Feng Bao
Hui Li
Guilin Wang (Eds.)

LNCS 5451

Information Security Practice and Experience

5th International Conference, ISPEC 2009
Xi'an, China, April 2009
Proceedings

Springer

Feng Bao   Hui Li   Guilin Wang (Eds.)

Information Security Practice and Experience

5th International Conference, ISPEC 2009
Xi'an, China, April 13-15, 2009
Proceedings

Springer

Volume Editors

Feng Bao
Institute for Infocomm Research (I^2R)
1 Fusionopolis Way, 19-01 Connexis (South Tower)
Singapore 138632, Singapore
E-mail: baofeng@i2r.a-star.edu.sg

Hui Li
Xidian University
School of Telecommunications Engineering
2 South Taibai Road, Xi'an, Shaanxi 710071, China
E-mail: xd.lihui@gmail.com

Guilin Wang
University of Birmingham
School of Computer Science
Birmingham, B15 2TT, UK
E-mail: g.wang@cs.bham.ac.uk
`http://www.cs.bham.ac.uk/~gzw/`

Library of Congress Control Number: Applied for

CR Subject Classification (1998): E.3, D.4.6, C.2.0, H.2.0, K.6.5, K.4.4

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-642-00842-9 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-00842-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper   SPIN: 12642663   06/3180   5 4 3 2 1 0

Preface

The 5th International Conference on Information Security Practice and Experience (ISPEC 2009) was held in Xi'an, China, April 13–15, 2009.

The ISPEC conference series is an established forum that brings together researchers and practitioners to provide a confluence of new information security technologies, including their applications and their integration with IT systems in various vertical sectors. In previous years, ISPEC has taken place in Singapore (2005), Hangzhou, China (2006), Hong Kong, China (2007), and Sydney, Australia (2008). For all sessions, as this one, the conference proceedings were published by Springer in the *Lecture Notes in Computer Science* series.

In total, 147 papers from 26 countries were submitted to ISPEC 2009, and 34 were finally selected for inclusion in the proceedings (acceptance rate 23%). The accepted papers cover multiple topics of information security and applied cryptography. Each submission was anonymously reviewed by at least three reviewers. We are grateful to the Program Committee, which was composed of more than 40 well-known security experts from 15 countries; we heartily thank them as well as all external reviewers for their time and valued contributions to the tough and time-consuming reviewing process.

In addition to the regular paper presentations, the program also featured four invited talks by Yupu Hu, from Xidian University, China; Youki Kadobayashi, from Nara Institute of Science and Technology, Japan; Mark Ryan, from the University of Birmingham, UK; and Gene Tsudik, from the University of California at Irvine, USA. We are grateful to them for accepting our invitation to speak at the conference.

The conference was organized and sponsored by Xidian University, China, co-organized by the School of Telecommunications Engineering, Xidian University, China; the Key Laboratory of Computer Networks and Information Security, Ministry of Education, China; and the National 111 Program of Introducing Talents of Discipline to Universities on Fundamental Theory and Technology of Modern Wireless Information Networks, China.

Special thanks are due to Ying Qiu for managing the review system, Libin Zhao for maintaining the conference website, and the Organizing Committee for dealing with local issues.

Last but not least, we would like to thank all the authors who submitted their papers to ISPEC 2009, and all the attendees from all over the world.

April 2009

Feng Bao
Hui Li
Guilin Wang

ISPEC 2009

5th International Conference
on Information Security Practice and Experience

Xi'an, China
April 13–15, 2009

Organized and Sponsored by

Xidian University, China

Co-organized by

School of Telecommunications Engineering, Xidian University, China.

Key Laboratory of Computer Networks and Information Security,
Ministry of Education, China.

National 111 Program of Introducing Talents of Discipline to
Universities on Fundamental Theory and Technology of
Modern Wireless Information Networks, China

General Chairs

Robert H. Deng	Singapore Management University, Singapore
Jianfeng Ma	Xidian University, China

Program Chairs

Feng Bao	I^{2}R, Singapore
Hui Li	Xidian University, China

Publication Chair

Guilin Wang	University of Birmingham, UK

Organizing Committee Chairs

Qingqi Pei	Xidian University, China
Xiaoyan Zhu	Xidian University, China
Yuanyuan Zuo	Xidian University, China

Program Committee

Kefei Chen	Shanghai Jiaotong University, China
Lily Chen	NIST, USA
Liqun Chen	HP Labs, UK
Dooho Choi	ETRI, Korea
Ed Dawson	QUT, Australia
Dengguo Feng	Chinese Academy of Sciences, China
Clemente Galdi	University of Naples "Federico II", Italy
David Galindo	ENS, France
Dieter Gollmann	TU Hamburg, Germany
Guang Gong	University of Waterloo, Canada
Javier Herranz	IIIA, Spain
Yupu Hu	Xidian University, China
Aggelos Kiayias	University of Connecticut, USA
Miroslaw Kutylowski	Wroclaw University of Technology, Poland
Jin Kwak	Soonchunhyang University, Korea
Xuejia Lai	Shanghai Jiaotong University, China
Benoît Libert	UCL, Belgium
Javier Lopez	University of Malaga, Spain
Michael Locasto	Dartmouth College, USA
Atsuko Miyaji	JAIST, Japan
Yi Mu	University of Wollongong, Australia
Elisabeth Oswald	University of Bristol, UK
Olivier Pereira	UCL, Belgium
Josef Pieprzyk	Macquarie University, Australia
Mark Ryan	University of Birmingham, UK
Sattar B. Sadkhan	University of Babylon, Iraq
Kouichi Sakurai	Kyushu University, Japan
Alice Silverberg	University of California at Irvine, USA
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Future University Hakodate, Japan
Toshiaki Tanaka	KDDI R&D Labs, Japan
Allan Tomlinson	Royal Holloway, UK
Jorge Villar	Universitat Politecnica de Catalunya, Spain
Guilin Wang	University of Birmingham, UK
Duncan Wong	City University of Hong Kong, China
Yongdong Wu	I^{2}R, Singapore
Chaoping Xing	NTU, Singapore
Heung Youl Youm	Soonchunhyang University, Korea
Fangguo Zhang	Sun Yat-Sen University, China
Rui Zhang	AIST, Japan
Huafei Zhu	I^{2}R, Singapore

External Reviewers

Toru Akishita
Nuttapong Attrapadung
Man Ho Au
Luigi Catuogno
Witold Charatonik
Jianhong Chen
Shengli Chen
Xiaofeng Chen
Tom Chothia
Jacek Cichon
Stefan Ciobaca
Baudoin Collard
Paolo D'Arco
Vanesa Daza
Hiroshi Doi
Ming Duan
Xinxin Fan
Hossein Ghodosi
Dong-Guk Han
Wei Han
Xuan Hong
Honggang Hu
Qiong Huang
Xinyi Huang
Vincenzo Iovino
Yuichi Kaji
John Kelsey
Ikkyun Kim
Juhan Kim
Namshik Kim
Fagen Li
Jin Li
Xiangxue Li
Zhijun Li
Joseph Liu
Yu Long
Krzysztof Majcher
Mark Manulis
Gogolewski Marcin
Andrew Moss
Mridul Nandi
Shivaramakrishnan Narayan
David Nowak
Kyunghee Oh
Takeshi Okamoto
Kazumasa Omote
Jose Antonio Onieva
Souradyuti Paul
Ray Perlner
Jason Reid
Chun Ruan
Germán Sáez
Masaaki Shirase
Leonie Simpson
Nigel Smart
Jason Smith
Ben Smyth
Marianne Swanson
Qiang Tang
Isamu Teranishi
Damien Vergnaud
Yongtao Wang
Bogdan Warinschi
Ralf-Philipp Weinmann
Wu Wei
Mi Wen
Yamin Wen
Stephen Wolthusen
Fubiao Xia
Jing Xu
Lingling Xu
Guomin Yang
Rehana Yasmin
Reza Z'Aba
Jinmin Zhong

Table of Contents

Efficient and Provable Secure Ciphertext-Policy Attribute-Based Encryption Schemes

Luan Ibraimi[1], Qiang Tang[1], Pieter Hartel[1], and Willem Jonker[1,2]

[1] Faculty of EEMCS, University of Twente, the Netherlands
[2] Philips Research, the Netherlands

Abstract. With a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme, a user's private key is associated with a set of attributes and the data is encrypted under an access policy defined by the message sender. A user can decrypt a ciphertext if and only if her attributes satisfy the access policy. In CP-ABE, since the message sender enforces the access policy during the encryption phase, the policy moves with the encrypted data. In this paper, we provide an efficient CP-ABE scheme which can express any access policy represented by a formula involving the *and* ($\wedge$) and *or* ($\vee$) operators. The scheme is secure under Decision Bilinear Diffie-Hellman (DBDH) assumption. Furthermore, we extend the expressiveness of the scheme by including the *of* operator in addition to $\wedge$ and $\vee$. We provide a comparison with some existing CP-ABE schemes and show that our schemes are more efficient.

1 Introduction

Public-key encryption is an asymmetric primitive which uses a pair of keys, where a private key which is kept secret and a public key which is widely distributed. If Alice wants to send a confidential message to Bob, she can encrypt the message with the public key of Bob and only Bob can decrypt the message using his private key. With a Public-Key Infrastructure (PKI), a public key must be obtained from, or at least be certified by the Trusted Third Party (TTP) of the PKI. With an Identity-Based Encryption (IBE) scheme [6,10,19], any string can be used to generate a public key without the involvement of the TTP. IBE thus creates a degree of flexibility that a PKI cannot offer. However, if Alice does not know the identity of her party, but instead she only knows certain attributes of the recipient, then the above solutions will not work.

The solution to this problem is provided by Attribute-Based Encryption (ABE) which identifies a user with a set of attributes [17]. In the seminal paper, Sahai and Waters use biometric measurements as attributes in the following way. A private key, associated with a set of attributes ω, can decrypt a ciphertext encrypted with a public key, associated with a set of attributes ω', only if the sets ω and ω' overlap sufficiently as determined by a threshold value t. We refer to the scheme, proposed by Sahai and Waters, as the SW scheme. A more general policy to decide which attributes are required to decrypt a message is provided by an access tree. For example the access tree $\tau = class1978 \wedge mycollege \vee myteacher$

states that all students with the attribute *class*1978 who studied at *mycollege* as well as the teacher possessing the attribute *myteacher* would satisfy the policy.

There are two variants of ABE, namely Key-Policy based ABE (KP-ABE) [12] and Ciphertext-Policy based ABE (CP-ABE)[3,9]. In KP-ABE, the ciphertext is associated with a set of attributes and the private key is associated with the access tree. The message sender does not define the privacy policy and has no control over who has access to the data, except for defining the set of descriptive attributes necessary to decrypt the ciphertext. The trusted authority who generates user's private key defines the combination of attributes for which the private key can be used. In CP-ABE, the idea is reversed: the ciphertext is associated with the access tree and the message sender determines the policy under which the data can be decrypted, while the private key is associated with a set of attributes.

Pirreti *et al.* [16] give a construction and implementation of a modified SW scheme, which, compared to the original scheme, reduces computational overhead during the encryption and the key generation phases. Goyal *et al.* [12] introduce the idea of KP-ABE and propose a new scheme. In their scheme, when a user makes a secret key request, the trusted authority determines which combination of attributes must appear in the ciphertext for the user to decrypt. In essence, this scheme is an extension of the SW scheme, where, instead of using the Shamir [18] secret sharing technique in the private key, the trusted authority uses a more generalized form of secret sharing to enforce a monotonic access tree. Chase [8] constructs a multi-authority ABE scheme, which allows multiple independent authorities to monitor attributes and distribute secret keys. A related work to KP-ABE is a predicate encryption paradigm or searching on encrypted data [1,5,7,14]. Predicate encryption has the advantages of providing ciphertext anonymity by hiding the access structures, however, the system is less expressive compared to schemes which leave the access structures in the clear. Smart [20] gives an access control data scheme which encrypts data to an arbitrary collection of identities using a variant of the Boneh-Franklin IBE scheme.

The first CP-ABE scheme proposed by Bethencourt *et al.* [3] uses threshold secret sharing to enforce the policy during the encryption phase. We refer to this scheme as the BSW scheme. The main drawback of the BSW scheme is that it requires polynomial interpolation to reconstruct the secret, thus many expensive pairing and exponentiation operations are required in the decryption phase. The scheme is secure in the generic group model, which provides evidence to the hardness of the problem, without giving security proof which reduces the problem of breaking the scheme to a well-studied complexity-theoretic problem. The CP-ABE scheme, proposed by Cheung and Newport [9], does not use threshold secret sharing but uses random elements to enforce the policy during the encryption phase. We refer to this scheme as the CN scheme. The CN scheme has two drawbacks. Firstly, it is not sufficiently expressive because it supports only policies with logical conjunction. Secondly, the size of the ciphertext and secret key increases linearly with the total number of attributes in the system. This makes the CN scheme inefficient. Goyal *et al.* [11] give a "bounded" CP-ABE construction. The disadvantage of their scheme is that the depth of the access trees d under which messages can be encrypted is defined in the setup phase. Thus, the message sender is restricted to use only an access tree which has the depth $d' \leq d$.

Contribution. In this paper we aim at designing efficient CP-ABE schemes, which can be proven based on standard complexity-theoretic assumptions. More specifically, our contribution is twofold:

- We present a new technique for realizing CP-ABE without using Shamir's threshold secret sharing. We first show such a construction which is referred to as as the basic CP-ABE scheme. In this scheme, the message sender defines the privacy policy through an access tree which is n-ary tree represented by $\wedge$ and $\vee$ nodes. Note that, realizing a scheme, which does not use threshold secret sharing, is important for resource constraint devices since calculating polynomial interpolations to construct the secret is computationally expensive. Compared to the CN scheme, our scheme requires fewer computations during the encryption, key generation and decryption phases.
- Next, we extend the basic CP-ABE scheme and provide a second CP-ABE scheme which uses Shamir's threshold secret sharing technique [18]. The access tree is an n-ary tree represented by $\wedge$, $\vee$ and *of* nodes. We compare the efficiency of our scheme with the BSW scheme and show that our scheme requires less computations in the key generation, encryption and decryption phases.

Organization. The rest of this paper is organized as follows. In Section 2 we review concepts of the access structure, secret sharing, CP-ABE, and bilinear pairing. In Section 3 we introduce the basic CP-ABE scheme which is secure under DBDH assumption in the selective-attribute model. In Section 4 we provide an extension to the basic CP-ABE scheme by including the *of* operator in addition to $\wedge$ and $\vee$. In the last section we conclude the paper.

2 Background Knowledge

In this section we introduce the notions related to access structure, secret sharing, the security definition of CP-ABE, and bilinear maps.

2.1 Access Structure

We restate the definition of Access Structure in [2].

Definition 1. (Access Structure). *Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\cdots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \cdots, P_n\}$, i.e., $A \subseteq 2^{\{P_1,P_2,\cdots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

2.2 Secret Sharing Schemes

In designing our CP-ABE schemes we will make use of two different secret-sharing schemes: unanimous consent control by modular addition scheme and the Shamir secret sharing scheme.

Unanimous Consent Control by Modular Addition Scheme. In a unanimous consent control by modular addition scheme [15], there is a dealer who splits a secret s into t shares in a such way that all shares are required to reconstruct the secret s. To share the secret s, $0 \leq s \leq p-1$ for some integer p, the dealer generates a $t-1$ random numbers s_i such that $1 \leq s_i \leq p-1$, $1 \leq i \leq t-1$ and $s_t = s - \sum_{i=1}^{t-1} s_i \; mod \; p$. The secret s is recovered by: $s = \sum_{i=1}^{t} s_i$. Shares s_i, $1 \leq i \leq t$ are distributed to parties P_i, $1 \leq i \leq t$. For each party P_i, $1 \leq i \leq t$, the shares are random numbers between 0 and $p-1$, thus no party has any information about s except the dealer.

Shamir's Secret Sharing Scheme. In Shamir's secret sharing technique [18] a secret s is divided into n shares in a such way that any subset of t shares, where $t \leq n$, can together reconstruct the secret; no subset smaller than t can reconstruct the secret. The technique is based on polynomial interpolation where a polynomial $y = f(x)$ of degree $t-1$ is uniquely defined by t points (x_i, y_i). The details of the scheme are as follows:

1. Setup. The dealer D wants to distribute the secret $s > 0$ among t users.
   1)D chooses a prime $p > max(s, n)$, and defines $a_0 = s$.
   2)D selects $t-1$ random coefficients $a_1, ..., a_{t-1}, 0 \leq a_j \leq p-1$, and defines the random polynomial over $\mathbb{Z}_p$, $f(x) = \Sigma_{j=0}^{t-1} a_j x^j$.
   3)D computes $s_i = f(i) \; mod \; p$, and sends securely the share s_i to user p_i together with the public index i.
2. Pooling of shares. Any group of t or more users pool their distinct shares $(x, y) = (i, s_i)$ allowing computation of the coefficients a_j of $f(x)$ by Lagrange interpolation, $f(x) = \Sigma_{i=0}^{t-1} l_j(x)$ where $l_j(x) = \Pi_{1 \leq j \leq t, j \neq i} \frac{x - x_j}{x_i - x_j}$. The secret is $f(0) = a_0 = s$.

2.3 Ciphertext-Policy ABE

According to the definition given in [3], a CP-ABE schemes consist of four polynomial time algorithms:

– $\mathsf{Setup}(k)$. The setup algorithm takes as input a security parameter k and outputs the public parameters pk and a master key mk.
– $\mathsf{Keygen}(\omega, mk)$. The algorithm takes as input the master key mk and a set of attributes ω. The algorithm outputs a secret key sk_ω associated with ω.
– $\mathsf{Encrypt}(m, \tau, pk)$. The encryption algorithm takes as input a message m, an access tree τ representing an access structure, and the public key pk. The algorithm will return the ciphertext c_τ such that only users who have the secret key generated from the attributes that satisfy the access tree will be able to decrypt the message.

– Decrypt(c_τ, sk_ω). The decryption algorithm takes as input a ciphertext c_τ, a secret key sk_ω associated with ω, and it outputs a message m or an error symbol $\perp$.

The semantic security against chosen-plaintext attack (CPA) is modeled in the selective-attribute (sAtt) model, where the adversary must provide the challenge access tree he wishes to attack before he receives the public parameters from the challenger. Suppose, that the adversary in the Init phase chooses the challenge access tree $\tau^* = (A \wedge B) \vee C$. In Phase1, the adversary can make secret key requests to Keygen oracle for any attribute set ω with the restriction that attributes $A, B, C \notin \omega$. The selective-attribute (sAtt) model can be considered to be analogous to the selective-ID model [4] used in identity-based encryption schemes, in which the adversary commits ahead of time the ID^* it will attack, and where the adversary can make secret key requests to Keygen oracle for any ID such that $ID \neq ID^*$.

