one-way hash function, and using $K_j = H_1(K_{j-1}, \beta_j)$ instead of $K_j = E_{\beta_j}(K_{j-1})$. After doing such a modification, the scheme meets the t-wise backward secrecy that any user who is newly joined the group cannot compute the previous group session keys. Because $H_1(\cdot)$ is a one-way hash function, given K_j and β_j, it is computationally infeasible to calculate K_{j-1}. Besides, we remove the session identifier sid_j from the scheme.

Scheme 1: A Modified Self-healing Key Distribution with Revocation Capability

Assume $U = \{U_1, U_2, ..., U_n\}$, a set of universal users in wireless sensor networks.

Setup. First, the group manager(GM) randomly chooses m polynomials of degree t, say $f_1(x), ..., f_m(x) \in F_q[x]$ and randomly chooses m numbers $\beta_1, ..., \beta_m \in F_q$;

Second, GM sends in a private way, for $i = 1, ..., n$, to user U_i as personal key $s_i = \{f_1(i), ..., f_m(i)\}$;

Third, GM chooses a prime key seed K_0 and kept it as his own secret;

Then, GM generates the group session keys for session 1 to session m by using one way hash function $H_1(\cdot)$ as follows:

$$K_j = H_1(K_{j-1}, \beta_j)_{j=1,...,m} \tag{2}$$

Broadcast: for $1 \leq j \leq m$. Let $R_j = \{U_{r_1}, ..., U_{r_{w_j}}\} \subseteq U$ be a set of revoked users for session j and before session j and $r_1, ..., r_{w_j}$ be the ID of the revoked users, such that $\mid R_j \mid = w_j \leq t$.

In session j, GM broadcasts the following message:

$$B_j = R_j \cup \{z_j(x) = A_j(x)K_j + f_j(x)\} \cup \{E_{K_j}(\beta_1), ..., E_{K_j}(\beta_j)\} \tag{3}$$

Where $A_j(x) = (x - r_1)(x - r_2)...(x - r_{w_j})$. Here the polynomial $A_j(x)$ is called revocation polynomial and $f_j(x)$ performs the role of masking polynomial.

Group Session Key and Self-Healing Key Recovery in Session j. If a non-revoked user $U_i (1 \leq i \leq n)$ receives the broadcast message B_j, U_i evaluates $z_j(x), A_j(x)$ at point i from Equation (3) to compute the group session key $K_j = (z_j(i) - f_j(i))/A_j(i)$, where $A_j(i) \neq 0$;

A revoked user who receives the broadcast message can recover neither the current group session key nor the self-healing keys. Because for any $U_i \in R_j$, $A_j(i) = 0$.

Once a non-revoked user U_i recovers the current group session key K_j, it can recover the self-healing keys $\beta_1, ..., \beta_j$.

Add Group Members. When GM wants to add a new user starting from session j, GM chooses a unique identity $v \in \{1, 2, ..., n\}$ for U_v, which is never used before. Then GM gives the personal key $s_v = \{f_j(v), ..., f_m(v)\}$ to U_v through the secure communication channel between them. Notice that if a user is revoked in session j, it must be revoked in all future sessions.

We can see that the modified scheme guarantees the security requirements in [10] and the communication overhead is $(t + 1 + j) \log q$ bits which equals to that of [10]. But as the original scheme [10], the modified scheme is also not resistant to the collusion attack. The storage overhead of the modified scheme is $(m - j + 1) \log q$ which is not as constant as that of [10]. But according to the Theorem 5.2 in [5], in an unconditionally secure self-healing key distribution scheme, every user who belongs to G_j(the non-revoked users in session j) has to store a personal key of at least $(m - j + 1) \log q$ bits, which comes from the personal key that each group user has to keep.So it is determined by the number of masking polynomials [3]. Thus the storage overhead of the modified scheme is one of the optimal one.

4 A New Self-healing Key Distribution with Revocation and Resistance to the Collusion Attack

In this section, we will improve the modified scheme(Scheme 1) to propose a new self-healing key distribution scheme which is resistant to the collusion attack.

Instead of allowing all non-revoked users to recover both the current self-healing key β_j and the previous self-healing keys $\beta_1, ..., \beta_{j-1}$ as given in Scheme 1, the new proposed scheme only allows users to recover the self-healing keys that were used after the user joined the group. That is to say, a new user who joins the group in session j is only allowed to recover the current self-healing key β_j, while an old non-revoked user is allowed to recover both the current self-healing key β_j and the previous self-healing keys since the time it joined the group.

Table 1. Notations

U	: the set of universal users in wireless sensor networks
U_i	: user in U
GM	: the group manager
n	: the total number of users in U and n be a positive integer
m	: the maximum of the session numbers and m be a positive integer
t	: the maximum number of the compromised users and t be a positive integer
F_q	: a finite field with order q and q is a large prime number($q > n$)
r_j	: a random number which GM selects for the user who joins in session j and $r_j \in F_q$, for any $i \neq j$, $r_j \neq r_i$
α_j	: a random number which GM selects for the user who joins in session j and $\alpha_j \in F_q$, for any $i \neq j$, $\alpha_j \neq \alpha_i$
s_i	: the personal secret of user U_i
$f_j(x)$	: user's personal secret polynomial for session j, which is randomly chosen by GM
B_j	: the broadcast message generated by GM during session j
K_j	: the group session key in session j generated by GM and $K_j = K_{j-1}^j$
K_j^0	: the seed of the hash key chain in session j
$K_j^{j'-1}$	: the key from the key chain generated by the GM in session $j$and only a user joined the group in session $j'(1 \leq j' < j)$ is allowed to get it from the j-th broadcast
$G_j^{j'}$	: the masking key of the key $K_j^{j'-1}(1 \leq j' < j)$ from the hash key chain
β_j	: the self-healing key randomly chosen by the GM for session j used to recover the group session key $K_j = K_j^{j-1}$ in session j
R_j	: the set of revoked users in and before session j and $R_j \subseteq U$
H_1	: a one-way hash function used to generate the seed of the new group session key and $H_1 : \{0,1\}^* \rightarrow F_q$
$H:$	: a one-way hash function and $H : \{0,1\}^* \rightarrow F_q$
$E_k(\cdot)$	: an encryption function
$D_k(\cdot)$	: a corresponding decryption function
$A_j(x)$	: the revoked polynomial in session j

Now, for clarity, we briefly list some of the symbols used throughout our new proposed scheme in Table 1. For details, we refer to [10].

4.1 Security Model

Our scheme should provide the following security.

Definition 1: (self-healing key distribution with *t*-revocation capability [10]) A key distribution scheme is a self-healing key distribution with t-revocation capability if the following conditions are true:

a) For each non-revoked user in session j, the group session key K_j is determined by the broadcast message B_j and the user's own secret;

b) What the non-revoked users learn from B_j or their own personal secret alone cannot determine the group session key K_j;

c) (t-revocation capability) For each session j, let R_j denotes a set of revoked users in and before session j, such that $\mid R_j \mid \leq t$, the group manager can generate a broadcast message B_j such that all the revoked users in R_j cannot recover the group session key K_j;

d) (self-healing property) Every U_i joins in or before session j_1 and not revoked before session $j_2 (j_1 < j_2)$ from broadcast B_{j_1} and B_{j_2}, where $1 \leq j_1 < j_2$, can recover all the keys $K_j (j = j_1, ..., j_2)$.

Definition 2: (*t*-wise forward secrecy [10]) A key distribution scheme guarantees forward secrecy if for any set $R \subseteq U$, where $\mid R \mid \leq t$, and all $U_i \in R$ are revoked in session j, the members in R together cannot get any information about K_j, even with the knowledge of group session keys before session j.

Definition 3: (*t*-wise backward secrecy [10]) A group session key distribution guarantees backward secrecy if for any set J, where $\mid J \mid \leq t$, and all $U_i \in J$ join after session j, the members in J together cannot get any information about K_j, even with the knowledge of group session keys after session j.

Definition 4: (resistance to the collusion attack [5]) Let $B \subseteq R_r \cup R_{r-1} \cup ... \cup R_1$ be a coalition of users removed before session r and let $C \subseteq J_s \cup J_{s+1} \cup ...$ be a coalition of users who join the group from session s. Let $\mid B \cup C \mid \leq t$ and $B \cup C \subseteq U$. Then, such a coalition does not get any information about keys K_j, for any $r \leq j < s$.