The game is carried out between a challenger and an adversary $\mathcal{A}$, where the challenger simulates the protocol execution and answers queries from $\mathcal{A}$. Specifically, the game is as follows:

1. Init. The adversary chooses the challenge access tree τ^* and gives it to the challenger.
2. Setup. The challenger runs Setup to generate (pk, mk) and gives the public key pk to the adversary $\mathcal{A}$.
3. Phase1. $\mathcal{A}$ makes a secret key request to the Keygen oracle for any attribute set $\omega = \{a_j | a_j \in \Omega\}$, with the restriction that $a_j \notin \tau^*$. The challenger returns Keygen(ω, mk).
4. Challenge. $\mathcal{A}$ sends to the challenger two messages m_0, m_1.The challenger picks a random bit $b \in \{0, 1\}$ and returns $c_b =$ Encrypt(m_b, τ^*, pk).
5. Phase2. $\mathcal{A}$ can continue querying Keygen with the same restriction as during Phase1.
6. Guess. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

Definition 2. *A CP-ABE scheme is said to be secure against an adaptive chosen-plaintext attack (CPA) if any polynomial-time adversary has only a negligible advantage in the IND-sAtt-CPA game, where the advantage is defined to be* $\epsilon = |\Pr[b' = b] - \frac{1}{2}|$.

2.4 Review of Pairing

We briefly review the basis of bilinear pairing. A pairing (or, bilinear map) satisfies the following properties:

1. $\mathbb{G}_0$ and $\mathbb{G}_1$ are two multiplicative groups of prime order p.
2. g is a generator of $\mathbb{G}_0$.
3. $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ is an efficiently-computable bilinear map with the following properties:

- Bilinear: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- Non-degenerate: $\hat{e}(g, g) \neq 1$.

$\mathbb{G}_0$ is said to be a bilinear group if the group operation in $\mathbb{G}_0$ can be computed efficiently and if there exists a group $\mathbb{G}_1$ and an efficiently-computable bilinear map $\hat{e}$ as defined above.

The Decision Bilinear Diffie-Hellman (DBDH) problem is defined as follows. Given $g, g^a, g^b, g^c \in \mathbb{G}_0$ as input, the adversary must distinguish a valid tuple $\hat{a}(g, g)^{abc} \in \mathbb{G}_1$ from the random element $Z \in \mathbb{G}_1$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the Decision Bilinear Diffie-Hellman (DBDH) problem in $\mathbb{G}_0$ if:

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, Z) = 0]| \geq \epsilon.$$

Here the probability is over the random choice of $a, b, c \in \mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_1$, and the random bits of $\mathcal{A}$ (the adversary is a nondeterministic algorithm).

Definition 3. *We say that the (t, ϵ)-DBDH assumption holds if no t-time algorithm has advantage at least ϵ in solving the DBDH problem in $\mathbb{G}_0$.*

3 The Basic CP-ABE Construction

In this paper, the access tree is a n-ary tree, in which leaves are attributes and inner nodes are $\wedge$ and $\vee$ boolean operators. Intuitively, the access tree is a policy which specifies which combination of attributes can decrypt the ciphertext. Consider the following example where a patient wants to specify access restrictions on his medical data. The patient can enforce the access policy in the encryption phase. Each member from the medical staff who has enough attributes should be able to decrypt the encrypted message. For instance, a patient wants to allow his data to be seen by Doctor A who works at Department A or by Doctor B who works at Department B. Using boolean operators the patient defines the following access policy: $\tau_{Data} = (Doc.A \wedge Dep.A) \vee (Doc.B \wedge Dep.B)$.

To decrypt the ciphertext which is encrypted according to the τ_{Data} access tree, the decryptor must possess a private key, which is associated with the attribute set which satisfies τ_{Data}. To determine whether or not an access tree is satisfied, we interpret each attribute as a logical variable. Possession of the secret key for the corresponding attribute makes the logical variable true. If the decryptor does not possess the attribute, the variable is false. For the policy above there are several different sets of attributes that can satisfy the access tree, such as the secret key associating with the attribute set $\{Doc.A, Dep.A\}$, the secret key associating with the attribute set $\{Doc.B, Dep.B\}$, or the secret key associating with all attributes defined in the access tree.

3.1 Description of the Basic Scheme

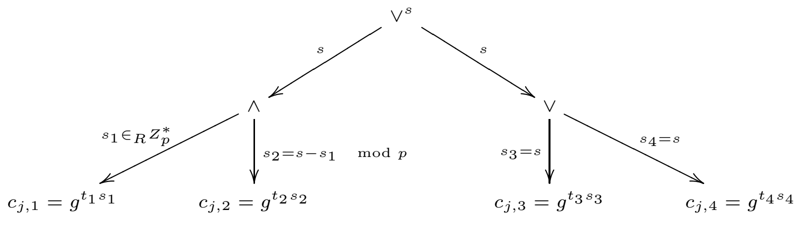
The polynomial time algorithms of the basic CP-ABE scheme are defined as follows.

1. $\mathsf{Setup}(k)$: On input of the security parameter k, this algorithm generates the following.
   (a) Generate a bilinear group $\mathbb{G}_0$ of prime order p with a generator g and bilinear map $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$
   (b) Generate the attribute set $\Omega = \{a_1, a_2, \ldots a_n\}$, for some integer n, and random elements $\alpha, t_1, t_2 \ldots t_n \in \mathbb{Z}_p$.

   Let $y = \hat{e}(g,g)^\alpha$ and $T_j = g^{t_j}$ $(1 \leq j \leq n)$. The public key is $pk = (\hat{e}, g, y, T_j\,(1 \leq j \leq n))$ and the master secret key is $mk = (\alpha, t_j\,(1 \leq j \leq n))$.
2. $\mathsf{Keygen}(\omega, mk)$: The algorithm performs as follows.
   (a) Select a random value $r \in \mathbb{Z}_p$ and compute $d_0 = g^{\alpha - r}$.
   (b) For each attribute a_j in ω, compute $d_j = g^{rt_j^{-1}}$.
   (c) Return the secret key $sk_\omega = (d_0, \forall a_j \in \omega : d_j)$
3. $\mathsf{Encrypt}(m, \tau, pk)$: To encrypt a message $m \in \mathbb{G}_1$ the algorithm proceeds as follows:
   (a) First level encryption: Select a random element $s \in \mathbb{Z}_p$ and compute $c_0 = g^s$ and
   $$c_1 = m \cdot y^s = m \cdot \hat{e}(g,g)^{\alpha s}$$
   (b) Second level encryption: Set the value of the root node of τ to be s, mark all child nodes as un-assigned, and mark the root node assigned. Recursively, for each un-assigned non-leaf node, do the following:
      – If the symbol is $\wedge$ and its child nodes are marked un-assigned, we use a unanimous consent control by modular addition scheme to assign a value to each child node. To do that, for each child node except the last one, assign a random value s_i where $1 \leq s_i \leq p-1$, and to the last child node assign the value $s_t = s - \sum_{i=1}^{t-1} s_i \mod p$. Mark this node assigned.
      – If the symbol is $\vee$, set the values of each child node to be s. Mark this node assigned.

   Values of the leaves of τ are used to produce ciphertext components.
   (c) For each leaf attribute $a_{j,i} \in \tau$, compute $c_{j,i} = T_j^{s_i}$ where i denotes the index of the attribute in the access tree. The index values are uniquely assigned to leave nodes in an ordering manner for a given access structure.
   (d) Return the ciphertext $c_\tau = (\tau, c_0, c_1, \forall a_{j,i} \in \tau : c_{j,i})$.

In the following figure, we show an example of assigning secret shares s_i to the access tree $\tau = (T_1 \wedge T_2) \vee (T_3 \vee T_4)$.



4. $\mathsf{Decrypt}(c_\tau, sk_\omega)$: If ω does not satisfy τ, return $\perp$, otherwise the algorithm chooses the smallest set $\omega' \subseteq \omega$ (we assume that this can be computed efficiently by the decryptor) that satisfies τ and performs as follows:

   (a) For every attribute $a_j \in \omega'$, compute

$$\begin{aligned}\prod_{a_j \in \omega'} \hat{e}(c_{j,i}, d_j) &= \prod_{a_j \in \omega'} \hat{e}(T_j^{s_i}, g^{rt_j^{-1}}) \\ &= \prod_{a_j \in \omega'} \hat{e}(g^{t_j s_i}, g^{rt_j^{-1}}) \\ &= \hat{e}(g,g)^{rs}\end{aligned}$$

   (b) Compute

$$\begin{aligned}\hat{e}(c_0, d_0) \cdot \hat{e}(g,g)^{rs} &= \hat{e}(g^s, g^{\alpha - r}) \cdot \hat{e}(g,g)^{rs} \\ &= \hat{e}(g^s, g^\alpha)\end{aligned}$$

   (c) Return m', where

$$\begin{aligned}m' &= \frac{c_1}{\hat{e}(g^s, g^\alpha)} \\ &= \frac{m \cdot \hat{e}(g,g)^{\alpha s}}{\hat{e}(g^s, g^\alpha)} \\ &= m\end{aligned}$$

3.2 Security and Efficiency Analysis

The proposed scheme is proven IND-sAtt-CPA secure under the DBDH assumption. For more details of the proof, the reader can refer to the full version of this paper [13].

The number of calculations in the Encryption algorithm depends on the number of attributes in the access tree τ. The encryption requires $|\tau| + 1$ exponentiations in $\mathbb{G}_0$ and one exponentiation in $\mathbb{G}_1$. The number of calculations in the KeyGen algorithm depends on the number of attributes in the set ω that the user has, namely $|\omega| + 1$ exponentiations in $\mathbb{G}_0$. The number of calculations in the decryption algorithm depends on the number of attributes in the attribute set ω'. The decryption requires $|\omega'| + 1$ pairing operations, $|\omega'|$ multiplications, but no exponentiations in $\mathbb{G}_1$.

In Table 1, we compare our CP-ABE scheme with the CN scheme. We count the number of calculations in the encryption, key generation, and decryption phases. Compared to the CN scheme, our scheme requires fewer computations in the encryption, key generation and decryption phase.

Table 1. The Comparison with the CN Scheme

	Our Scheme			The CN Scheme		
	Exp.($\mathbb{G}_0$)	Exp.($\mathbb{G}_1$)	Pairing	Exp.($\mathbb{G}_0$)	Exp.($\mathbb{G}_1$)	Pairing
Encrypt	$\lvert\tau\rvert+1$	1	/	$\lvert\Omega\rvert+1$	1	/
Keygen	$\lvert\omega\rvert+1$	/	/	$\lvert\Omega\rvert+1$	/	/
Decrypt	/	/	$\lvert\omega'\rvert+1$	/	/	$\lvert\Omega\rvert+1$
	Ω is the set of all attributes defined in the Setup phase τ is the access tree ω is the set of attributes the user has, $\omega \subseteq \Omega$ ω' the set of attributes satisfying the access tree, $\omega' \subseteq \omega$					

4 Extension of the Expressiveness

In the basic scheme, the access tree is a n-ary tree represented by $\wedge$ and $\vee$ nodes, which allows the user who performs encryption to express any privacy policy using boolean formulas. Similar to the BSW scheme, we would like to have an n-ary access tree which supports the *of* operator. The essential idea is to allow the encryptor to define the minimum number of attributes from a given list of attributes that the decryptor has to posses in order to decrypt the message. For instance, to decrypt the ciphertext encrypted under the policy τ =2 *of (class1978, mycollege, myteacher)*, the decryptor must have at least two out of three attributes. We extend the basic CP-ABE scheme to support the *of* operator as follows:

1. Setup and KeyGen are the same as in basic CP-ABE scheme.
2. Encrypt(m, τ, pk) : To encrypt a message $m \in \mathbb{G}_1$ the algorithm proceeds as follows:

   (a) First level encryption: Select a random element $s \in \mathbb{Z}_p$ and compute $c_0 = g^s$ and

   $$\begin{aligned} c_1 &= m \cdot y^s \\ &= m \cdot \hat{e}(g, g)^{\alpha s} \end{aligned}$$

   (b) Second level encryption: Set the value of the root node to be s, mark all child nodes as un-assigned, and mark the root node assigned. Recursively, for each un-assigned non-leaf node, do the following:

   – If the symbol is *of* (threshold operator), and its child nodes are marked un-assigned, the secret s is divided using (t, n) Shamir's secret sharing technique where $t \neq n$, and n is the total number of child nodes and t is the number of child nodes necessary to reconstruct the secret. To each child node a share secret $s_i = f(i)$ is assigned. Mark this node assigned.
   – If the symbol is $\wedge$, and its child nodes are marked un-assigned, the secret s is divided using (t, n) Shamir's secret sharing technique where $t = n$, and n is the number of the child nodes. To each child node a share secret $s_i = f(i)$ is assigned. Mark this node assigned.

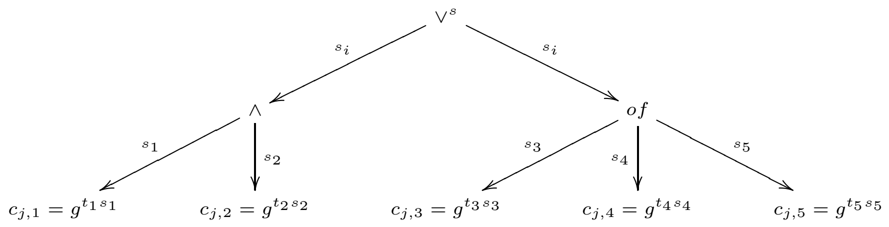
  - If the symbol is $\vee$, and its child nodes are marked un-assigned, the secret s is divided using (t, n) Shamir's secret sharing technique where $t = 1$ and n is the number of the child nodes. To each child node a share secret $s_i = f(i)$ is assigned. Mark this node assigned.

  Values of the leaves of τ are used to produce ciphertext components.

(c) For each leaf attribute $a_{j,i} \in \tau$, compute $c_{j,i} = T_j^{s_i}$, where i denotes the index of the attribute in the access tree.

(d) Return the ciphertext: $c_\tau = (\tau, c_0, c_1, \forall a_{j,i} \in \tau : c_{j,i})$.

In the following figure, we show an example of assigning secret shares s_i to the access tree $\tau = (T_1 \wedge T_2) \vee 2\ of\ (T_3, T_4, T_5)$.



3. $\mathsf{Decrypt}(c_\tau, sk_\omega)$: If ω does not satisfy τ, return $\perp$, otherwise the algorithm chooses the smallest set $\omega' \subseteq \omega$ that satisfies τ and performs as follows:

(a) For every attribute $a_j \in \omega'$, compute

$$\begin{aligned}\prod_{a_j \in \omega'} \hat{e}(c_{j,i}, d_j)^{l_i(0)} &= \hat{e}(T_j^{s_i}, g^{rt_j^{-1}})^{l_i(0)} \\ &= \prod_{a_j \in \omega'} \hat{e}(g^{t_j s_i}, g^{rt_j^{-1}})^{l_i(0)} \\ &= \prod_{a_j \in \omega'} \hat{e}(g, g)^{rs_i l_i(0)} \\ &= \hat{e}(g, g)^{rs}\end{aligned}$$

$l_i(0)$ is a Lagrange coefficient and can be computed by everyone who knows the index of the attribute in the access tree.

(b) Compute

$$\begin{aligned}\hat{e}(c_0, d_0) \cdot \hat{e}(g, g)^{rs} &= \hat{e}(g^s, g^{\alpha - r}) \cdot \hat{e}(g, g)^{rs} \\ &= \hat{e}(g^s, g^\alpha)\end{aligned}$$

(c) Return m', where

$$m' = \frac{c_1}{\hat{e}(g^s, g^\alpha)} = \frac{m \cdot \hat{e}(g, g)^{\alpha s}}{\hat{e}(g^s, g^\alpha)} = m$$

Table 2. The Comparison with the BSW scheme

	Our Scheme			The BSW Scheme		
	Exp.($\mathbb{G}$)	Exp.($\mathbb{G}_1$)	Pairing	Exp.($\mathbb{G}$)	Exp.($\mathbb{G}_1$)	Pairing
Encrypt	$\|\tau\|+1$	1	/	$2\|\tau\|+1$	1	/
KeyGen	$\|\omega\|+1$	/	/	$2\|\omega\|+1$	/	/
Decrypt	/	$\|\omega'\|$	$\|\omega'\|+1$	/	$\|\omega'\|$	$2\|\omega'\|$
(Note:)	Ω is the set of all attributes defined in the Setup phase					
	τ is the access tree					
	ω is the set of attributes the user has, $\omega \subseteq \Omega$					
	ω' the set of attributes satisfying the access tree, $\omega' \subseteq \omega$					

The proposed scheme is proven IND-sAtt-CPA secure under the DBDH assumption as shown in the full version of this paper [13]. In Table 2, we give a comparison of the efficiency of our extended CP-ABE scheme with the BSW scheme. Compared to the BSW scheme, our scheme requires fewer computations in the encryption, key generation and decryption phases.

5 Conclusion and Future Work

We have shown how to improve the efficiency of a CP-ABE scheme. Firstly, we presented a new technique to construct a CP-ABE scheme which does not use threshold secret sharing. The encryptor specifies the policy in the encryption phase using an n-ary tree which consists from $\vee$ and $\wedge$ nodes. Secondly, we presented a modified scheme which is more expressive compared to the basic scheme. In the modified scheme, the policy can be expressed as an n-ary access tree which consists of $\vee$, $\wedge$ and *of* nodes. We have shown that these schemes require less computations than some other similar schemes.

Acknowledgments

We thank Asim Muhammad and Peter Van Liesdonk for their suggestions and comments.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution, PhD. Thesis, Department of Computer Science,Technion (1996)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-Policy Attribute-Based Encryption. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society, Los Alamitos (2007)
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (Without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
8. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
9. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 456–465. ACM, New York (2007)
10. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
11. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM, New York (2006)
13. Ibraimi, L., Tang, Q., Hartel, P., Jonker, W.: Efficient and provable secure ciphertext-policy attribute-based encryption schemes. Technical Report TR-CTIT-08-75, CTIT, University of Twente (2008)
14. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
15. Menezes, A., Oorschot, P.V., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
16. Pirretti, M., Traynor, P., McDaniel, P., Waters, B.: Secure attribute-based systems. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 99–112 (2006)
17. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
18. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
19. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
20. Smart, N.P.: Access control using pairing based cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 111–121. Springer, Heidelberg (2003)

A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length

Keita Emura[1], Atsuko Miyaji[1], Akito Nomura[2],
Kazumasa Omote[1], and Masakazu Soshi[3]

[1] School of Information Science, Japan Advanced Institute of Science and Technology,
1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan
{k-emura,miyaji,omote}@jaist.ac.jp

[2] Graduate School of Natural Science and Technology, Kanazawa University,
Kakuma-machi, Kanazawa, Ishikawa, 920-1192, Japan
anomura@t.kanazawa-u.ac.jp

[3] Graduate School of Information Sciences, Hiroshima City University, 3-4-1
Ozuka-Higashi, Asa-Minami-Ku, Hiroshima, 731-3194, Japan
soshi@hiroshima-cu.ac.jp

Abstract. An Attribute-Based Encryption (ABE) is an encryption scheme, where users with some attributes can decrypt ciphertexts associated with these attributes. However, the length of the ciphertext depends on the number of attributes in previous ABE schemes. In this paper, we propose a new Ciphertext-Policy Attribute-Based Encryption (CP-ABE) with constant ciphertext length. Moreover, the number of pairing computations is also constant.

Keywords: Attribute-based encryption, Ciphertext-Policy, Constant Ciphertext Length.

1 Introduction

A user identity (such as the name, e-mail address and so on) can be used for accessing control of some resources. For example, in Identity-Based Encryption (IBE) schemes such as [4,6], an encryptor can restrict a decryptor to indicate the identity of the decryptor. An Attribute-Based Encryption (ABE) is an encryption scheme, where users with some attributes can decrypt the ciphertext associated with these attributes. The first ABE scheme has been proposed in [13], which is inspired by IBE. Although IBE schemes have a restriction such that an encryptor only indicates a single decryptor, in ABE schemes, an encryptor can indicate many decryptors by assigning common attributes of these decryptors such as gender, age, affiliation and so on. There are two kinds of ABE, Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE). KP-ABE [9,13] are schemes such that each private key is associated with an access structure. CP-ABE [2,7,8,12,14] are schemes such that each ciphertext is associated with an access structure. This means that an encryptor can decide who should or should not be allowed to decrypt. However, in all previous ABE schemes [2,7,8,9,12,13,14], the length of the ciphertext depends on the number of attributes. Also, the number of pairing computations depends on the number of attributes. A Predicate Encryption Scheme (PES), where secret keys correspond to predicates, and where ciphertexts are associated with attributes, has been proposed in [5,10]. It is shown that PES can be regarded as a kind of CP-ABE (see Appendix A and B in [12] for details). However, both the [5] and [10] schemes also have the same problems, in that the length of the ciphertext and the number of pairing computations are not constant.

Contribution. In this paper, for the first time we propose a CP-ABE with constant length of ciphertext and constant length of the number of pairing computations. The access structure used in our CP-ABE is constructed by AND-gates on multi-valued attributes. This is a subset of the access structures used in [7,12]. Although previous CP-ABE schemes [2,7,8,12,14] can complement our access structures, the length of the ciphertext depends on the number of attributes. This means that, until our work, to the best of our knowledge, there has been no scheme that enables a constant ciphertext length with AND-gates on multi-valued attributes.

Organization: The paper is organized as follows: Some definitions are presented in Section 2. The previous scheme is introduced in Section 3. Our scheme is described in Section 4. The security proof is presented in Section 5. Efficiency comparisons are made in Section 6.

2 Preliminary

In this section, some definitions are presented. Note that $x \in_R S$ means x is randomly chosen for a set S.

2.1 Bilinear Groups and Complexity Assumption

Definition 1 (**Bilinear Groups**). *Bilinear groups and a bilinear map are defined as follows:*

1. *$\mathbb{G}_1$ and $\mathbb{G}_T$ are cyclic groups of prime order p.*
2. *g_1 is a generator of $\mathbb{G}_1$.*
3. *e is an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ with the following properties.*
   - *Bilinearity : for all $u, u', v, v' \in \mathbb{G}_1$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$.*
   - *Non-degeneracy : $e(g_1, g_1) \neq 1_{\mathbb{G}_T}$ ($1_{\mathbb{G}_T}$ is the $\mathbb{G}_T$'s unit).*

Definition 2 (**DBDH assumption**). *The Decision Bilinear Diffie-Hellman (DBDH) problem in $\mathbb{G}_1$ is a problem, for input of a tuple $(g_1, g_1^a, g_1^b, g_1^c, Z) \in \mathbb{G}_1^4 \times \mathbb{G}_T$ to decide $Z = e(g_1, g_1)^{abc}$ or not. An algorithm $\mathcal{A}$ has advantage ϵ in solving DBDH problem in $\mathbb{G}_1$ if $Adv_{DBDH}(\mathcal{A}) := |\Pr[\mathcal{A}(g_1, g_1^a, g_1^b, g_1^c, e(g_1, g_1)^{abc}) = 0] - \Pr[\mathcal{A}(g_1, g_1^a, g_1^b, g_1^c, e(g_1, g_1)^z) = 0]| \geq \epsilon(\kappa)$, where $e(g_1, g_1)^z \in \mathbb{G}_T \setminus \{e(g_1, g_1)^{abc}\}$. We say that the DBDH assumption holds in $\mathbb{G}_1$ if no PPT algorithm has an advantage of at least ϵ in solving the DBDH problem in $\mathbb{G}_1$.*

2.2 Definition of Access Structures

Several access structures such as the threshold structure [13], the tree-based access structure [2,8], AND-gates on positive and negative attributes with wildcards [7], AND-gates on multi-valued attributes with wildcards [12], and the linear access structure [14] are used in previous ABE schemes. In our scheme, the sum of master keys are used to achieve the constant ciphertext length. Therefore, we use AND-gates on multi-valued attributes (which can be represented by using the sum of master keys) as follows:

Definition 3. *Let $\mathcal{U} = \{att_1, \ldots, att_n\}$ be a set of attributes. For $att_i \in \mathcal{U}$, $S_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}$ is a set of possible values, where n_i is the number of possible values for att_i. Let $L = [L_1, L_2, \ldots, L_n]$, $L_i \in S_i$ be an attribute list for a user, and $W = [W_1, W_2, \ldots, W_n]$, $W_i \in S_i$ be an access structure. The notation $L \models W$ expresses that an attribute list L satisfies an access structure W, namely, $L_i = W_i$ $(i = 1, 2, \ldots, n)$.*

The number of access structures is $\prod_{i=1}^{n} n_i$. For each att_i, an encryptor has to *explicitly* indicate a status $v_{i,*}$ from $S_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}$.

Differences between the previous AND-gate structures [7,12] and ours
If $n_i = 2$ $(i = 1, 2, \ldots, n)$, then our structure is the same as the access structures [7] excluding wildcards. In [12], an access structure W is defined as $W = [W_1, W_2, \ldots, W_n]$ for $W_i \subseteq S_i$, and $L \models W$ is defined as $L_i \in W_i$ $(i = 1, 2, \ldots, n)$. This means that our access structure is a subset of these in [7,12]. However, even if previous CP-ABE schemes [7,12] use our access structure, then the length of the ciphertext depends on the number of attributes.

2.3 Ciphertext-Policy Attribute-Based Encryption Scheme (CP-ABE)

CP-ABE is described using four algorithms, Setup, KeyGen, Encrypt and Decrypt [7].

Definition 4. *CP-ABE*

Setup: *This algorithm takes as input the security parameter κ, and returns a public key PK and a master secret key MK.*

KeyGen: *This algorithm takes as input PK, MK and a set of attributes L, and returns a secret key SK_L associated with L.*

Encrypt: *This algorithm takes as input PK, a message M and an access structure W. It returns a ciphertext C with the property that a user with SK_L can decrypt C if and only if $L \models W$.*

Decrypt: *This algorithm takes as input PK, C which was encrypted by W, and SK_L. It returns M if SK_L is associated with $L \models W$.*

2.4 Selective Game for CP-ABE

We use the definition of “Selective Game” for CP-ABE [7]. This CP-ABE game captures the indistinguishability of messages and the collusion-resistance of secret keys, namely, attackers cannot generate a new secret key by combining their secret keys. To capture the collusion-resistance, multiple secret key queries can be issued by the adversary $\mathcal{A}$ after the challenge phase. This means that $\mathcal{A}$ can issue the KeyGen queries L_1 and L_2 such as $(L_1 \not\models W^*) \wedge (L_2 \not\models W^*)$ and $(L_1 \cup L_2) \models W^*$. This collusion-resistance is an important property of CP-ABE scheme, which has not been considered in the Hierarchical IBE (HIBE) scheme such as [3].

Definition 5. *Selective Game for CP-ABE*

Init: *The adversary $\mathcal{A}$ sends the challenge access structure W^* to the challenger.*

Setup: *The challenger runs* Setup *and* KeyGen*, and gives PK to $\mathcal{A}$.*

Phase 1: *$\mathcal{A}$ sends an attribute list L to the challenger for a* KeyGen *query, where $L \not\models W^*$. The challenger answers with a secret key for these attributes. Note that these queries can be repeated adaptively.*

Challenge: *$\mathcal{A}$ sends two equal-length messages M_0 and M_1 to the challenger. The challenger chooses $\mu \in_R \{0,1\}$, and runs $C^* = \mathsf{Encrypt}(PK, M_\mu, W^*)$. The challenger gives the challenge ciphertext C^* to $\mathcal{A}$.*

Phase 2: *Same as Phase 1. $\mathcal{A}$ sends L to the challenger for a* KeyGen *query. The challenger answers with a secret key for these attributes. Note that $L \not\models W^*$, and these queries can be repeated adaptively.*

Guess: *$\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$.*

The advantage of $\mathcal{A}$ is defined as $Adv(\mathcal{A}) := |\Pr(\mu' = \mu) - \frac{1}{2}|$.

3 The Previous CP-ABE

In this section, we summarize the previous CP-ABE [7]. Let $\bar{\mathcal{U}} = \{\neg att_1, \ldots, \neg att_n\}$ a set of negative attributes for a set of attributes $\mathcal{U}$. We refer to attributes $att_i \in \mathcal{U}$ and their negations $\neg att_i$ as literals. Let $W = \bigwedge_{att_i \in I} \bar{att_i}$ be an access structure, where $I \subseteq \mathcal{U}$ and $\bar{att_i}$ is either att_i or $\neg att_i$. The public key elements T_i, T_{n+i}, T_{2n+i} correspond to the three properties of att_i, namely, *positive*, *negative* and *don't care.*

Protocol 1. *CP-ABE [CN07] [7]*

Setup(1^κ): *A trusted authority TA chooses a prime number p, a bilinear group $\mathbb{G}_1$ with order p, a generator $g_1 \in \mathbb{G}_1$, $y \in_R \mathbb{Z}_p$ and $t_i \in_R \mathbb{Z}_p$ $(i = 1, 2, \ldots, 3n)$, and computes $Y = e(g_1, g_1)^y$ and $T_i = g_1^{t_i}$ $(i = 1, 2, \ldots, 3n)$. TA outputs $PK = (e, g_1, Y, T_1, \ldots, T_{3n})$ and $MK = (y, t_1, \ldots, t_{3n})$.*

KeyGen(PK, MK, S): *Every* $att_i \notin S$ *is implicitly considered to be a negative attribute.* TA *chooses* $r_i \in_R \mathbb{Z}_p$ $(i = 1, 2, \ldots, n)$, *sets* $r = \sum_{i=1}^{n} r_i$, *and computes* $\hat{D} = g_1^{y-r}$. TA *computes* D_i *and* F_i *as follows:*

$$D_i = \begin{cases} g_1^{\frac{r_i}{t_i}} & (att_i \in S) \\ g_1^{\frac{r_i}{t_{n+i}}} & (att_i \notin S) \end{cases}, \quad F_i = g_1^{\frac{r_i}{t_{2n+i}}} (att_i \in \mathcal{U})$$

TA *outputs* $SK = (\hat{D}, \{D_i, F_i\}_{i \in [1,n]})$.

Encrypt(PK, M, W): *Let* $W = \bigwedge_{att_i \in I} \bar{att}_i$. *An encryptor chooses* $s \in_R \mathbb{Z}_p$, *and computes* $\tilde{C} = M \cdot Y^s$ *and* $\hat{C} = g_1^s$. *The encryptor computes* C_i *as follows:*

$$C_i = \begin{cases} T_i^s & (\bar{att}_i = att_i) \\ T_{n+i}^s & (\bar{att}_i = \neg att_i) \\ T_{2n+i}^s & (att_i \in \mathcal{U} \setminus I) \end{cases}$$

The encryptor outputs $C = (W, \tilde{C}, \hat{C}, \{C_i\}_{i \in [1,n]})$.

Decrypt(PK, C, SK): *A decryptor computes the pairing* $e(C_i, D_i)$ $(att_i \in I)$ *and* $e(C_i, F_i)$ $(att_i \notin I)$ *as follows:*

$$e(C_i, D_i) = \left\{ \begin{array}{ll} e(g_1^{t_i \cdot s}, g_1^{\frac{r_i}{t_i}}) & (\bar{att}_i = att_i) \\ e(g^{t_{n+i} \cdot s}, g_1^{\frac{r_i}{t_{n+i}}}) & (\bar{att}_i = \neg att_i) \end{array} \right\} = e(g_1, g_1)^{r_i \cdot s} \ (att_i \in I)$$

$$e(C_i, F_i) = e(g_1^{t_{2n+i} \cdot s}, g_1^{\frac{r_i}{t_{2n+i}}}) = e(g, g)^{r_i \cdot s} \ (att_i \notin I)$$

Then $\tilde{C} / (e(\hat{C}, \hat{D}) \prod_{i=1}^{n} e(g_1, g_1)^{r_i \cdot s}) = M \cdot e(g_1, g_1)^{sy} / e(g_1, g_1)^{s(y-r)} e(g_1, g_1)^{sr}$ $= M$ *holds.*

To compute $e(g_1, g_1)^{sr}$, the decryptor has to compute either $e(C_i, D_i)$ or $e(C_i, F_i)$ for each i. This means that all C_i are included in a ciphertext, and thus the length of a ciphertext depends on the number of attributes. This scheme does not provide for adding new attributes after Setup. If some attributes are added after Setup, then some users who have already obtained the secret key can decrypt a ciphertext which one must not be able to decrypt. For example, let $\mathcal{U} = \{att_1, att_2\}$, and assume that a user U has secret keys of att_1 and att_2, and that a ciphertext C is associated with $W = att_1 \wedge att_2$. Then, U can decrypt a ciphertext associated with $att_1 \wedge att_2 \wedge att_3$ without a secret key of att_3. Concretely, U ignores a part of the ciphertext for att_3. CP-ABE schemes which enable the addition of new attributes after Setup have been proposed in BSW07 [2] and NYO08 [12] (which is the second construction of the NYO08 paper). If a user wants to decrypt a ciphertext with an access structure including newly added attributes, then the user must obtain a new secret key (including newly added attributes) from the trusted authority again. However, the security proof of both schemes contains no reduction, namely, it is proven under the generic group heuristic.

4 Our Construction

In this section, we propose a constant ciphertext length CP-ABE with a function of adding new attributes after Setup. Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be cyclic groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear map. Let $\mathcal{U} = \{att_1, \ldots, att_n\}$ be a set of attributes; $S_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}$ be a set of possible values with $n_i = |S_i|$; $L = [L_1, L_2, \ldots, L_n]$ $(L_i \in S_i)$ be an attribute list for a user; and $W = [W_1, W_2, \ldots, W_n]$ $(W_i \in S_i)$ be an access structure.

4.1 Proposed Scheme

Protocol 2. ***Our CP-ABE Scheme with Constant Ciphertext Length***

$\mathsf{Setup}(1^\kappa)$**:** *A trusted authority TA chooses a prime number p, a bilinear group $(\mathbb{G}_1, \mathbb{G}_T)$ with order p, a generator $g_1 \in \mathbb{G}_1$, $h \in \mathbb{G}_1$, $y \in_R \mathbb{Z}_p$ and $t_{i,j} \in_R \mathbb{Z}_p$ $(i \in [1,n], j \in [1,n_i])$. TA computes $Y = e(g_1,h)^y$ and $T_{i,j} = g_1^{t_{i,j}}$ $(i \in [1,n], j \in [1,n_i])$. TA outputs $PK = (e, g_1, h, Y, \{T_{i,j}\}_{i\in[1,n],j\in[1,n_i]})$ and $MK = (y, \{t_{i,j}\}_{i\in[1,n],j\in[1,n_i]})$. Note that $\forall L, L'$ $(L \neq L')$, $\sum_{v_{i,j}\in L} t_{i,j} \neq \sum_{v_{i,j}\in L'} t_{i,j}$ is assumed*

$\mathsf{KeyGen}(PK, MK, L)$**:** *$TA$ chooses $r \in_R \mathbb{Z}_p$, outputs $SK_L = (h^y(g_1^{\sum_{v_{i,j}\in L} t_{i,j}})^r, g_1^r)$, and gives SK_L to a user with L.*

$\mathsf{Encrypt}(PK, M, W)$**:** *An encryptor chooses $s \in_R \mathbb{Z}_p$, and computes $C_1 = M \cdot Y^s$, $C_2 = g_1^s$ and $C_3 = (\prod_{v_{i,j}\in W} T_{i,j})^s$. The encryptor outputs $C = (W, C_1, C_2, C_3)$.*

$\mathsf{Decrypt}(PK, C, SK_L)$**:** *A decryptor computes what follows:*

$$\frac{C_1 \cdot e(C_3, g_1^r)}{e(C_2, h^y(g_1^{\sum_{v_{i,j}\in L} t_{i,j}})^r)} = \frac{M \cdot e(g,h)^{sy} e(g_1,g_1)^{sr\sum_{v_{i,j}\in W} t_{i,j}}}{e(g_1,h)^{sy} e(g_1,g_1)^{sr\sum_{v_{i,j}\in L} t_{i,j}}} = M$$

4.2 Construction of Secret Keys $t_{i,j}$

In our scheme, $\sum_{v_{i,j}\in L} t_{i,j} \neq \sum_{v_{i,j}\in L'} t_{i,j}$ is assumed. If there exist L and L' $(L \neq L')$ such that $\sum_{v_{i,j}\in L} t_{i,j} = \sum_{v_{i,j}\in L'} t_{i,j}$, a user with the attribute list L' can decrypt a ciphertext associated with W, where $L' \not\models W$ and $L \models W$. Remark that the assumption holds with overwhelming probability $\frac{p(p-1)\cdots(p-(N-1))}{p^N} > \frac{(p-(N-1))^N}{p^N} = (1 - \frac{N-1}{p})^N > 1 - \frac{N(N-1)}{p} > 1 - \frac{N^2}{p}$, where $N := \prod_{i=1}^n n_i$. Therefore, if each secret key $t_{i,j}$ is chosen at random from $\mathbb{Z}_p$, then our assumption is natural.