4.2 A New Self-healing Key Distribution with Revocation and Resistance to the Collusion Attack

Scheme 2: A New Self-healing Key Distribution with Revocation and Resistance to the Collusion Attack

Assume $U = \{U_1, U_2, ..., U_n\}$, a set of universal users in wireless sensor networks. Fig 1 illustrate how the Scheme 2 works in session j. This scheme consists

of four phases: Phase 1 - Setup, Phase 2 - Broadcast, Phase 3 - Group Session Key and Self-Healing Key Recovery in Session j, Phase 4 - Add Group Members.

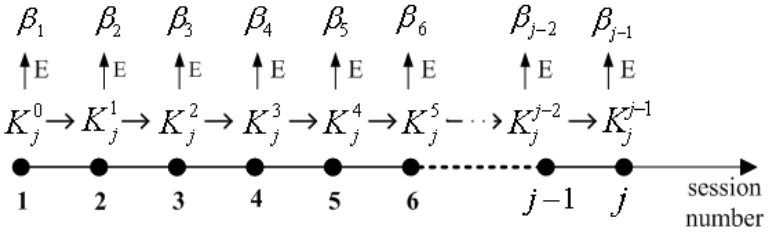

Fig. 1. The Group Session Key And Self-haling Keys in Session j

Phase 1 - Setup. GM first randomly chooses m polynomials of degree t, say $f_1(x), ..., f_m(x) \in F_q[x]$ and randomly chooses m numbers $r_1, ..., r_m \in F_q$ and keep them as his secret.

Second, GM randomly chooses numbers $\alpha_1, ..., \alpha_m \in F_q$.

Third, GM sends in a private way, for $i = 1, ..., n$, to user U_i as personal key $s_i = \{\alpha_{j'}; r_{j'} \cdot f_{j'}(i), ..., r_{j'} \cdot f_m(i)\}$($j'$ denotes the session number which the user joins the group and $\alpha_{j'} \in \{\alpha_1, ..., \alpha_m\}$, $r_{j'} \in \{r_1, ..., r_m\}$). Specifically, a user U_v who joins in session 1, will receive $s_v = \{\alpha_1; r_1 \cdot f_1(v), ..., r_1 \cdot f_m(v)\}$ while a user U_k who joins in session j will receive $s_k = \{\alpha_j; r_j \cdot f_j(k), ..., r_j \cdot f_m(k)\}$. Notice that none of the user has the knowledge of a single value of $r_{j'}$ or $f_j(i)$, user only knows $r_{j'} \cdot f_j(i)(j = 1, ..., m)$.

GM then chooses a prime key seeds $K_0 \in F_q$ and kept it as his own secret;

GM randomly chooses m number $\beta_1, ..., \beta_m \in F_q$;

GM computes m key seeds and the corresponding m key chains by using two one-way hash functions $H_1(\cdot)$ and $H(\cdot)$(For convenience of description, we use two hash functions, but in fact they are the same.). For session 1: the key seed is $K_1^0 = H_1(K_0, \beta_1)$ and key chain is $\{K_1^0\}$; for session 2: the key seed is $K_2^0 = H_1(K_1, \beta_2)$ and key chain is $\{K_2^0, K_2^1\}$; for session $j(1 \le j \le m)$, the key seed is:

$$K_j^0 = H_1(K_{j-1}, \beta_j) \tag{4}$$

the key chain for session $j(1 \le j \le m)$ is:

$$\{K_j^0, K_j^1, K_j^2 ..., K_j^{j-1}\}_{j=1,...,m} \tag{5}$$

where $K_j^1 = H(K_j^0), K_j^2 = H(K_j^1) = H^2(K_j^0), ..., K_j = K_j^{j-1} = H(K_j^{j-2}) = ... = H^{j-1}(K_j^0)$ and $H^i(\cdot)$ denotes applying i times hash operation. $K_j = K_j^{j-1}$ is the group session key in session j.

Thus we can see that the size of each key chain equals to the session number $j(1 \le j \le m)$. Specifically, in session 1, the size of key chain is 1; in session j, the size of key chain is j and in session m, the size of key chain is m.