5 Security Analysis

Theorem 1. *Our scheme satisfies the indistinguishability of messages under the DBDH assumption.*

Proof. We suppose that the adversary $\mathcal{A}$ wins the selective game for CP-ABE with the advantage ϵ. Then we can construct an algorithm $\mathcal{B}$ that breaks the DBDH assumption with the advantage $\frac{\epsilon}{2}(1 - \frac{N^2}{p})$, where $N := \prod_{i=1}^{n} n_i$ is the number of expressed access structures. The DBDH challenger selects $a, b, c, z \in_R \mathbb{Z}_p$, $\nu \in_R \{0, 1\}$, and g_1, where $\langle g_1 \rangle = \mathbb{G}_1$. If $\nu = 0$, then $Z = e(g_1, g_1)^{abc}$. Otherwise, if $\nu = 1$, then $Z = e(g_1, g_1)^z$. The DBDH challenger gives the DBDH instance $(g_1, g_1^a, g_1^b, g_1^c, Z) \in \mathbb{G}_1^4 \times \mathbb{G}_T$ to $\mathcal{B}$. First, $\mathcal{B}$ is given the challenge access structure W^* from $\mathcal{A}$. Let $W^* = [W_1^*, \ldots, W_n^*]$. $\mathcal{B}$ selects $u \in_R \mathbb{Z}_p^*$, and sets $h = g_1^u$ and $Y = e(g_1^a, (g_1^b)^u) = e(g_1, h)^{ab}$. Moreover, $\mathcal{B}$ selects $t'_{i,j} \in_R \mathbb{Z}_p$ $(i \in [1, n], j \in [1, n_i])$, and sets $t_{i,j} = t'_{i,j}$ (in the case where $v_{i,j} = W_i^*$) and $t_{i,j} = bt'_{i,j}$ (in the case where $v_{i,j} \neq W_i^*$), and computes public keys $T_{i,j}$ $(i \in [1, n], j \in [1, n_i])$ as follows:

$$T_{i,j} = g_1^{t_{i,j}} = \begin{cases} g_1^{t'_{i,j}} & (v_{i,j} = W_i^*) \\ (g_1^b)^{t'_{i,j}} & (v_{i,j} \neq W_i^*) \end{cases}$$

$\mathcal{B}$ gives $PK = (e, g_1, h, Y, \{T_{i,j}\}_{i \in [1,n], j \in [1,n_i]})$ to $\mathcal{A}$. For KeyGen query L, there exists $v_{i,\ell}$ such that $v_{i,\ell} = L_i \wedge v_{i,\ell} \neq W_i^*$, since $L \not\models W^*$. Therefore, $\sum_{v_{i,j} \in L} t_{i,j}$ can be represented as $\sum_{v_{i,j} \in L} t_{i,j} = T_1 + bT_2$, where $T_1, T_2 \in \mathbb{Z}_p$. Note that both T_1 and T_2 are represented by the sum of $t'_{i,j}$. Therefore, $\mathcal{B}$ can compute T_1 and T_2. $\mathcal{B}$ chooses $\beta \in_R \mathbb{Z}_p$, sets $r := \frac{\beta - ua}{T_2}$, and computes $SK_L = ((g_1^b)^\beta g_1^{\frac{T_1}{T_2}\beta} (g_1^a)^{-\frac{T_1 u}{T_2}}, g_1^{\frac{\beta}{T_2}} (g_1^a)^{-\frac{u}{T_2}})$. We show that SK_L is a valid secret key as follows:

$$\begin{aligned}(g_1^b)^\beta g_1^{\frac{T_1}{T_2}\beta} (g_1^a)^{-\frac{T_1 u}{T_2}} &= g_1^{uab} \cdot g_1^{-uab} (g_1^b)^\beta g_1^{\frac{T_1}{T_2}\beta} (g_1^a)^{-\frac{T_1 u}{T_2}} \\ &= g_1^{uab} \cdot g_1^{\frac{T_1}{T_2}(\beta - ua)} \cdot g_1^{b(\beta - ua)} \\ &= g_1^{uab} (g_1^{T_1} \cdot g_1^{bT_2})^{\frac{\beta - ua}{T_2}} \\ &= g_1^{uab} (g_1^{T_1 + bT_2})^{\frac{\beta - ua}{T_2}} \\ &= h^y (g_1^{\sum_{v_{i,j} \in L} t_{i,j}})^r,\end{aligned}$$

and

$$g_1^{\frac{\beta}{T_2}} (g_1^a)^{-\frac{u}{T_2}} = g_1^{\frac{\beta - ua}{T_2}} = g_1^r$$

If $T_2 = 0 \bmod p$, then $\mathcal{B}$ aborts. If $T_2 = 0 \bmod p$ holds, then there exists L such that $\sum_{v_{i,j} \in L} t_{i,j} = \sum_{v_{i,j} \in W^*} t_{i,j}$ holds. Therefore, this probability is at most $\frac{N^2}{p}$. See Section 4.2 for details. For the challenge ciphertext, $\mathcal{B}$ chooses

Table 1. Size of each value

	PK	MK	SK	Ciphertext
SW05 [13]	$n\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(n+1)\|\mathbb{Z}_p\|$	$r_2\|\mathbb{G}_1\|$	$r_1\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
GPSW06 [9]	$n\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(n+1)\|\mathbb{Z}_p\|$	$r_2\|\mathbb{G}_1\|$	$r_1\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
CN07 [7]	$(3n+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(3n+1)\|\mathbb{Z}_p\|$	$(2n+1)\|\mathbb{G}_1\|$	$(n+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
BSW07 [2]	$3\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$\|\mathbb{Z}_p\| + \|\mathbb{G}\|$	$(2n+1)\|\mathbb{G}_1\|$	$(2r_2+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
NYO08 [12]	$(2N'+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(2N'+1)\|\mathbb{Z}_p\|$	$(3n+1)\|\mathbb{G}_1\|$	$(2N'+1)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
W08 [14]	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$\|\mathbb{G}_1\|$	$(1+n+r_2)\|\mathbb{G}_1\|$	$(1+r_1n)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$
Our scheme	$(2N'+3)\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(N'+1)\|\mathbb{Z}_p\|$	$2\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$

Table 2. Computational time of each algorithm

	Enc.	Dec.
SW05 [13]	$r_1\mathbb{G}_1 + 2\mathbb{G}_T$	$r_1C_e + (r_1+1)\mathbb{G}_T$
GPSW06 [9]	$r_1\mathbb{G}_1 + 2\mathbb{G}_T$	$r_1C_e + (r_1+1)\mathbb{G}_T$
CN07 [7]	$(n+1)\mathbb{G}_1 + 2\mathbb{G}_T$	$(n+1)C_e + (n+1)\mathbb{G}_T$
BSW07 [2]	$(2r_1+1)\mathbb{G}_1 + 2\mathbb{G}_T$	$2r_1C_e + (2r_1+2)\mathbb{G}_T$
NYO08 [12]	$(2N'+1)\mathbb{G}_1 + 2\mathbb{G}_T$	$(3n+1)C_e + (3n+1)\mathbb{G}_T$
W08 [14]	$(1+3r_1n)\mathbb{G}_1 + 2\mathbb{G}_T$	$(1+n+r_1)C_e + (3r_1-1)\mathbb{G}_1 + 3\mathbb{G}_T$
Our scheme	$(n+1)\mathbb{G}_1 + 2\mathbb{G}_T$	$2C_e + 2\mathbb{G}_T$

$\mu \in_R \{0,1\}$, computes $C_1^* = M_\mu \cdot Z^u$, $C_2^* = g_1^c$ and $C_3^* = (g_1^c)^{\sum_{v_{i,j} \in W^*} t'_{i,j}}$, and sends (C_1^*, C_2^*, C_3^*) to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $\mu' \in \{0,1\}$. $\mathcal{B}$ outputs 1 if $\mu' = \mu$, or outputs 0 if $\mu' \neq \mu$. If $Z = e(g_1, g_1)^{abc}$, then (C_1^*, C_2^*, C_3^*) is a valid ciphertext associated with W^*. Therefore, $\mathcal{A}$ has the advantage ϵ. Hence, $\Pr[\mathcal{B} \to 1 | Z = e(g_1, g_1)^{abc}] = \Pr[\mu' = \mu | Z = e(g_1, g_1)^{abc}] = \frac{1}{2} + \epsilon$. Otherwise, if $Z = e(g_1, g_1)^z$, $\mathcal{A}$ has no advantage to distinguish a bit μ, since all parts of the challenge ciphertext when $\mu = 0$ and when $\mu = 1$ have the same distributions. Hence, $\Pr[\mathcal{B} \to 0 | Z = e(g_1, g_1)^z] = \Pr[\mu' \neq \mu | Z = e(g_1, g_1)^z] = \frac{1}{2}$. It follows that $\mathcal{B}$'s advantage in the DBDH game is $\frac{\epsilon}{2}(1 - \frac{N^2}{p})$. $\square$

Although a symmetric bilinear map is required in this proof, our scheme can be proven with an asymmetric bilinear map such as the Weil or Tate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ over MNT curves [11], where $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct groups. Then the indistinguishability of messages can be proven under the DBDH assumption over $\mathbb{G}_2$ [1].

6 Comparison

Let PK, MK, SK and Ciphertext be the size of the public key, of the master key, of the secret key, and the ciphertext length excluding the access structure, respectively. Moreover, Enc. and Dec. are the computational times of encryption and decryption, respectively. We use the terms DBDH, DMBDH [13] and D-Linear [12] to refer to the Decision Bilinear Diffie-Hellman assumption, the

Table 3. Some properties of ABE schemes

	Policy	Recipient Anonymity	Assumption
SW05 [13]	Key	No	DMBDH
GPSW06 [9]	Key	No	DBDH
CN07 [7]	Ciphertext	No	DBDH
BSW07 [2]	Ciphertext	No	Generic Group Model
NYO08 [12]	Ciphertext	Yes	DBDH, D-Linear
W08 [14]	Ciphertext	No	DBDH
Our scheme	Ciphertext	No	DBDH

Table 4. Expressiveness of policy

SW05 [13]	Threshold Structure
GPSW06 [9]	Tree-based Structure
CN07 [7]	AND-gates on positive and negative attributes with wildcards
BSW07 [2]	Tree-Based Structure
W08 [14]	Linear Structure
NYO08 [12]	AND-gates on multi-valued attributes with wildcards
Our scheme	AND-gates on multi-valued attributes

Table 5. Performance Results for $n = 3$

	Enc. Time	Dec. Time
CN07 [7]	0.028sec	0.031sec
NYO08 [12]	0.032sec	0.078sec
Our scheme	0.015sec	0.015sec

Decision Modified Bilinear Diffie-Hellman assumption and the Decision Linear assumption, respectively. The notation $|\mathbb{G}|$ is the bit-length of the element which belongs to $\mathbb{G}$. Let the notations $k\mathbb{G}$ and kC_e (where $k \in \mathbb{Z}_{>0}$) be the k-times calculation over the group $\mathbb{G}$ and pairing, respectively. Let $\mathcal{U} = \{att_1, att_2, \ldots, att_n\}$ be the set of attributes. Let γ_1 ($|\gamma_1| = r_1$) be a set of attributes associated with the ciphertext, and γ_2 ($|\gamma_2| = r_2$) a set of attributes associated with the secret key. Actually, γ_2 is different for each user. Let $N' := \sum_{i=1}^{n} n_i$ be the total number of possible statements of attributes. The computational time over $\mathbb{Z}_p$ is ignored as usual.

Our scheme is efficient in that the ciphertext length and the costs of decryption do not depend on the number of attributes. Especially, the number of pairing computations is constant. No previous schemes provide these properties. An access structure is constructed by AND-gates on multi-valued attributes defined in section 2.2, which is a subset of the access structures in [12]. Although previous CP-ABE schemes [2,7,8,12,14] can complement our access structures, the length of the ciphertext depends on the number of attributes. To the best of our knowledge, our scheme is the first constant ciphertext length CP-ABE with

AND-gates on multi-valued attributes. In future work, we plan to construct a CP-ABE with both a constant ciphertext length and more flexible structures, such as linear structures.

Our scheme does not provide recipient anonymity. Some parts of a ciphertext for attributes $(C_2, C_3) = (g_1^s, g_1^{s\sum_{v_{i,j} \in W} t_{i,j}})$ is a DDH (Decision Diffie-Hellman)-tuple. Therefore, some information about attributes is exposed. Concretely, for an access structure W', an attacker can run the DDH test $e(C_2, \prod_{v_{i,j} \in W'} T_{i,j}) \overset{?}{=} e(C_3, g_1)$. Then, the attacker can determine whether an encryptor used the policy W' or not. We expect that our scheme will enable the property of the hidden encryptor-specified policies when a DDH-hard bilinear group is applied. However, we could not give the proof of security. Added to this, our scheme is inefficient in that the size of public key grows linearly with the number of attributes. There are rooms for argument on these points.

The CN07 scheme [7], the NYO08 scheme [12] and ours are implemented with *the same access structure* $\{v_{1,1}, v_{2,1}, v_{3,1}\}$, by using the Pairing-Based Cryptography (PBC) Library ver. 0.4.18 [15]. The performance results are shown in Table 5. Our experiment was performed by using a PC with an Intel(R) Core(TM)2 Duo CPU P8400 2.26GHz Windows Vista Home Premium Edition Service Pack 1. The execution of our scheme takes a very small amount of time, which is quite feasible for practical implementation. When $n = 3$, our decryption algorithm is approximately twice as fast as that of the CN07 scheme, and approximately five times faster than that of the NYO08 scheme.

7 Conclusion

In this paper, we propose a constant ciphertext length CP-ABE with AND-gates on multi-valued attributes. Moreover, the number of pairing computations is also constant. To the best of our knowledge, this is the first such construction.

References

1. Abdalla, M., Dent, A.W., Malone-Lee, J., Neven, G., Phan, D.H., Smart, N.P.: Identity-based traitor tracing. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 361–376. Springer, Heidelberg (2007)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. 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)
5. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)

6. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
7. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 456–465. ACM Press, New York (2007)
8. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
9. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
10. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
11. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for fr-reduction. IEICE transactions on fundamentals of electronics, communications and computer sciences 84(5), 1234–1243 (2001)
12. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008)
13. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
14. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Cryptology ePrint report 2008/290 (September 1, 2008)
15. The Pairing-Based Cryptography (PBC) Library, http://crypto.stanford.edu/pbc/

RSA-Based Certificateless Public Key Encryption

Junzuo Lai[1], Robert H. Deng[2], Shengli Liu[1], and Weidong Kou[3]

[1] Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200030, China
{laijunzuo,slliu}@sjtu.edu.cn
[2] School of Information Systems,
Singapore Management University, Singapore 178902
robertdeng@smu.edu.sg
[3] School of Computer Science and Technology
Xi Dian University, Xi'an 710071, China
kou_weidong@yahoo.com.cn

Abstract. Certificateless Public Key Cryptography was first introduced by Al-Riyami and Paterson in order to eliminate the inherent key-escrow problem of Identity-Based Cryptography. In this paper, we present a new practical construction of certificateless public key encryption scheme without paring. Our scheme is, in the random oracle model, provably secure under the assumption that the RSA problem is intractable.

Keywords: Certificateless public key encryption, RSA.

1 Introduction

In order to solve the key escrow problem that is inherent in identity-based cryptography (IBC) [20], while at the same time, eliminate the use of certificates in the traditional public key cryptography (PKC), Al-Riyami and Paterson [1] introduced the concept of certificateless public key cryptography (CL-PKC). Different from IBC, a user's public key in CL-PKC is no longer an arbitrary string; instead, the public key is generated by the user based the user's secret information as well as a partial private key obtained from a trusted authority called Key Generation Center (KGC). As such, public keys in CL-PKC do not need to be explicitly certified. Note here that the KGC does not know the user's private keys since they contain secret information generated by the users themselves, thereby removing the escrow problem in IBC.

Since the introduction of CL-PKC [1], many concrete constructions of certificateless public key encryption (CL-PKE) schemes have been proposed. The schemes in [3,6,21,22] were proven secure in the random oracle model [4] while the schemes in [14] and [18] are secure without the random oracles.

There were also efforts to construct generic CL-PKC schemes. The first generic CL-PKE scheme was proposed in [23] and was later shown in [16] to be insecure under the model of [1]. In [5], the authors extended the concept of key encapsulation mechanism to IBE and CL-PKE, and built generic constructions of identity-based key encapsulation mechanism and certificateless public key encapsulation mechanism.

One notable feature in the research of CL-PKE has been the development of a number of alternative security models that are substantially weaker than the original model of [1]. These different models are summarized by Dent [7]. Moreover, Dent et al. [8] presents a generic construction as well as a concrete construction for certificateless encryption schemes that are provably secure against strong adversaries in the standard model.

Au et al. [2] pointed out the weakness of the previous security models and analyzed some previous schemes under an enhanced malicious KGC model. They showed that the CL-PKE scheme in [16] is secure against malicious KGC attacks under random oracle assumption. Hwang and Liu [13] proposed a new CL-PKE scheme which is secure against malicious KGC attacks. Its security is proven in the standard model. In addition, Huang and Wong [12] proposed a generic construction of certificateless encryption which is proven secure against malicious KGC attacks in the standard model.

Other Related Work. Gentry [10] introduced a different but related concept named certificate based encryption (CBE). This approach is closer to the context of a traditional PKI model as it involves a certification authority (CA) providing an implicit certification service for clients' public keys. Liu et al. proposed the first self-generated-certificate public key encryption (SGC-PKE) scheme in [14], which defends the DoD attack that exists in CL-PKE. Lai and Kou [15] proposed a SGC-PKE scheme without using pairing.

Contribution. In spite of the recent advances in implementation technique, the paring computation is still considered as expensive compared with the "standard" operations such as modular exponentiations in finite fields. Baek et al. [3] proposed the first CL-PKE scheme without pairing, which was related to the early works on the self-certified keys [11,19].

In this paper, inspired by the identity-based key agreement protocol proposed by Okamoto and Tanaka [17] and whose security relies on the RSA problem, we present a new CL-PKE scheme without paring. Due to the extensive deployment of RSA, our scheme is better off in compatibility with the existing cryptosystems. In addition, in [3], the Type I adversary is not allowed to replace the challenge identity's public key, which is the main attacking means of Type I adversary. Compared with the scheme in [3], our scheme does not have this limitation.