Phase 2 - Broadcast (for $1 \leq j \leq m$). Assume $R_j = \{U_{r_1}, ..., U_{r_{w_j}}\} \subseteq U$ be a set of revoked user in and before session j, and $r_1, ..., r_{w_j}$ be the IDs of the revoked users, such that $\mid R_j \mid = w_j \leq t$

GM then generates a masking key sequence $\{G_j^1, G_j^2, ..., G_j^{j-1}, G_j^j\}$ of size j for session j by applying XOR on both $\alpha_{j'}$ and every key from the one-way hash key chain, where

$$G_j^{j'} = K_j^{j'-1} \oplus \alpha_{j'} \qquad (j = 1, ..., m; j' = 1, ..., j) \tag{6}$$

$\alpha_{j'}$ denotes the secret of the users who join the group in session j' and $\oplus$ denotes XOR operation.

In session j, GM broadcasts the following message:

$$\begin{aligned} B_j = R_j \cup \{z_j^{j'}(x) = A_j(x)G_j^{j'} + r_{j'} \cdot f_j(x)\}_{j'=1,...,j} \\ \cup \{E_{K_j^0}(\beta_1), E_{K_j^1}(\beta_2), ..., E_{K_j^{j-1}}(\beta_j)\} \end{aligned} \tag{7}$$

Where $A_j(x) = (x - r_1)(x - r_2)...(x - r_{w_j})$ Here the polynomial $A_j(x)$ is called revocation polynomial and $r_{j'} \cdot f_j(x)$ performs the role of masking polynomial.

Phase 3 - Group Session Key and Self-Healing Key Recovery in Session j. If a non-revoked user $U_i (1 \leq i \leq n)$ who joins the group in session $j' (1 \leq j' < j)$ receives the broadcast message B_j, the user U_i can recover the group session key and self-healing key as follows:

U_i evaluates $z_j^{j'}(x), A_j(x)$ at point i from Equation (7) to compute the masking key $G_j^{j'} = (z_j^{j'}(i) - r_{j'} \cdot f_j(i))/A_j(i)$, where $A_j(i) \neq 0$;

U_i computes $K_j^{j'-1}$ by applying XOR both on $G_j^{j'}$ and $\alpha_{j'}$ as given below

$$K_j^{j'-1} = G_j^{j'} \oplus \alpha_{j'} \qquad (j = 1, ..., m; j' = 1, ..., j) \tag{8}$$

$\alpha_{j'}$ denotes the secret of the users who join the group in session j' and $\oplus$ denotes XOR operation.

U_i computes all the future keys $K_j^i (j' - 1 < i \leq j - 1)$ from the same key chain by using one-way hash function $H(\cdot)$(see Equation (5)) and $K_j = K_j^{j-1} = H^{j-j'}(K_j^{j'-1})$ is the current group session key;

U_i also uses Equation (7) to decrypt $\{E_{K_j^{j'-1}}(\beta_{j'})\}_{j'=1,...,j}$ by using the corresponding keys in the current key chain to get the corresponding self-healing keys $\{\beta_{j'})\}_{j'=1,...,j}$;

A revoked user who receives the broadcast message can recover neither the current group session key nor the self-healing keys. Because for any $U_i \in R_j$, $A_j(i) = 0$.

Phase 4 - Add Group Members. When GM wants to add a new user starting from session j, it chooses a unique identity $v \in \{1, 2, ..., n\}$ for U_v, which is never

used before. Then GM gives the personal key $s_v = \{\alpha_j; r_j \cdot f_j(v), ..., r_j \cdot f_m(v)\}$ to U_v through the secure communication channel between them. Notice that if a user is revoked in session j, it means that it must be revoked in all future sessions.

5 Analysis

In this section, we will show that the proposed new scheme satisfies the requirements of self-healing key distribution with revocation capacity. Then we will show that the proposed new scheme also guarantees forward security, backward security and resistance to the collusion attack.

5.1 Self-healing Property

Assume $U_i \in U$ joins the group in session j_1 and revoked after session $j_2(j_1 < j_2)$. U_i receives the broadcast message $B_{j_1}(1 \leq j_1)$ and $B_{j_2}(1 \leq j_1 < j_2)$, but missed the broadcast message $B_j(j_1 < j < j_2)$. We now show that U_i can still recover all the missed group session keys $K_j = K_j^{j-1}(j_1 < j < j_2)$ as follows to explain the self-healing property.