Organization. The rest of the paper is organized as follow. We give some related definitions in Section 2. The model of CL-PKE is also reviewed in this section. The proposed CL-PKE scheme and its security analysis is presented in Section 3. Finally concluding remarks are given in Section 4.

2 Preliminaries

2.1 Computational Problems

Definition 1. *The RSA problem is, given a randomly generated RSA modulus n, an exponent e and a random z, to find $y \in \mathbb{Z}_n^*$ such that $y^e = z$.*

Definition 2. *The Computational Diffie-Hellman (CDH) problem in $\mathbb{Z}_n^*$ is, given p,q,n (where $p = 2p'+1, q = 2q'+1, n = pq$ with p', q' being two equal-length large primes), $g \in \mathbb{Z}_n^*$ of order $p'q'$, g^a and g^b for uniformly chosen $a, b \in \mathbb{Z}_n^*$, to compute g^{ab}.*

2.2 Certificateless Public Key Encryption

A generic CL-PKE is a tuple of algorithm described as follows [3]:

- Setup: Takes as input a security parameter κ and outputs a common parameter *params* and a master secret *msk*.
- PartialKeyExtract: Takes as input *params*, *msk* and an identity ID. It outputs a partial private key d_{ID}.
- SetSecretValue: Takes as input *params* and an identity ID. It outputs a secret value s_{ID}.
- SetPrivateKey: Takes as input *params*, d_{ID} and s_{ID}. It outputs a private key $\mathsf{SK}_{\mathsf{ID}}$.
- SetPublicKey: Takes as input *params*, d_{ID} and s_{ID}. It outputs a public key $\mathsf{PK}_{\mathsf{ID}}$.
- Enc: Takes as input *params*, a message m, a receiver's identity ID and $\mathsf{PK}_{\mathsf{ID}}$. It outputs a ciphertext c.
- Dec: Takes as input *params*, $\mathsf{SK}_{\mathsf{ID}}$ and a ciphertext c. It outputs a message m or the failure symbol $\perp$.

We insist that CL-PKE satisfies the obvious correctness requirement that decapsulation "undoes" encapsulation.

Note that, the above model of CL-PKE is slightly weaker than the original one given in [1] as a user must authenticate herself to the KGC in order to obtain a partial private key to create a public key, while the original CL-PKE model does not require a user to contact the KGC to setup her public keys. However, as argued in [3], this modified model preserves the unique property of CL-PKE that no certificates are required in order to guarantee the authenticity of public keys, which is the main motivation of CL-PKE.

Security Model. There are two types of adversaries [1]. Type I adversary models an "outsider" adversary, who does not have the KGC's master secret key but it can replace public keys of arbitrary identities with other public keys of its own choices. It can also obtain partial and full secret keys of arbitrary identities. Type II adversary models an "honest-but-curious" KGC, who knows the master secret key (hence it can compute partial secret key by itself). It is still allowed to obtain full secret key for arbitrary identities but is not allowed to replace public keys at any time.

Security in CL-PKE is defined using the following game between an attack algorithm $\mathcal{A}$ and a challenger.

Setup. The challenger runs the Setup algorithm and gives $\mathcal{A}$ the resulting system parameter *params*. If $\mathcal{A}$ is of Type I, the challenger keeps the master secret key *msk* to itself; otherwise, it gives *msk* to $\mathcal{A}$.

Query phase 1 The adversary $\mathcal{A}$ adaptively issues the following queries:

- **Public-Key-Request** query: On input an identity ID, the challenger runs $\mathsf{SetPublicKey}(params, d_{\mathsf{ID}}, s_{\mathsf{ID}})$, where the partial private key d_{ID} and the secret value s_{ID} of the identity ID are obtained from $\mathsf{PartialKeyExtract}$ and $\mathsf{SetSecretValue}$, respectively, and forwards the result to the adversary.
- **Partial-Key-Extract** query: On input an identity ID, the challenger runs $\mathsf{PartialKeyExtract}(params, msk, \mathsf{ID})$ and returns the result to $\mathcal{A}$. Note that it is only useful to Type I adversary.
- **Private-Key-Request** query: On input an identity ID, the challenger runs $\mathsf{SetPrivateKey}(params, d_{\mathsf{ID}}, s_{\mathsf{ID}})$, where the partial private key d_{ID} and the secret value s_{ID} of the identity ID are obtained from $\mathsf{PartialKeyExtract}$ and $\mathsf{SetSecretValue}$, respectively, and forwards the result to the adversary. It outputs $\perp$ if the uesr's public key has been replaced in the case of Type I adversary.
- **Public-Key-Replace** query (for Type I adversary only): On input an identity and a valid public key, it replaces the associated user's public key with the new one.
- **Dec** query: On input a ciphertext and an identity, returns the decrypted message using the private key corresponding to the current value of the public key associated with the identity of the user.

Challenge query: After making a polynomial number of queries, $\mathcal{A}$ outputs two messages m_0, m_1 and an identity ID^*. The challenger picks a random bit $\beta \in \{0, 1\}$, sets $c^* = \mathsf{Enc}(parmas, m_\beta, \mathsf{ID}^*, \mathsf{PK}_{\mathsf{ID}^*})$ and sends c^* to $\mathcal{A}$.

Query phase 2 $\mathcal{A}$ makes a new sequence of queries.

Guess $\mathcal{A}$ outputs a bit $\beta^{'}$. It wins the game if $\beta^{'} = \beta$ under the following conditions:

- At any time, ID^* has not been submitted to the **Private-Key-Request** query.
- $(c^*, \mathsf{ID}^*, \mathsf{PK}_{\mathsf{ID}^*})$ have not been submitted to the **Dec** query.
- If it is Type I adversary, ID^* cannot be equal to an identity for which both the public key has been replaced and the partial private key has been extracted.

We define $\mathcal{A}$'s advantage in attacking the certificateless public key encryption $\mathsf{CL\text{-}PKE}$ as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CL\text{-}PKE}} = |\mathsf{Pr}[\beta = \beta^{'}] - \frac{1}{2}|.$$

Definition 3. *We say that a certificateless public key encryption* CL-PKE *is* $(t, q_{pub}, q_{par}, q_{prv}, q_d, \epsilon)$-IND-CCA *secure against Type I (resp. Type II) adversary* $\mathcal{A}_I$ *(resp.* $\mathcal{A}_{II}$*), if for all t-time algorithms* $\mathcal{A}_I$ *(resp.* $\mathcal{A}_{II}$*) making at most* q_{pub} ***Public-Key-Request*** *queries,* q_{par} ***Partial-Key-Extract*** *queries,* q_{prv} ***Private-Key-Request*** *queries and* q_d ***Dec*** *queries, have advantage at most* ϵ *in winning the above game.*

IND-CPA security is defined similarly, but with the restriction that the adversary cannot make **Dec** queries.

Definition 4. *We say that a certificateless public key encryption* CL-PKE *is* $(t, q_{pub}, q_{par}, q_{prv}, \epsilon)$-IND-CPA *secure, if it is* $(t, q_{pub}, q_{par}, q_{prv}, 0, \epsilon)$-IND-CCA *secure.*

3 Our Scheme

Our CL-PKE scheme is inspired by the RSA-based key agreement protocol [17] introduced by Okamoto and Tanaka. We first present our scheme and then show that it is IND-CPA secure. However, it is easy to turn our IND-CPA secure CL-PKE scheme into an IND-CCA secure CL-PKE scheme using the technique proposed by Fujisaki and Okamoto [9], as did in [3].

Setup(κ) Given a security parameter κ, a RSA group $< n, p, q, e, d, g >$ is generated, where p', q' are κ-bit prime numbers, $p = 2p' + 1, q = 2q' + 1, n = pq, e < \phi(n), \gcd(e, \phi(n)) = 1, ed \equiv 1(\text{mod } \phi(n))$, and ϕ denotes the Euler totient function. Chooses two cryptographic hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_n^*, H_2 : \mathbb{Z}_n^* \rightarrow \{0,1\}^l$, where l is the length of the plaintext message. The master secret key is defined as $msk = d$. The common parameter is $params = (n, e, H, H_2)$.

PartialKeyExtract($params, msk$, ID) Given $params$, $msk = d$ and an identity $\mathsf{ID} \in \{0,1\}^*$, outputs the partial private key

$$d_{\mathsf{ID}} = H(\mathsf{ID})^d.$$

SetSecretValue($params$, ID) Given $params$ and an identity ID, randomly chooses $x_{\mathsf{ID}} \in \mathbb{Z}_n^*$ and outputs

$$s_{\mathsf{ID}} = x_{\mathsf{ID}}.$$

SetPrivateKey($params, d_{\mathsf{ID}}, s_{\mathsf{ID}}$) Given $params$, the partial private key d_{ID} and the secret value $s_{\mathsf{ID}} = x_{\mathsf{ID}}$ of an identity ID, outputs

$$\mathsf{SK}_{\mathsf{ID}} = x_{\mathsf{ID}}.$$

SetPublicKey($params, d_{\mathsf{ID}}, s_{\mathsf{ID}}$) Given $params$, the partial private key $d_{\mathsf{ID}} = H(\mathsf{ID})^d$ and the secret value $s_{\mathsf{ID}} = x_{\mathsf{ID}}$ of an identity ID, outputs

$$\mathsf{PK}_{\mathsf{ID}} = H(\mathsf{ID})^{d + x_{\mathsf{ID}}}.$$

$\mathsf{Enc}(params, m, \mathsf{ID}, \mathsf{PK}_{\mathsf{ID}})$ Given *params*, a message m and the public key $\mathsf{PK}_{\mathsf{ID}}$ of an identity ID, randomly chooses $r \in \mathbb{Z}_n^*$ and computes

$$c_1 = H(\mathsf{ID})^{er}, c_2 = H_2(\mathsf{PK_{ID}}^{er} H(\mathsf{ID})^{-r}) \oplus m,$$

then outputs $c = (c_1, c_2)$.

$\mathsf{Dec}(params, \mathsf{SK}_{\mathsf{ID}}, c)$ Given *params*, the private key $\mathsf{SK}_{\mathsf{ID}}$ of an identity ID, and a ciphertext $c = (c_1, c_2)$, outputs

$$m = H_2(c_1^{\mathsf{SK}_{\mathsf{ID}}}) \oplus c_2.$$

It can be easily seen that the above decryption algorithm is consistent, i. e.,

$$\begin{aligned}\mathsf{PK_{ID}}^{er} H(\mathsf{ID})^{-r} &= H(\mathsf{ID})^{(d+x_{\mathsf{ID}})er} H(\mathsf{ID})^{-r} \\ &= H(\mathsf{ID})^{r} H(\mathsf{ID})^{erx_{\mathsf{ID}}} H(\mathsf{ID})^{-r} \\ &= c_1^{x_{\mathsf{ID}}} = c_1^{\mathsf{SK}_{\mathsf{ID}}}.\end{aligned}$$

We now prove the security of the scheme by two theorems.

Theorem 1. *Assume the hash functions H, H_2 are random oracles and the RSA problem is (t, ϵ)-intractable. Then, the above CL-PKE scheme is $(t', q_{pub}, q_{par}, q_{prv}, \epsilon')$IND-CPA secure against Type I adversary $\mathcal{A}_I$ for*

$$t > t' + t_{ex}(q_H + q_{pub}), \epsilon > \frac{2\epsilon'}{q_{H_2}\tau(q_{par} + q_{prv} + 1)},$$

where t_{ex} denotes the time for computing exponentiation in $\mathbb{Z}_n^*$, τ denotes the base of the natural logarithm and q_H (resp. q_{H_2}) denotes the number of H (resp. H_2) queries by the adversary.

Proof. Let $\mathcal{A}_I$ be a Type I adversary that $(t', q_{pub}, q_{par}, q_{prv}, \epsilon')$-breaks the IND-CPA security of the certificateless public key encryption scheme described above. We construct an algorithm $\mathcal{B}$, that solves the RSA problem, as follows. $\mathcal{B}$ is given an instance of the RSA problem, which consists of (n, e, z). $\mathcal{B}$'s goal is to find $y \in \mathbb{Z}_n^*$ such that $y^e = z$. It interacts with $\mathcal{A}_I$ as follows.

Setup $\mathcal{B}$ maintains three lists H-List,H$_2$-List and KeyList. Initially the lists are empty. The common parameter $params = (n, e)$ is sent to $\mathcal{A}_I$. The master secret key $msk = d$, where $ed \equiv 1 (\mathrm{mod}\ \phi(n))$, is unknown to $\mathcal{B}$.

Query phase 1 $\mathcal{A}_I$ adaptively issues H, H$_2$, **Public-Key-Request**, **Partial-Key-Extract**, **Private-Key-Request** and **Public-Key-Replace** queries. $\mathcal{B}$ answers them as follows:

– H query on ID: If a record $(\mathsf{ID}, h_{\mathsf{ID}}, f_{\mathsf{ID}}, coin)$ appears in the H-List, sends h_{ID} to $\mathcal{A}_I$; otherwise, $\mathcal{B}$ picks $coin \in \{0, 1\}$ at random such that $\Pr[coin = 0] = \rho$. (ρ will be determined later.) Then, randomly chooses $f_{\mathsf{ID}} \in \mathbb{Z}_n^*$. Finally, the record $(\mathsf{ID}, h_{\mathsf{ID}} = z^{coin} \cdot f_{\mathsf{ID}}^e, f_{\mathsf{ID}}, coin)$ is added to the H-List and h_{ID} is sent to $\mathcal{A}_I$.

- H_2 query on ω: If a record (ω, k) appears in the H_2-List, sends k to $\mathcal{A}_I$; otherwise, picks $k \in \{0,1\}^l$ at random, adds the record (ω, k) to H_2-List and sends k to $\mathcal{A}_I$.
- **Public-Key-Request** query on ID: Randomly chooses $x_{\mathsf{ID}} \in \mathbb{Z}_n^*$ and searches H-List for a record $(\mathsf{ID}, h_{\mathsf{ID}}, f_{\mathsf{ID}}, coin)$. Then, adds the record $(\mathsf{ID}, \mathsf{PK}_{\mathsf{ID}} = f_{\mathsf{ID}} h_{\mathsf{ID}}^{x_{\mathsf{ID}}}, \mathsf{SK}_{\mathsf{ID}} = x_{\mathsf{ID}}, coin)$ to $\mathsf{KeyList}$ and sends $\mathsf{PK}_{\mathsf{ID}}$ to $\mathcal{A}_I$.
- **Partial-Key-Extract** query on ID: Searches H-List for a record $(\mathsf{ID}, h_{\mathsf{ID}}, f_{\mathsf{ID}}, coin)$. If $coin = 0$, sends f_{ID} to $\mathcal{A}_I$; otherwise, aborts and terminates.
- **Private-Key-Extract** query on ID: Searches $\mathsf{KeyList}$ for a record $(\mathsf{ID}, \mathsf{PK}_{\mathsf{ID}}, \mathsf{SK}_{\mathsf{ID}}, coin)$. If $coin = 0$, sends $\mathsf{SK}_{\mathsf{ID}}$ to $\mathcal{A}_I$; otherwise, aborts and terminates.
- **Public-Key-Replace** query on $(\mathsf{ID}, \mathsf{PK}'_{\mathsf{ID}})$: Replaces $\mathsf{PK}_{\mathsf{ID}}$ with $\mathsf{PK}'_{\mathsf{ID}}$.

Challenge $\mathcal{A}_I$ submits two messages m_0, m_1 and an identity ID^* with the public key $\mathsf{PK}_{\mathsf{ID}^*}$. $\mathcal{B}$ searches H-List for a record $(\mathsf{ID}^*, h_{\mathsf{ID}^*}, f_{\mathsf{ID}^*}, coin)$. If $coin = 0$, it aborts and terminates; otherwise, $\mathcal{B}$ picks $r \in \mathbb{Z}_n^*$ at random. Let $r^* = d + r$, which is unknown to $\mathcal{B}$. Then $\mathcal{B}$ randomly chooses $c_2^* \in \{0,1\}^l$ and computes

$$c_1^* = H(\mathsf{ID}^*)^{er^*} = H(\mathsf{ID}^*)^{e(d+r)} = h_{\mathsf{ID}^*}^{1+er}.$$

Finally, it sends $c^* = (c_1^*, c_2^*)$ to $\mathcal{A}_I$.

Query phase 2 $\mathcal{A}_I$ makes a new sequence of queries, and $\mathcal{B}$ responds as in Query phase 1.

Guess: Finally, the adversary $\mathcal{A}_I$ outputs a bit β'. $\mathcal{B}$ picks a tuple (ω, k) from H_2-List at random and outputs $\frac{\mathsf{PK}_{\mathsf{ID}^*}^{1+er}}{\omega h_{\mathsf{ID}^*}^{r} f_{\mathsf{ID}^*}}$ as the solution to the RSA problem.

Probability Analysis: Let AskH_2^* denotes the event that $\mathsf{PK}_{\mathsf{ID}^*}^{er^*} H(\mathsf{ID}^*)^{-r^*}$ has been queried to H_2. Note that,

$$\begin{aligned}
\mathsf{PK}_{\mathsf{ID}^*}^{er^*} H(\mathsf{ID}^*)^{-r^*} &= \mathsf{PK}_{\mathsf{ID}^*}^{e(d+r)} H(\mathsf{ID}^*)^{-d-r} \\
&= \mathsf{PK}_{\mathsf{ID}^*}^{1+er} h_{\mathsf{ID}^*}^{-d} h_{\mathsf{ID}^*}^{-r} \\
&= \mathsf{PK}_{\mathsf{ID}^*}^{1+er} (z f_{\mathsf{ID}^*}^{e})^{-d} h_{\mathsf{ID}^*}^{-r} \\
&= \mathsf{PK}_{\mathsf{ID}^*}^{1+er} z^{-d} f_{\mathsf{ID}^*}^{-1} h_{\mathsf{ID}^*}^{-r}.
\end{aligned}$$

If the event AskH_2^* happens, then $\mathcal{B}$ will be able to solve the RSA problem by choosing a tuple (ω, k) from the H_2-List and computing $\frac{\mathsf{PK}_{\mathsf{ID}^*}^{1+er}}{\omega h_{\mathsf{ID}^*}^{r} f_{\mathsf{ID}^*}}$ with the probability at least $\frac{1}{q_{H_2}}$, where q_{H_2} is the number of H_2 queries by the adversary. If the event AskH_2^* does not happen, $\mathcal{B}$'s simulations are perfect and are identically distributed as the real one from the construction.

We observe that the probability that $\mathcal{B}$ does not abort during the simulation is given by $\rho^{q_{par}+q_{prv}}(1-\rho)$ which is maximized at $\rho = 1 - 1/(q_{par} + q_{prv} + 1)$. Hence the probability that $\mathcal{B}$ does not abort is at most $\frac{1}{\tau(q_{par}+q_{prv}+1)}$, where τ denotes the base of the natural logarithm.

Now, the event $\mathsf{AskH}_2^* | \neg \mathsf{Abort}$ denoted by Good, where Abort denotes the event that $\mathcal{B}$ aborts during the simulation. If Good dose not happen, it is clear that the adversary does not gain any advantage greater than 1/2 to guess β. Namely, we have $\mathsf{Pr}[\beta' = \beta | \neg \mathsf{Good}] \leq 1/2$. Hence, by splitting $\mathsf{Pr}[\beta' = \beta]$, we obtain $|\mathsf{Pr}[\beta' = \beta] - \frac{1}{2}| \leq \frac{1}{2}\mathsf{Pr}[\mathsf{Good}]$. To sum up, we have $\epsilon > \frac{2\epsilon'}{q_{H_2}\tau(q_{par}+q_{prv}+1)}$.

Time Complexity. In the simulation, $\mathcal{B}$'s overhead is dominated by the exponentiation computation in response to $\mathcal{A}_I$'s H and **Public-Key-Request** queries. So, we have $t > t' + t_{ex}(q_H + q_{pub})$, where t_{ex} denotes the time for computing exponentiation in $\mathbb{Z}_n^*$.

This concludes the proof of Theorem 1.

Theorem 2. *Assume the hash functions H and H_2 are random oracles and the CDH problem is (t, ϵ)-intractable. Then, the above CL-PKE scheme is $(t', q_{pub}, q_{par}, q_{prv}, \epsilon')$ IND-CPA secure against Type II adversary $\mathcal{A}_{II}$ for*

$$t > t' + t_{ex}(q_H + q_{pub}), \epsilon > \frac{2\epsilon'}{q_{H_2}\tau(q_{prv}+1)},$$

where t_{ex} denotes the time for computing exponentiation in $\mathbb{Z}_n^*$, τ denotes the base of the natural logarithm and q_H (resp. q_{H_2}) denotes the number of H (resp. H_2) queries by the adversary.

Proof. Let $\mathcal{A}_{II}$ be a Type II adversary that $(t', q_{pub}, q_{par}, q_{prv}, \epsilon')$-breaks the IND-CPA security of the CL-PKE scheme described above. We construct an algorithm $\mathcal{B}$, that solves the CDH problem, as follows. $\mathcal{B}$ is given an instance of the CDH problem, which consists of (n, p, q, g, g^a, g^b). $\mathcal{B}$'s goal is to compute g^{ab}. It interacts with $\mathcal{A}_{II}$ as follows.

Setup $\mathcal{B}$ maintains three lists H-List, H_2-List and KeyList. Initially the lists are empty. Then $\mathcal{B}$ picks $e < \phi(n), \gcd(e, \phi(n)) = 1$ at random and computes d such that $ed \equiv 1 (\mathrm{mod}\ \phi(n))$, where ϕ denotes the Euler totient function. (It can be computed by p, q.) Finally, it sends the common parameter $params = (n, e)$ and the master secret key $msk = d$ to $\mathcal{A}_{II}$.

Query phase 1 $\mathcal{A}_{II}$ adaptively issues H, H_2, **Public-Key-Request** and **Private-Key-Request** queries. $\mathcal{B}$ answers them in the following way:

- H query on ID: If a record $(\mathsf{ID}, h_{\mathsf{ID}}, t_{\mathsf{ID}})$ appears in the H-List, $\mathcal{B}$ sends h_{ID} to $\mathcal{A}_I$; otherwise, $\mathcal{B}$ randomly chooses t_{ID} such that $t_{\mathsf{ID}} < \phi(n)$, $\gcd(t_{\mathsf{ID}}, \phi(n)) = 1$, adds the record $(\mathsf{ID}, h_{\mathsf{ID}} = g^{t_{\mathsf{ID}}}, t_{\mathsf{ID}})$ to H-List and sends h_{ID} to $\mathcal{A}_{II}$.
- H_2 query on ω: If a record (ω, k) appears in the H_2-List, $\mathcal{B}$ sends k to $\mathcal{A}_I$: otherwise, $\mathcal{B}$ picks $k \in \{0,1\}^l$ at random, adds the record (ω, k) to H_2-List and sends k to $\mathcal{A}_{II}$.
- **Public-Key-Request** query on ID: $\mathcal{B}$ searches H-List for a record $(\mathsf{ID}, h_{\mathsf{ID}}, t_{\mathsf{ID}})$. Then, it picks $coin \in \{0,1\}$ at random such that $\mathsf{Pr}[coin = 0] = \rho$ (ρ will be determined later). Finally, it randomly chooses $x_{\mathsf{ID}} \in \mathbb{Z}_n^*$, adds the record $(\mathsf{ID}, \mathsf{PK}_{\mathsf{ID}} = h_{\mathsf{ID}}^{d+x_{\mathsf{ID}}} \cdot (g^a)^{t_{\mathsf{ID}} \cdot coin} = h_{\mathsf{ID}}^{d+a\cdot coin+x_{\mathsf{ID}}}, \mathsf{SK}_{\mathsf{ID}} = x_{\mathsf{ID}}, coin)$ to KeyList and sends $\mathsf{PK}_{\mathsf{ID}}$ to $\mathcal{A}_{II}$.

- **Private-Key-Extract** query on ID: $\mathcal{B}$ searches $\mathsf{KeyList}$ for a record $(\mathsf{ID}, \mathsf{PK}_{\mathsf{ID}}, \mathsf{SK}_{\mathsf{ID}}, coin)$. If $coin = 0$, it sends the $\mathsf{SK}_{\mathsf{ID}}$ to $\mathcal{A}_{II}$; otherwise, it aborts and terminates.

Challenge $\mathcal{A}_{II}$ submits two messages m_0, m_1 and an identity ID^* with the public key $\mathsf{PK}_{\mathsf{ID}^*}$. $\mathcal{B}$ searches H-List for a record $(\mathsf{ID}^*, h_{\mathsf{ID}^*}, t_{\mathsf{ID}^*})$ and $\mathsf{KeyList}$ for a record $(\mathsf{ID}^*, \mathsf{PK}_{\mathsf{ID}^*}, \mathsf{SK}_{\mathsf{ID}^*} = x_{\mathsf{ID}^*}, coin)$. If $coin = 0$, it aborts and terminates; otherwise, $\mathcal{B}$ randomly chooses $c_2^* \in \{0,1\}^l$. Let $r^* = b$, which is unknown to $\mathcal{B}$. Then $\mathcal{B}$ computes

$$c_1^* = (g^b)^{et_{\mathsf{ID}^*}} = (g^{t_{\mathsf{ID}^*}})^{er^*} = h_{\mathsf{ID}^*}^{er^*} = H(\mathsf{ID}^*)^{er^*}.$$

and sends $c^* = (c_1^*, c_2^*)$ to $\mathcal{A}_{II}$.

Query phase 2 $\mathcal{A}_{II}$ makes a new sequence of queries, and $\mathcal{B}$ responds as in Query phase 1.

Guess Finally, the adversary $\mathcal{A}_{II}$ outputs a bit β'. $\mathcal{B}$ picks a tuple (ω, k) from H_2-List at random and outputs $\frac{\omega^{dt_{\mathsf{ID}^*}^{-1}}}{(g^b)^{x_{\mathsf{ID}^*}}}$ as the solution to the CDH problem. Note that, $\mathcal{B}$ knows p, q, so $t_{\mathsf{ID}^*}^{-1}$ can be computed.

Probability Analysis: Let AskH_2^* denotes the event that $\mathsf{PK}_{\mathsf{ID}^*}^{er^*} H(\mathsf{ID}^*)^{-r^*}$ has been queried to H_2. Note that,

$$\begin{aligned}\mathsf{PK}_{\mathsf{ID}^*}^{er^*} H(\mathsf{ID}^*)^{-r^*} &= \mathsf{PK}_{\mathsf{ID}^*}^{eb} H(\mathsf{ID}^*)^{-b} \\ &= h_{\mathsf{ID}^*}^{eb(d+a+x_{\mathsf{ID}^*})} h_{\mathsf{ID}^*}^{-b} \\ &= h_{\mathsf{ID}^*}^{eb(a+x_{\mathsf{ID}^*})} \\ &= (g^{ab})^{et_{\mathsf{ID}^*}} (g^b)^{et_{\mathsf{ID}^*} x_{\mathsf{ID}^*}}.\end{aligned}$$

If the event AskH_2^* happens, then $\mathcal{B}$ will be able to solve the CDH problem by choosing a tuple (ω, k) from the H_2-List and computing $\frac{\omega^{dt_{\mathsf{ID}^*}^{-1}}}{(g^b)^{x_{\mathsf{ID}^*}}}$ with the probability at least $\frac{1}{q_{H_2}}$, where q_{H_2} is the number of H_2 queries by the adversary. If the event AskH_2^* does not happen, $\mathcal{B}$'s simulations are perfect and are identically distributed as the real one form the construction.

We observe that the probability that $\mathcal{B}$ does not abort during the simulation is given by $\rho^{q_{prv}}(1-\rho)$ which is maximized at $\rho = 1 - 1/(q_{prv}+1)$. Hence, the probability that $\mathcal{B}$ does not abort is at most $\frac{1}{\tau(q_{prv}+1)}$, where τ denotes the base of the natural logarithm.

Now, the event $\mathsf{AskH}_2^* | \neg\mathsf{Abort}$ denoted by Good, where Abort denotes the event that $\mathcal{B}$ aborts during the simulation. If Good dose not happen, it is clear that the adversary does not gain any advantage greater than 1/2 to guess β. Namely, we have $\mathsf{Pr}[\beta' = \beta | \neg\mathsf{Good}] \leq 1/2$. Hence, by splitting $\mathsf{Pr}[\beta' = \beta]$, we obtain $|\mathsf{Pr}[\beta' = \beta] - \frac{1}{2}| \leq \frac{1}{2}\mathsf{Pr}[\mathsf{Good}]$. To sum up, we have $\epsilon > \frac{2\epsilon'}{q_{H_2}\tau(q_{prv}+1)}$.

Time Complexity. In the simulation, $\mathcal{B}$'s overhead is dominated by the exponentiation computation in response to $\mathcal{A}_{II}$'s H and **Public-Key-Request**

query. So, we have $t > t' + t_{ex}(q_H + q_{pub})$, where t_{ex} denotes the time for computing exponentiation in $\mathbb{Z}_n^*$.

This concludes the proof of Theorem 2.

4 Conclusion

We have presented a new practical CL-PKE scheme that does not depend on the paring. We have proven that our scheme is, in the random oracle model, secure under the assumption that the RSA problem is intractable.

However, the model of our scheme is slightly weaker than the original model [1]. It is still an open problem to design a CL-PKE scheme without paring in the original model [1] that is IND-CCA secure, even relies on the random oracles.

Acknowledgement

This research is supported by the Office of Research, Singapore Management University.

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. Au, M., Chen, J., Liu, J., Mu, Y., Wong, D., Yang, G.: Malicious KGC attacks in certificateless cryptography. In: ASIACCS 2007, pp. 302–311. ACM Press, New York (2007)
3. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005)
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
5. Bentahar, K., Farshim, P., Malone-Lee, J.: Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058 (2005), `http://eprint.iacr.org/2005/058`
6. Cheng, Z., Comley, R.: Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012 (2005), `http://eprint.iacr.org/2005/012`
7. Dent, A.W.: A survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211 (2006), `http://eprint.iacr.org/2006/211`
8. Dent, A., Libert, B., Paterson, K.: Certificateless encryption schemes strongly secure in the standard model. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 344–359. Springer, Heidelberg (2008)
9. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetirc and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

10. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
11. Girault, M.: Self-certified public keys. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 490–497. Springer, Heidelberg (1991)
12. Huang, Q., Wong, D.S.: Generic certificateless encryption in the standard model. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 278–291. Springer, Heidelberg (2007)
13. Hwang, Y.H., Liu, J.K., Chow, S.S.M.: Certificateless Public Key Encryption Secure against KGC Attacks in the Standard Model. Journal of Universal Computer Science, Special Issue on Cryptography in Computer System Security 14(3), 463–480 (2008)
14. Liu, J., Au, M., Susilo, W.: Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In: ASIACCS 2007, pp. 273–283. ACM Press, New York (2007)
15. Lai, J., Kou, W.: Self-Generated-Certificate Public Key Encryption Without Pairing. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 476–489. Springer, Heidelberg (2007)
16. Libert, B., Quisquater, J.: On constructing certificateless cryptosystems from identity based encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006)
17. Okamoto, E., Tanaka, K.: Key Distribution System Based on Identification Infromation. IEEE J. Selected Areas in Communications 7, 481–485 (1989)
18. Park, J.H., Choi, K.Y., Hwang, J.Y., Lee, D.H.: Certificateless Public Key Encryption in the Selective-ID Security Model (Without Random Oracles). In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 60–82. Springer, Heidelberg (2007)
19. Petersen, H., Horster, P.: Self-certified keys - concepts and applications. In: 3rd Int. Conference on Communications and Multimedia Security, pp. 102–116. Chapman and Hall, Boca Raton (1997)
20. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
21. Shi, Y., Li, J.: Provable efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/287 (2005), http://eprint.iacr.org/2005/287
22. Sun, Y., Zhang, F., Baek, J.: Strongly Secure Certificateless Public Key Encryption without Pairing. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 194–208. Springer, Heidelberg (2007)
23. Yum, D.H., Lee, P.J.: Generic construction of certificateless encryption. In: Laganá, A., Gavrilova, M.L., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. LNCS, vol. 3043, pp. 802–811. Springer, Heidelberg (2004)

Strongly Unforgeable ID-Based Signatures without Random Oracles

Chifumi Sato[1], Takeshi Okamoto[2], and Eiji Okamoto[3]

[1] SBI Net Systems Co., Ltd., Meguro Tokyu Bldg., 5th Floor, 2–13–17 Kamiosaki Shinagawa-Ku Tokyo, 141–0021, Japan
c-sato@sbins.co.jp

[2] Department of Computer Science, Faculty of Health Sciences, Tsukuba University of Technology, 4–12–7 Kasuga Tsukuba-shi, Ibaraki, 305–0821, Japan
ken@cs.k.tsukuba-tech.ac.jp

[3] Graduate School of Systems and Information Engineering, University of Tsukuba, 1–1–1 Tennodai Tsukuba-shi, Ibaraki, 305–8573, Japan
okamoto@risk.tsukuba.ac.jp

Abstract. In this paper, we construct a strongly unforgeable ID-based signature scheme without random oracles. The signature size of our scheme is smaller than that of other schemes based on varieties of the Diffie–Hellman problem or the discrete logarithm problem. The security of the scheme relies on the difficulty to solve three problems related to the Diffie–Hellman problem and a one-way isomorphism.

Keywords: Digital signatures, ID-based signatures, Strong unforgeability, Standard models.

1 Introduction

In 1984, Shamir [19] introduced the concept of ID-based cryptosystems, in which the private key of an entity was generated from his identity information (e.g. an e-mail address, a telephone number, etc.) and a master key of a trusted third party called a Private Key Generator (PKG). The advantage of this cryptosystem is that certificates as used in a traditional public key infrastructure can be eliminated. The first ID-based signature (IBS) scheme was proposed by Shamir [19]. Later, many IBS schemes were presented in [7,12,15,18].

For (ID-based) signatures [3,5,6,8,11,16,21,20] or ID-based encryptions [2,20], constructing schemes whose security can be proved without random oracles is one of the most important themes of study, since commonly used hash functions such as MD5 or SHA-1 are not random oracles.

It is known that strongly unforgeable IBS schemes can be constructed with the approach of attaching certificates to strongly unforgeable (non-ID-based) signatures. This approach is mentioned in passing within several papers [9,1,10]. We can construct strongly unforgeable IBS schemes without random oracles by applying the approach to strongly unforgeable signature schemes without random oracles such as the Boneh–Boyen [3], the Zhang–Chen–Susilo–Mu [21], the

Camenisch–Lysyanskaya [6], the Okamoto [14] or the Boneh–Shen–Waters [5]. However, these constructions need at least six signature parameters to include a public key of the signer and two ordinary signatures.

Also, Huang–Wong–Zhao [13] proposed a general method to transform (weakly) unforgeable IBS schemes into strongly unforgeable ones by attaching strong one-time signatures. Therefore, this enables us to construct strongly unforgeable IBS schemes without random oracles by applying it on them to any unforgeable ones such as the Paterson–Schuldt [16]. However, in this transformation, signature sizes of the IBS scheme depend on the public key size and signature size of the underlying strong one-time signature scheme. Almost all the current one-time signature schemes suffer from a drawback that these signature sizes are quite large in practice. Note that a strongly unforgeable signature scheme (in the sense of Definition 2 in [13]) is also a strong one-time signature scheme (Definition 3 in [13]). However, by using a strongly unforgeable signature scheme such as [3,21,6,14,5] instead of the one-time signature, these constructions also need at least six signature parameters.

In this paper, we propose a strongly unforgeable IBS scheme without random oracles, with five signature parameters. The security of the scheme relies on the difficulty to solve three problems related to the Diffie–Hellman (DH) problem, and a one-way isomorphism that no PPT adversary can find the inverse one. The signature size of our scheme is smaller than that of other schemes based on varieties of the DH problem or the discrete logarithm problem.[1] One of the reasons why the number of parameters can be reduced from six to five is that our scheme is directly constructed without applying [9,1,10] or [13].

The paper is organized in the following way. In Section 2, we prepare for the construction of our scheme, along with its proof of security. In Section 3, we will provide two new assumptions related to the DH problem, and make a proposal for our ID-based signature scheme. We prove our scheme satisfying security of strong unforgeability in Section 4, and discuss efficiency in Section 5. We provide conclusions in Section 6.

2 Preliminaries

The aim of this section is to define a one-way isomorphism, a bilinear map, the co-Diffie–Hellman (co-DH) problem, an IBS scheme and the strong unforgeability.

2.1 One-Way Isomorphism and Bilinear Map

The following definitions are due to [17,4]. We assume that

[1] Currently, the most practical strongly unforgeable signature schemes [11,8] without random oracles are constructed based on the Strong RSA assumption. It is known that each component in the parameters of the signature and the public key generated by these schemes needs to be at least 1024-bits in size. On the other hand, it is sufficient to be 160-bits in size for signature schemes constructed based on the discrete logarithm problem (including varieties of the DH problem) over elliptic curves. In this paper, we only consider such schemes.