(1) After receiving broadcast message B_{j_1} and B_{j_2}, U_i can compute the masking key $G_{j_1}^{j_1}$ and $G_{j_2}^{j_1}$ by using Equation (7) as described in Phase 3 in section 4.2;

(2) U_i can compute the group session key $K_{j_1} = K_{j_1}^{j_1-1}$ for session j_1 and the key $K_{j_2}^{j_1-1}$ in session j_2 by Equation (8) as described as in Phase 3 in section 4.2;

(3) U_i uses $K_{j_2}^{j_1-1}$ to generates part of j_2-th one-way hash key chain $\{K_{j_2}^{j_1}, K_{j_2}^{j_1+1}, ..., K_{j_2}^{j_2-1}\}$ which U_i is entitled to get by using the hash function(see Equation (5)) and $K_{j_2} = K_{j_2}^{j_2-1}$ is the group session key for session j_2;

(4) U_i decrypts the $\{E_{K_{j_2}^{j_1-1}}(\beta_{j_1}), ..., E_{K_{j_2}^{j_2-1}}(\beta_{j_2})\}$ by using $\{K_{j_2}^{j_1-1}, ..., K_{j_2}^{j_2-1}\}$ respectively to get the self-healing keys β_{j_1},..., β_{j_2} between session j_1 and session j_2;

(5) U_i gets all the missed group session keys seeds $K_j^0(j_1 < j < j_2)$ by using Equation (4) and (5);

(6) Finally, U_i can recover all the missed group session keys $K_j = K_j^{j-1}$, for all $j = j_1 + 1, j_1 + 2, ..., j_2 - 1$ by using doing step (5) repeatedly.

Thus the proposed new scheme guarantees the self-healing property.

5.2 Forward Secrecy

Consider $R \subseteq U$, where $\mid R \mid \leq t$, and $U_i \in R$ revoked in session j. Then neither a single user in R nor a set of users in R can get any information about the group session key $K_j(j = j, ..., m)$, even with the previous group session keys and their personal secrets. Because the way that users in R to get the j-th group session key is either getting the corresponding self-healing keys which are all randomly

chosen or recovering one of the personal secret polynomials $r_{j'} f_j(x)(j' = 1, ..., j)$, but we will show that none of the two methods can be obtained.

First, each self-healing key is encrypted by the corresponding group session key $K_j = K_j^{j-1}$ and the corresponding group session key is masked by the corresponding masking key $G_j^{j'}$. But users in R cannot evaluate any of the masking keys $G_j^{j'}$, because for any $U_i \in R_j$, $A_j(i) = 0$. Therefore, the users in R cannot know the future group session keys by getting the values of the self-healing keys unless they guess the corresponding group session key $K_j^{j'-1}$ rightly.

Second, we will show that it is also impossible for users in R to recover any of the personal secret polynomials $r_{j'} \cdot f_j(x)(j' = 1, ..., j)$ in session j. Users in R have to get at least $t + 1$ points on each polynomial $r_{j'} \cdot f_j(x)$, but $\mid R \mid \leq t$ and the maximum number of points they can get on each polynomial $r_{j'} \cdot f_j(x)$ is t. So it is impossible for users in R to recover any of the polynomial $r_{j'} \cdot f_j(x)$.

Therefore, it is impossible for users to recover the future group session keys. Thus our proposed scheme is forward secure.

5.3 Backward Secrecy

Consider $J \subseteq U$, where $\mid J \mid \leq t$, and $U_v \in J$ joined after session j. Then neither a single user in J nor a set of users in J cannot get any information about any previous session key K_j even with all the future session keys and their personal secret keys.

First, from the broadcast B_{j+1}, users in J can only get the current group session key K_{j+1} which is the last key in the one-way hash key chain, therefore user in J can only decrypt the current self-healing key β_{j+1}. As $K_{j+1} = K_{j+1}^j = H(K_{j+1}^{j-1}) = ... = H^{j+1}(K_{j+1}^0)$ and $K_{j+1}^0 = H_1((K_j, \beta_{j+1})$, where $H_1(\cdot)$ and $H(\cdot)$ are two one-way hash functions, it is computationally infeasible for any user in to compute the previous keys in the key chain, so it is impossible for them to compute the previous group session keys.

Second, we will show that it is also impossible for users in J to recover the personal secret polynomials $r_j \cdot f_j(x)$. Users in J have to get at least $t+1$ points on the polynomial $r_j \cdot f_j(x)$, but J join after session j, so all the user in J do not have any secret keys used in session j. So it is impossible for users in R adopting this method to recover the polynomial $r_j \cdot f_j(x)$.

Hence the users in J cannot get the previous group session keys and the proposed scheme is backward secure.