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p;
- g_2 is a generator of $\mathbb{G}_2$;
- f: $\mathbb{G}_2 \rightarrow \mathbb{G}_1$ is a one-way isomorphism satisfying $f(g_2^x) = g_1^x$, where $x \in \mathbb{Z}_p$ and g_1 is a generator of $\mathbb{G}_1$;
- $e :\ \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is the cryptographic *bilinear map* satisfying the following properties:
  Bilinearity: $e(u^a, v^b) = e(u, v)^{ab}$ for any $u \in \mathbb{G}_1,\ v \in \mathbb{G}_2$ and any $a, b \in \mathbb{Z}$.
  Non-degenerate: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ for $\langle g_1 \rangle = \mathbb{G}_1$ and $\langle g_2 \rangle = \mathbb{G}_2$.
  Computable: There is an efficient algorithm to compute $e(u, v)$ for any $u \in \mathbb{G}_1$ and any $v \in \mathbb{G}_2$.

2.2 The Co-Diffie–Hellman Problem

We provide the co-DH problem in $(\mathbb{G}_2, \mathbb{G}_1)$ as follows. Given

$$(g_1, g_2, g_2^x, g_2^y)$$

as input for random generators $g_1 \in_{\mathrm{R}} \mathbb{G}_1, g_2 \in_{\mathrm{R}} \mathbb{G}_2$ and random numbers $x, y \in_{\mathrm{R}} \mathbb{Z}_p^*$, compute g_1^{xy}. We say that algorithm $\mathcal{A}$ has an advantage ε in solving the co-DH problem in $(\mathbb{G}_2, \mathbb{G}_1)$ if

$$\Pr\left[\mathcal{A}\left(g_1, g_2, g_2^x, g_2^y\right) = g_1^{xy}\right] \geq \varepsilon\ ,$$

where the probability is over the choice $g_1 \in_{\mathrm{R}} \mathbb{G}_1,\ g_2 \in_{\mathrm{R}} \mathbb{G}_2$, $x, y \in_{\mathrm{R}} \mathbb{Z}_p^*$ and the random bits of $\mathcal{A}$.

Assumption 1. *The* (t, ε)*-co-Diffie–Hellman (co-DH) Assumption holds in* $(\mathbb{G}_2, \mathbb{G}_1)$ *if no t-time adversary has an advantage of at least* ε *in solving the co-DH problem in* $(\mathbb{G}_2, \mathbb{G}_1)$.

Notice that, if we set $g_1 := f(g_2) \in \mathbb{G}_1$ for the one-way isomorphism $f :\ \mathbb{G}_2 \rightarrow \mathbb{G}_1$ and the random generator $g_2 \in_{\mathrm{R}} \mathbb{G}_2$, then the generator g_1 is not random.

2.3 ID-Based Signature Schemes

The definition of the IBS scheme in this section is due to [16].

An IBS scheme consists of four phases: *Setup, Extract, Sign* and *Verify* as follows.

Setup: A security parameter is taken as input and returns `params` (system parameters) and `master-key`. The system parameters include a decision of a finite message space $\mathcal{M}$, and a decision of a finite signature space $\mathcal{S}$. Intuitively, the system parameters will be publicly known, while the `master-key` will be known only to the Private Key Generator (PKG).

Extract: The output from Setup (`params`, `master-key`) is taken along with an arbitrary ID $\in \{0, 1\}^*$ as input, and returns a private key d. Here ID is an arbitrary string that will be used as a public key, and d is the corresponding private sign key. The Extract phase extracts a private key from the given public key, and is performed by the PKG.

Sign: A message $M \in \mathcal{M}$, a private key d and `params` are taken as input. It returns a signature $\sigma \in \mathcal{S}$.

Verify: A message $M \in \mathcal{M}$, $\sigma \in \mathcal{S}$, ID and `params` are taken as input. It returns `valid` or `invalid`.

The parameters in Sign and Verify are used in a different order later on. These four phases must satisfy the standard consistency constraint, namely when d is the private key generated by phase *Extract* when it is given ID as the public key, then

$$\forall M \in \mathcal{M}, \forall \sigma := \mathit{Sign}(\texttt{params}, d, M) : \Pr[\mathit{Verify}(\texttt{params}, \mathrm{ID}, M, \sigma) = \texttt{valid}] = 1.$$

2.4 Strong Unforgeability

The definition of the strong unforgeability in this section is due to [3,5,16]. In particular, Paterson–Schuldt [16] defined the unforgeability and the strong unforgeability. However, their construction of the IBS scheme satisfied only the unforgeability.

Strong unforgeability is defined using the following game between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$:

Setup: The challenger $\mathcal{B}$ takes a security parameter k and runs the Setup phase of the IBS scheme. It gives the adversary $\mathcal{A}$ the resulting system parameters `params`. It keeps the `master-key` to itself.

Queries: The adversary $\mathcal{A}$ adaptively makes a number of different queries to the challenger $\mathcal{B}$. Each query can be one of the following.

– **Extract Queries** $(\mathbf{ID}_i)$**:** The challenger $\mathcal{B}$ responds by running phase Extract to generate the private key d_i corresponding to the public key ID_i issued by $\mathcal{A}$. It sends d_i to the adversary $\mathcal{A}$.

– **Signature Queries** $(\mathbf{ID}_i, M_{i,j})$**:** For each query $(\mathrm{ID}_i, M_{i,j})$ issued by $\mathcal{A}$ the challenger $\mathcal{B}$ responds by first running Extract to obtain the private key d_i of ID_i, and then running Sign to generate a signature $\sigma_{i,j}$ of $(\mathrm{ID}_i, M_{i,j})$, and sending $\sigma_{i,j}$ to $\mathcal{A}$.

Output: Finally $\mathcal{A}$ outputs a pair $(\mathrm{ID}_*, M_*, \sigma_*)$. If σ_* is a valid signature of (ID_*, M_*) according to Verify, $\mathrm{ID}_* \notin \{\mathrm{ID}_i\}$ for Extract Queries and $(\mathrm{ID}_*, M_*, \sigma_*) \notin \{(\mathrm{ID}_i, M_{i,j}, \sigma_{i,j})\}$ for Signature Queries, then $\mathcal{A}$ wins.

We define $\texttt{AdvSig}_{\mathcal{A}}$ to be the probability that $\mathcal{A}$ wins the above game, taken over the coin tosses made by $\mathcal{B}$ and $\mathcal{A}$.

Definition 1. An adversary $\mathcal{A}$ $(q_{\mathrm{e}}, q_{\mathrm{s}}, t, \varepsilon)$-breaks an ID-based signature (IBS) scheme if $\mathcal{A}$ runs in a time of at most t, $\mathcal{A}$ makes at most q_{e} Extract Queries, at most q_{s} Signature Queries, and $\texttt{AdvSig}_{\mathcal{A}}$ is at least ε. An IBS scheme is $(q_{\mathrm{e}}, q_{\mathrm{s}}, t, \varepsilon)$-strongly existential unforgeable under an adaptive chosen message attack, strongly unforgeable, if no adversary $(q_{\mathrm{e}}, q_{\mathrm{s}}, t, \varepsilon)$-breaks it.

3 Our Scheme

In this section, we provide two new assumptions and propose an IBS scheme.

3.1 Underlying Proposed Problems

We provide Assumptions 2 and 3 related to the DH problem.

The first problem is defined as follows. Given

$$\left(g_1, g_2, g_1^x, g_2^{r_i}, g_2^{x+1/r_i} \middle| \; i = 1, \ldots, q\right)$$

as input for random generators $g_1 \in_{\mathrm{R}} \mathbb{G}_1$, $g_2 \in_{\mathrm{R}} \mathbb{G}_2$ and random numbers x, $r_1, \ldots, r_q \in_{\mathrm{R}} \mathbb{Z}_p^*$, compute $\left(g_1^{r_*}, g_2^{x+1/r_*}\right)$ for some $r_* \in \mathbb{Z}_p^*$ and $r_* \notin \{r_1, \ldots, r_q\}$. Note that the index $x + 1/r_i$ means $x + (1/r_i)$. We say that algorithm $\mathcal{A}$ has an advantage ε in solving the first problem if

$$\Pr\left[\mathcal{A}\left(g_1, g_2, g_1^x, g_2^{r_i}, g_2^{x+1/r_i} \middle| \; i = 1, \ldots, q\right) = \left(g_1^{r_*}, g_2^{x+1/r_*}\right)\right] \geq \varepsilon \ ,$$

where the probability is over the choice $g_1 \in_{\mathrm{R}} \mathbb{G}_1$, $g_2 \in_{\mathrm{R}} \mathbb{G}_2$, $x, r_1, \ldots, r_q \in_{\mathrm{R}} \mathbb{Z}_p^*$, some $r_* \in \mathbb{Z}_p^*$, $r_* \notin \{r_1, \ldots, r_q\}$ and the random bits of $\mathcal{A}$.

Assumption 2. *A (q, t, ε)-Assumption II holds if no t-time adversary has an advantage of at least ε in solving the first problem.*

The second problem is defined as follows. Given

$$\left(g_1, g_2, g_1^x, g_2^{1/x}, g_2^{r_i}, g_2^{xr_i}, g_2^{x+1/r_i} \middle| \; i = 1, \ldots, q\right)$$

as input for random generators $g_1 \in_{\mathrm{R}} \mathbb{G}_1$, $g_2 \in_{\mathrm{R}} \mathbb{G}_2$ and random numbers $x, r_1, \ldots, r_q \in_{\mathrm{R}} \mathbb{Z}_p^*$, compute $\left(g_2^{r_*}, g_2^{xr_*}, g_2^{x+1/r_*}\right)$ for some $r_* \in \mathbb{Z}_p^*$ and $r_* \notin \{r_1, \ldots, r_q\}$. We say that algorithm $\mathcal{A}$ has an advantage ε in solving the second problem if

$$\Pr\left[\mathcal{A}\left(g_1, g_2, g_1^x, g_2^{1/x}, g_2^{r_i}, g_2^{xr_i}, g_2^{x+1/r_i} \middle| \; i = 1, \ldots, q\right) = \left(g_2^{r_*}, g_2^{xr_*}, g_2^{x+1/r_*}\right)\right] \geq \varepsilon \ ,$$

where the probability is over the choice $g_1 \in_{\mathrm{R}} \mathbb{G}_1$, $g_2 \in_{\mathrm{R}} \mathbb{G}_2$, $x, r_1, \ldots, r_q \in_{\mathrm{R}} \mathbb{Z}_p^*$, some $r_* \in \mathbb{Z}_p^*$, $r_* \notin \{r_1, \ldots, r_q\}$ and the random bits of $\mathcal{A}$.

Assumption 3. *A (q, t, ε)-Assumption III holds if no t-time adversary has an advantage of at least ε in solving the second problem.*

If we set $g_1 := f(g_2) \in \mathbb{G}_1$ for the one-way isomorphism $f : \mathbb{G}_2 \to \mathbb{G}_1$ and the random generator $g_2 \in_{\mathrm{R}} \mathbb{G}_2$, then the generator g_1 is not random in the two assumptions. The existence of f was proved by Saito–Hoshino–Uchiyama–Kobayashi [17], on multiplicative cyclic groups constructed on non-supersingular elliptic curves. Security of our scheme is essentially based on the co-DH assumption, our proposed two assumptions, and the isomorphism f. In particular, our proposed two assumptions which are defined in a rigorous manner contribute to prove the security of strong unforgeability for our scheme.

3.2 Scheme

We shall give an IBS scheme. This scheme consists of four phases: *Setup, Extract, Sign* and *Verify*. For the moment we shall assume that the identity ID are elements in $\{0,1\}^{n_1}$, but the domain can be extended to all of $\{0,1\}^*$ using a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{n_1}$. Similarly, we shall assume that the signature message M to be signed are elements in $\{0,1\}^{n_2}$.

Setup: The PKG chooses multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of sufficiently large prime order p, a random generator g_2 of $\mathbb{G}_2$, the one-way isomorphism $f : \mathbb{G}_2 \to \mathbb{G}_1$ with $g_1 := f(g_2)$, and the cryptographic bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. He generates $MK := g_2^{\alpha} \in \mathbb{G}_2$ from a random number $\alpha \in_{\mathrm{R}} \mathbb{Z}_p^*$, and calculates $A_1 := f(MK)\ (= g_1^{\alpha}) \in \mathbb{G}_1$.

$$\begin{tikzcd}
\mathbb{Z}_p^* \arrow[r] & \mathbb{G}_2 \arrow[r, "f"] & \mathbb{G}_1 \\
\alpha \arrow[r, mapsto] & MK := g_2^{\alpha} \arrow[r, mapsto] & A_1 := f(MK)\ (= g_1^{\alpha})
\end{tikzcd}$$

Also he generates $u' := g_2^{x'} \in \mathbb{G}_2, U = (u_1, \ldots, u_{n_1}) := (g_2^{x_1}, \ldots, g_2^{x_{n_1}}) \in \mathbb{G}_2^{n_1}, v' := g_2^{y'} \in \mathbb{G}_2$, and $V = (v_1, \ldots, v_{n_2}) := (g_2^{y_1}, \ldots, g_2^{y_{n_2}}) \in \mathbb{G}_2^{n_2}$ for random numbers $x', x_1, \ldots, x_{n_1}, y', y_1, \ldots, y_{n_2} \in_{\mathrm{R}} \mathbb{Z}_p^*$. The master secret `master-key` is MK and the public parameter are

$$\texttt{params} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, f, g_1, g_2, A_1, u', U, v', V) \ .$$

Extract: Let ID be an n_1-bit identity and id_k $(k = 1, \ldots, n_1)$ denote the kth bit of ID. To generate a private key d_{ID} for $\mathrm{ID} \in \{0,1\}^{n_1}$, the PKG picks a random number $s \in_{\mathrm{R}} \mathbb{Z}_p^*$, and computes

$$d_{\mathrm{ID}} = (d_1, d_2) := \left(g_2^s, g_2^{\alpha} \cdot \left(u' \prod_{k=1}^{n_1} u_k^{\mathrm{id}_k} \right)^{1/s} \right) \in \mathbb{G}_2^2 \ .$$

Sign: Let M be an n_2-bit signature message to be signed and m_k $(k = 1, \ldots, n_2)$ denote the kth bit of M. A signature $\sigma := (\sigma_1, \ldots, \sigma_5)$ of (ID, M) is generated as follows.

$$\begin{aligned}
(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) &:= \left(f(d_1), g_2^r, d_1^r, d_2, d_1 \cdot \left(v' \prod_{k=1}^{n_2} v_k^{m_k} \right)^{1/r} \right) \\
&= \left(g_1^s, g_2^r, g_2^{sr},\ g_2^{\alpha} \cdot \left(u' \prod_{k=1}^{n_1} u_k^{\mathrm{id}_k} \right)^{1/s}, g_2^s \cdot \left(v' \prod_{k=1}^{n_2} v_k^{m_k} \right)^{1/r} \right)
\end{aligned}$$

for a random number $r \in \mathbb{Z}_p^*$.

Verify: Given `params`, (ID, M) and $\sigma = (\sigma_1, \ldots \sigma_5)$, verify

$$e(\sigma_1, \sigma_2) = e(g_1, \sigma_3)\ ,$$
$$e\left(A_1^{-1} \cdot f(\sigma_4), \sigma_3\right) = e\left(f(\sigma_2), u' \prod_{k=1}^{n_1} u_k^{\mathrm{id}_k}\right) \text{ and}$$
$$e\left(\sigma_1^{-1} \cdot f(\sigma_5), \sigma_2\right) = e\left(g_1, v' \prod_{k=1}^{n_2} v_k^{m_k}\right)\ .$$

If the equalities hold the result is `valid`; otherwise the result is `invalid`.

If an entity with identity ID constructs a signature $\sigma = (\sigma_1, \ldots, \sigma_5)$ on a message M as described in the Sign phase above, it is easy to see that σ will be accepted by a verifier. Thus the scheme is correct.

4 Security Proof

Theorem 1. *Suppose that the (t_0, ε_0)-co-DH Assumption in $(\mathbb{G}_2, \mathbb{G}_1)$, $(q_1, t_1, \varepsilon_1)$-Assumption II and $(q_2, t_2, \varepsilon_2)$-Assumption III hold with $g_1 := f(g_2)$. Then the proposed ID-based signature scheme is $(q_\mathrm{e}, q_\mathrm{s}, t, \varepsilon)$-strongly unforgeable, provided that $q_\mathrm{e} \le q_1$, $q_\mathrm{s} \le q_2$, $t \le \min(t_0, t_1, t_2) - O((q_\mathrm{e} + q_\mathrm{s})T)$ and $\varepsilon\left(1 - 2(q_\mathrm{e} + q_\mathrm{s})/p\right) \ge \varepsilon_0 + \varepsilon_1 + \varepsilon_2$, where T is the maximum time for an exponentiation in $\mathbb{G}_2$.*

An outline of our proof is as follows. Suppose that there exists an adversary, $\mathcal{A}$, who breaks our IBS scheme in Section 3, and a challenger, $\mathcal{B}$, takes the Assumption II challenge. After $\mathcal{A}$ and $\mathcal{B}$ execute the strongly unforgeable game, $\mathcal{A}$ outputs a valid tuple for an identity, a message and a signature. Then $\mathcal{B}$ will compute the Assumption II response which is `valid`. The tuple from $\mathcal{A}$ must not contradict the co-DH assumption and the Assumption III.

Proof. Suppose that there exists an adversary, $\mathcal{A}$, who $(q_\mathrm{e}, q_\mathrm{s}, t, \varepsilon)$-breaks our IBS scheme. We construct a simulator, $\mathcal{B}$, to play the Assumption II game. The simulator $\mathcal{B}$ will take the Assumption II challenge

$$\left(g_1, g_2, g_1^{\alpha}, g_2^{s_i}, g_2^{\alpha + 1/s_i}\,\middle|\, i = 1, \ldots, q_1\right)$$

for $\alpha, s_1, \ldots, s_{q_1} \in_\mathrm{R} \mathbb{Z}_p^*$, and run $\mathcal{A}$ executing the following steps.

4.1 Simulator Description

Setup: The simulator $\mathcal{B}$ generates $u' := g_2^{x'} \in \mathbb{G}_2, U = (u_1, \ldots, u_{n_1}) := (g_2^{x_1}, \ldots, g_2^{x_{n_1}}) \in \mathbb{G}_2^{n_1}, v' := g_2^{y'} \in \mathbb{G}_2$, and $V = (v_1, \ldots, v_{n_2}) := (g_2^{y_1}, \ldots, g_2^{y_{n_2}}) \in \mathbb{G}_2^{n_2}$ for random numbers $x', x_1, \ldots, x_{n_1}, y', y_1, \ldots, y_{n_2} \in_\mathrm{R} \mathbb{Z}_p^*$, and sends

$$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, f, g_1, g_2, g_1^{\alpha}, u', U, v', V)$$

to $\mathcal{A}$.