5.4 Resistance to a Collusion Attack

Consider a set of users $B \subseteq R_{j_1} \cup R_{j_1-1} \cup ... \cup R_1$ who revoked in or before session j_1 and a set of users $C \subseteq J_{j_2} \cup J_{j_2+1} \cup ...$ who join in session $j_2 (B \cup C \subseteq U)$. We will show that if B and C are disjoint and $\mid B \cup C \mid \leq t$, users in B colluding with the users in C, are unable to recover $K_j = K_j^{j-1} (j_1 \leq j < j_2)$ from the broadcast message B_{j_1} and B_{j_2}.

In order to recover $K_j = K_j^{j-1}(j_1 \leq j < j_2)$, $B \cup C$ must recover all the self-healing keys between β_{j_1} and β_{j_2}. However, both B and C don't have the right to access these self-healing keys, for they can't evaluate the keys in the key chain, thus they can't use these keys to decrypt the self-healing keys $\beta_j(j_1 \leq j < j_2)$.

On one hand, the newly joined user from C can only get the group session key $K_j = K_j^{j-1}(j \geq j_2)$ and corresponding self-healing key $\beta_j(j \geq j_2)$(the same as described in section 5.3). On the other hand, users who revoked before session j_1 can only get the group session keys $K_j = K_j^{j-1}(j < j_1)$ and $\beta_j(j < j < j_1)$ because for any $U_i \in B$, $A_j(i) = 0(j \geq j_1)$. As a result, all the information they have putting together does not contain all the self-healing keys between β_{j_1} and β_{j_2-1}, so that they can't get the group session keys $K_j = K_j^{j-1}(j_1 \leq j < j_2)$. It is easy to see that the proposed scheme resists to such collusion attack.

6 Efficiency Comparisons with Some Previous Schemes

In this section, we will compare our schemes with other previous ones in different aspects such as communication complexity, storage overhead, forward security, backward security and soundness to collusion attack. From table 2, we can see clearly that the scheme 2 achieves some advantages such as t-wise forward secrecy, backward secrecy and resistance to collusion attack, all of which are the fundamental ingredients in the security requirements.

Table 2. Performance Comparisons

Scheme	Storage Overhead	Communication Overhead	Group Session Key Secrecy	Forward Secrecy	Backward Secrecy	Resistance to Collusion Attack
C3 of [2]	$(m-j+1)^2 \log q$	$(mt^2+2mt+m+t)\log q$	Yes	Yes	Yes	Yes
S3 of [3]	$(m-j+1)\log q$	$(2tj+j)\log q$	Yes	Yes	Yes	Yes
S3 of [5]	$2(m-j+1)\log q$	$[(m+j+1)t+(m+1)]\log q$	Yes	Yes	Yes	Yes
[10]	$3\log q$	$(t+1+j)\log q$	No	No	No	No
[13]	$(m-j+1)\log q$	$(t+1)\log q$	Yes	Yes	Yes	No
Scheme 1	$(m-j+1)\log q$	$(t+1+j)\log q$	Yes	Yes	Yes	No
Scheme 2	$(m-j+2)\log q$	$[(t+1)j+j]\log q$	Yes	Yes	Yes	Yes

The storage requirement in schemes 1 is $(m-j+1)\log q$ bits which is not constant, But respect to Theorem 5.2 in [5], the storage overhead of the modified scheme, in some sense, is optimal.

As in the original scheme [10], the communication overhead in scheme 1 is $(t + 1 + j) \log q$ bits, which significantly reduce the communication overhead comparing to some of the previous schemes. However, as the original scheme [10], scheme 1 does not resist to the collusion attack.

In the proposed scheme 2, the broadcast message B_j consist of a set of revoked users R_j, j t-degree polynomials $\{z_j^{j'}\}_{j'=1,\ldots,j}$, a sequence $\{E_{K_j^{j'-1}}(\beta_{j'})\}_{j'=1,\ldots,j}$ (see Equation (7)). One can ignore the communication overhead for the broadcast of the set R_j, because the member IDs can be selected from a small finite field [3]. So the communication overhead of scheme 2 is $[(t + 1)j + j] \log q$ bits.

The communication overhead of scheme 2 is larger than [10]'s results from the different structure of the broadcast message B_j. First, we adopt j t-degree polynomials $\{z_j^{j'}\}_{j'=1,\ldots,j}$ instead of using only one t-degree polynomial in session j. Second, we use different group session keys to encrypt the corresponding self-healing keys instead of just using the same current group session key to encrypt all the self-healing keys. This structure makes it impossible for any old user and any new one to launch a collusion attack to attain the group session keys which they are not entitled to get. Because this structure limits the right of the users to get the self-healing keys. Without the right self-healing keys, no one can recover the corresponding group session keys. In order to guarantee this property, the proposed scheme 2, to some extent, sacrifices the communication overhead. Fortunately, however, the communication complexity of scheme 2 is acceptable comparing to other previous schemes. We consider this sacrifice is worthwhile, because the sensor nodes are vulnerable to compromise and an adversary can easily launch a collusion attack by compromising an old node and a new one.

7 Conclusions

In this paper, we analyze and present two flaws in scheme [10], then we propose a modified scheme(Scheme 1) to overcome the two flaws. Then we improve the modified scheme to propose a new self-healing key distribution scheme(Scheme 2). Scheme 2 not only achieves forward secrecy and backward secrecy, but also resists to the collusion attack. It is significant to have the ability to resist to the collusion attack in the wireless sensor networks. Because the sensor nodes are lack of the physical protection and are prone to be compromised by adversaries.

Our future work would be mainly focused on reducing the storage and communication overhead. In addition, we seek way to expand the users' life time.

Acknowledgments. The work is supported by the National Natural Science Foundation of China (Grant no. 60773035); the Key Projects Foundation of Ministry of Education of China (Grant no. 205136). The Foundation of Science and Technology Bureau of Sichuan Province, China (Grant no. 05JY029-131).

References

1. Camtepe, S., Yener, B.: Key Distribution Mechanisms for Wireless Sensor Networks: A Survey. Technical Report, TR-05-07, Rensselaer Polytechnic Institute (2005)
2. Staddon, J., Miner, S., Franklin, M., Balfanz, D., Malkin, M., Dean, D.: Self-healing Key Distribution with Revocation. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 241–257 (2002)
3. Liu, D., Ning, P., Sun, K.: Efficient Self-healing Group Key Distribution with Revocation Capability. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, New York, pp. 2003–231 (2003)
4. More, S., Malkin, M., Staddon, J.: Sliding-window Self-healing Key Distribution with Revocation. In: ACM Workshop on Survivable and Self-regenerative Systems (2003)
5. Blundo, C., Darco, P., Santis, A.D., Listo, M.: Design of Self-healing Key Distribution Schemes. Des. Codes Cryptography (2004)
6. Blundo, C.: Randomness in Self-Healing Key Distribution Schemes. Theory and Practice in Information-Theoretic Security, 80–84 (2005)
7. Blundo, C., D'Arco, P., Listo, M.: A Flaw in a Self-Healing Key Distribution Scheme. In: Proceedings of Information Theory Workshop, Paris, pp. 163–166 (2003)
8. Blundo, C., D'Arco, P., Santis, A., Listo, M.: Definitions and Bounds for Self-healing Key Distribution. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 234–245. Springer, Heidelberg (2004)
9. Hong, D., Kang, J.: An Efficient Key Distribution Scheme with Selfhealing Property. IEEE Communication Letters 9, 759–761 (2005)
10. Dutta, R., Wu, Y., Mukhopadhyay, S.: Constant Storage Self-Healing Key Distribution with Revocation in Wireless Sensor Network. In: IEEE International Conference on Communications, 2007, pp. 1323–1332 (2007)
11. Shamir, A.: How to share a secret. Communications of ACM 22, 612–613 (1979)
12. Xukai, Z., Yuan-Shun, D.: A Robust and Stateless Self-Healing Group Key Management Scheme. In: International Conference on Communication Technology, pp. 1–4 (2006)
13. Dutta, R., Chang, E., Mukhopadhyay, S.: Efficient Self-healing Key Distribution with Revocation for Wireless Sensor Networks Using One Way Hash Chains. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 385–400. Springer, Heidelberg (2007)
14. Chadha, A., Yonghe, L., Das, S.K.: Group Key Distribution via Local Collaboration in Wireless Sensor Networks. In: Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pp. 46–54 (2005)
15. Biming, T., Mingxing, H.: A Self-healing Key Distribution Scheme with Novel Properties. International Journal of Network Security 7(1), 115–120 (2008)

Author Index