Queries: The adversary $\mathcal{A}$ adaptively makes a number of different queries to the challenger $\mathcal{B}$.

Assume that $\mathcal{U}_{\mathrm{e}}$ is the subscript set of identities in Extract Queries, $\mathcal{U}_{\mathrm{s}}$ is that of identities in Signature Queries, $\mathcal{U} := \mathcal{U}_{\mathrm{e}} \cup \mathcal{U}_{\mathrm{s}}$, and $\mathcal{M}_{\mathrm{s}}^{i}$ is that of messages in Signature Queries for the ID_i $(i \in \mathcal{U}_{\mathrm{s}})$.

Each query can be one of the following.

– **Extract Queries:** The adversary $\mathcal{A}$ adaptively issues Extract Queries ID_i $(i \in \mathcal{U}_{\mathrm{e}})$. Assume that

$$X_i := x' + \sum_{k=1}^{n_1} \mathrm{id}_{i,k}\ x_k\ , \tag{1}$$

where $\mathrm{ID}_i := (\mathrm{id}_{i,1}, \ldots, \mathrm{id}_{i,n_1}) \in \{0,1\}^{n_1}$.
(4.1-E1) If $X_i \equiv 0 \pmod{p}$, $\mathcal{B}$ aborts this game.
(4.1-E2) Otherwise (i.e. $X_i \not\equiv 0 \pmod{p}$), $\mathcal{B}$ does not abort the game, and generates $d_i = (d_{i,1}, d_{i,2})$ of ID_i:

$$(d_{i,1}, d_{i,2}) := \left((g_2^{s_i})^{X_i}\ , g_2^{\alpha + 1/s_i} \right) \tag{2}$$

$$= \left(g_2^{\overline{s_i}}, g_2^{\alpha} \cdot \left(u' \prod_{k=1}^{n_1} u_k^{\mathrm{id}_{i,k}} \right)^{1/\overline{s_i}} \right) \tag{3}$$

and sends it to $\mathcal{A}$. Here $\overline{s_i} := s_i X_i \bmod p$ $(i \in \mathcal{U}_{\mathrm{e}})$. (Notice that, by eliminating all $s_i \in_{\mathrm{R}} \mathbb{Z}_p^*$ in (2), we can regard all $\overline{s_i} \in_{\mathrm{R}} \mathbb{Z}_p^*$ as random numbers in (3).)

– **Signature Queries:** The adversary $\mathcal{A}$ adaptively issues Signature Queries $(\mathrm{ID}_i, M_{i,j})$ $\left(i \in \mathcal{U}_{\mathrm{s}}, j \in \mathcal{M}_{\mathrm{s}}^{i}\right)$. Assume that X_i is from (1) for $i \in \mathcal{U}_{\mathrm{s}}$ and

$$Y_{i,j} := y' + \sum_{k=1}^{n_2} m_{i,j,k}\ y_k\ , \tag{4}$$

where $M_{i,j} := (m_{i,j,1}, \ldots, m_{i,j,n_2}) \in \{0,1\}^{n_2}$.
(4.1-S1) If $X_i \equiv 0 \pmod{p}$ or $Y_{i,j} \equiv 0 \pmod{p}$, $\mathcal{B}$ aborts this game.
(4.1-S2) Otherwise (i.e. $X_i \not\equiv 0 \pmod{p}$ and $Y_{i,j} \not\equiv 0 \pmod{p}$), $\mathcal{B}$ does not abort the game, and generates $\sigma_{i,j} = (\sigma_{i,j,1}, \ldots, \sigma_{i,j,5})$ of $(\mathrm{ID}_i, M_{i,j})$:

$$\begin{cases} \sigma_{i,j,1} := (g_1^{s_i})^{X_i} = g_1^{\overline{s_i}} \\ \sigma_{i,j,2} := (g_2)^{r_{i,j} Y_{i,j} / X_i} = g_2^{\overline{r_{i,j}}} \\ \sigma_{i,j,3} := (g_2^{s_i})^{r_{i,j} Y_{i,j}} = g_2^{\overline{s_i}\ \overline{r_{i,j}}} \\ \sigma_{i,j,4} := g_2^{\alpha + 1/s_i} = g_2^{\alpha} \cdot \left(g_2^{X_i} \right)^{1/\overline{s_i}} = g_2^{\alpha} \cdot \left(u' \prod_{k=1}^{n_1} u_k^{\mathrm{id}_{i,k}} \right)^{1/\overline{s_i}} \\ \sigma_{i,j,5} := \left(g_2^{s_i + 1/r_{i,j}} \right)^{X_i} = g_2^{\overline{s_i}} \cdot \left(g_2^{Y_{i,j}} \right)^{1/\overline{r_{i,j}}} = g_2^{\overline{s_i}} \cdot \left(v' \prod_{k=1}^{n_2} v_k^{m_{i,j,k}} \right)^{1/\overline{r_{i,j}}} \end{cases}$$

and sends it to $\mathcal{A}$. Here $\overline{s_i} := s_i X_i \bmod p$ $(i \in \mathcal{U}_{\mathrm{s}})$ and $\overline{r_{i,j}} := r_{i,j} Y_{i,j} / X_i \bmod p$ $\left(i \in \mathcal{U}_{\mathrm{s}}, j \in \mathcal{M}_{\mathrm{s}}^i\right)$. (Notice that, by eliminating all $s_i, r_{i,j} \in_{\mathrm{R}} \mathbb{Z}_p^*$, we can regard all $\overline{s_i}, \overline{r_{i,j}} \in_{\mathrm{R}} \mathbb{Z}_p^*$ as random numbers.)

Output: The adversary $\mathcal{A}$ outputs $(\mathrm{ID}_*, M_*, \sigma_*)$ such that $\sigma_* = (\sigma_{*,1}, \ldots, \sigma_{*,5}) \in \mathbb{G}_2^5$ is a valid signature of (ID_*, M_*), $\mathrm{ID}_* \notin \{\mathrm{ID}_i \mid i \in \mathcal{U}_{\mathrm{e}}\}$ and $(\mathrm{ID}_*, M_*, \sigma_*) \notin \left\{(\mathrm{ID}_i, M_{i,j}, \sigma_{i,j}) \mid i \in \mathcal{U}_{\mathrm{s}}, j \in \mathcal{M}_{\mathrm{s}}^i\right\}$.

Artificial Abort: Assume that

$$X_* := x' + \sum_{k=1}^{n_1} \mathrm{id}_{*,k}\ x_k \ \text{ and } \ Y_* := y' + \sum_{k=1}^{n_2} m_{*,k}\ y_k\ , \tag{5}$$

where $\mathrm{ID}_* := (\mathrm{id}_{*,1}, \ldots, \mathrm{id}_{*,n_1}) \in \{0,1\}^{n_1}$ and $M_* := (m_{*,1}, \ldots, m_{*,n_2}) \in \{0,1\}^{n_2}$. If $\mathrm{ID}_* \neq \mathrm{ID}_i$ and $X_* \equiv X_i \pmod p$ for some $i \in \mathcal{U}$, or if $M_* \neq M_{i,j}$ and $Y_* \equiv Y_{i,j} \pmod p$ for some $i \in \mathcal{U}_{\mathrm{s}}$ and $j \in \mathcal{M}_{\mathrm{s}}^i$, then $\mathcal{B}$ aborts this game.

4.2 Analysis

The adversary $\mathcal{A}$ cannot distinguish the above game from Simulator Description with the abort when $X_i \equiv 0 \pmod p$ and $Y_{i,j} \not\equiv 0 \pmod p$ or $X_i \not\equiv 0 \pmod p$ and $Y_{i,j} \equiv 0 \pmod p$, and the strongly unforgeable game without this abort, since

$$\Pr\left[\bigcup_{i \in \mathcal{U}} X_i \equiv 0 \pmod p \cup \bigcup_{\substack{i \in \mathcal{U}_{\mathrm{s}},\\ j \in \mathcal{M}_{\mathrm{s}}^i}} Y_{i,j} \equiv 0 \pmod p\right] \leq \frac{q_{\mathrm{e}} + q_{\mathrm{s}}}{p}$$

and this probability is negligible when $q_{\mathrm{e}} + q_{\mathrm{s}} \ll p$. Thus we shall consider only the game from Simulator Description.

Since σ_* is `valid`, we assume that

$$\begin{cases} \sigma_{*,1} := g_1^{\overline{s_*}} \\ \sigma_{*,2} := g_2^{\overline{r_*}} \\ \sigma_{*,3} := g_2^{\overline{s_*}\ \overline{r_*}} \\ \sigma_{*,4} := g_2^{\alpha} \cdot \left(u' \prod_{k=1}^{n_1} u_k^{\mathrm{id}_{*,k}}\right)^{1/\overline{s_*}} = g_2^{\alpha + X_*/\overline{s_*}} \\ \sigma_{*,5} := g_2^{\overline{s_*}} \cdot \left(v' \prod_{k=1}^{n_2} v_k^{m_{*,k}}\right)^{1/\overline{r_*}} = g_2^{\overline{s_*} + Y_*/\overline{r_*}} \end{cases}$$

where $\overline{s_*}, \overline{r_*} \in \mathbb{Z}_p^*$.

(4.2-1) If $X_* \equiv 0 \pmod p$, $\sigma_{*,4} = g_2^{\alpha}$. Then $\mathcal{B}$ generates $\left(g_1^{s_*}, g_2^{\alpha + 1/s_*}\right)$ for some $s_* \in \mathbb{Z}_p^*$ and $s_* \notin \{s_1, \ldots, s_q\}$, which is a valid output of the Assumption II challenge.

(4.2-2) Otherwise (i.e. $X_* \not\equiv 0 \pmod p$).

(4.2-2.1) Suppose that $\mathrm{ID}_* \notin \{\mathrm{ID}_i \mid i \in \mathcal{U}_\mathrm{e}\}$ (which is an assumption of the strong unforgeability) and $(\mathrm{ID}_*, \overline{s_*}) \notin \{(\mathrm{ID}_i, \overline{s_i}) \mid i \in \mathcal{U}_\mathrm{s}\}$. Then, it is sufficient to consider

$$\begin{aligned}\Pr\Big[\mathcal{A}\Big(&g_1, g_2, g_1^{\alpha}, g_2^{x'}, g_2^{x_1}, \ldots, g_2^{x_{n_1}}, g_2^{y'}, g_2^{y_1}, \ldots, g_2^{y_{n_2}},\\ &g_2^{X_i}, g_2^{Y_{i,j}}, g_2^{\overline{s_i}}, g_2^{\overline{r_{i,j}}}, g_2^{\overline{s_i}\,\overline{r_{i,j}}}, g_2^{\alpha+X_i/\overline{s_i}}, g_2^{\overline{s_i}+Y_{i,j}/\overline{r_{i,j}}} \Big| \; i \in \mathcal{U}_\mathrm{s}, j \in \mathcal{M}_\mathrm{s}^i\Big)\\ &= \Big(g_2^{X_*}, g_2^{Y_*}, g_1^{\overline{s_*}}, g_2^{\overline{r_*}}, g_2^{\overline{s_*}\,\overline{r_*}}, g_2^{\alpha+X_*/\overline{s_*}}, g_2^{\overline{s_*}+Y_*/\overline{r_*}}\Big)\Big] \,, \qquad (6)\end{aligned}$$

in the case that $\mathcal{A}$ knows all $g_2^{\overline{s_i}}$ $(= d_{i,1})$. This means that $\mathcal{U} = \mathcal{U}_\mathrm{e} = \mathcal{U}_\mathrm{s}$. Suppose that the probability (6) $\geq \varepsilon'$ for some $\varepsilon' \leq \varepsilon$.

Then the probability (6) can be reduced to a contradiction of either the co-DH assumption or Assumption II .

(4.2-2.2) Otherwise (i.e. $\mathrm{ID}_* \notin \{\mathrm{ID}_i \mid i \in \mathcal{U}_\mathrm{e}\}$ and $(\mathrm{ID}_*, \overline{s_*}) = (\mathrm{ID}_l, \overline{s_l})$ for some $l \in \mathcal{U}_\mathrm{s}$), then $X_* = X_l$. It is sufficient to consider

$$\begin{aligned}\Pr\Big[\mathcal{A}\Big(&g_1, g_2, g_2^{y'}, g_2^{y_1}, \ldots, g_2^{y_{n_2}}, g_2^{Y_{l,j}}, g_{\mathbf{1}}^{\overline{s_l}}, g_2^{1/\overline{s_l}}, g_2^{\overline{r_{l,j}}}, g_2^{\overline{s_l}\,\overline{r_{l,j}}}, g_2^{\overline{s_l}+Y_{l,j}/\overline{r_{l,j}}} \Big| \; j \in \mathcal{M}_\mathrm{s}^l\Big)\\ &= \Big(g_2^{Y_*}, g_2^{\overline{r_*}}, g_2^{\overline{s_l}\,\overline{r_*}}, g_2^{\overline{s_l}+Y_*/\overline{r_*}}\Big)\Big] \qquad (7)\end{aligned}$$

in the case that $\mathcal{A}$ knows $x', x_1, \ldots, x_{n_1}$ and g_2^{α}. Suppose that the probability (7) $\geq \varepsilon''$ for $\varepsilon' + \varepsilon'' = \varepsilon$.

Then the probability (7) can be reduced to a contradiction of either the co-DH assumption or Assumption III .

We omit the proof of the limited range of values $(q_\mathrm{e}, q_\mathrm{s}, t, \varepsilon)$.

Therefore, we proved Theorem 1. $\square$

5 Efficiency

In this section, we consider efficiency of strongly unforgeable IBS schemes without random oracles.

Huang–Wong–Zhao [13] proposed a general method to transform unforgeable IBS schemes into strongly unforgeable ones by attaching strong one-time signatures. Table 1 shows efficiency of IBS schemes from the Huang–Wong–Zhao [13]. For $(x_\mathrm{r}, y_\mathrm{r})$ of the row in the table, x_r represents the number of signature parameters and y_r that of the bilinear maps. For $(x_\mathrm{c}; y_\mathrm{c})$ of the column, x_c the

Table 1. Efficiency of IBS schemes from the transformation in Huang–Wong–Zhao [13]

Strong one-time signatures Unforgeable IBS schemes	[3] (4; 1)	[21] (4; 2)	[6] (5; 4)	[14] (4; 2)	[5] (4; 2)
Paterson–Schuldt [16] (3, 3)	7/4	7/5	8/7	7/5	7/5

Table 2. Efficiency of IBS schemes from the transformation in [1,9,10]

Signature schemes $\mathcal{S}'$ / Certificates (signature) schemes $\mathcal{S}$			[3] (4; 1)	[21] (4; 2)	[6] (5; 4)	[14] (4; 2)	[5] (4; 2)
Boneh–Boyen	[3]	(2, 1)	6/2	6/3	7/5	6/3	6/3
Zhang–Chen–Susilo–Mu	[21]	(2, 2)	6/3	6/4	7/6	6/4	6/4
Camenisch–Lysyanskaya ver.A	[6]	(3, 4)	7/5	7/6	8/8	7/6	7/6
Okamoto	[14]	(3, 2)	7/3	7/4	8/6	7/4	7/4
Boneh–Shen–Waters	[5]	(3, 2)	7/3	7/4	8/6	7/4	7/4

number of signature parameters and public keys, and y_c that of the bilinear maps. For x_t/y_t in the table, x_t $(= x_c + x_r)$ the number of signature parameters for each strongly unforgeable IBS scheme; and y_t $(= y_c + y_r)$ that of the bilinear maps. Notice that we count the number of the signature parameters to be small. However, these constructions need at least six signature parameters.

Also, it is known that strongly unforgeable IBS schemes can be constructed with the approach of attaching certificates to strongly unforgeable (non-ID-based) signatures. Table 2 shows efficiency of IBS schemes from this construction in [1,9,10].

All these constructions need at least six signature parameters. In our scheme of Section 3.2, it is sufficient to be five. On the other hand, our scheme is inefficient since the bilinear map is used six times during one iteration of verification in the scheme.

6 Conclusions

In this paper, we proposed a strongly unforgeable IBS scheme without random oracles, with five signature parameters, based on three problems related to the DH problem and a one-way isomorphism. However, our scheme is inefficient since the bilinear map (the pairing) is used six times during one iteration of verification in the scheme. Our next step is to propose more efficient schemes with the same security (or we have a possibility that the six times have not been a problem by a future study of the computation process rate of the bilinear map).

References

1. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 268–286. Springer, Heidelberg (2004)
2. 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)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

4. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational Diffie–Hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)
6. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
7. Cha, J.C., Cheon, J.H.: An identity-based signature from gap diffie-hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2002)
8. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. In: ACM Conference on Computer and Communications Security, pp. 46–51 (1999)
9. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong key-insulated signature schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Heidelberg (2002)
10. Galindo, D., Herranz, J., Kiltz, E.: On the generic construction of identity-based signatures with additional properties. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 178–193. Springer, Heidelberg (2006)
11. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signatures without the random oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
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. Huang, Q., Wong, D.S., Zhao, Y.: Generic transformation to strongly unforgeable signatures. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 1–17. Springer, Heidelberg (2007)
14. 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)
15. Paterson, K.G.: ID-based signatures from pairings on elliptic curves. Cryptology ePrint Archive, Report 2002/004 (2002), http://eprint.iacr.org/
16. Paterson, K.G., Schuldt, J.C.N.: Efficient identity-based signatures secure in the standard model. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006)
17. Saito, T., Hoshino, F., Uchiyama, S., Kobayashi, T.: Candidate one-way functions on non-supersingular elliptic curves. IEICE Transactions 89-A(1), 144–150 (2006)
18. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS 2000, Okinawa, Japan (2000)
19. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
20. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
21. Zhang, F., Chen, X., Susilo, W., Mu, Y.: A new signature scheme without random oracles from bilinear pairings. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 67–80. Springer, Heidelberg (2006)

On the Security of a Certificate-Based Signature Scheme and Its Improvement with Pairings

Jianhong Zhang

College of Science, North China University of Technology,
Beijing 100041, P.R.China
jhzhangs@gmail.com

Abstract. In traditional public key signature, the public key of a signer is essentially a random string selected from a given set. It is infeasible to prove that a party is indeed the signer for a given signature. In general, the public key of a user needs a management authority to authenticate it. It results in that traditional public key cryptosystem (PKC) requires high maintenance cost for certificate management. Although, identity based cryptosystem (IBC) reduces the overhead of management, it suffers from the drawback of key escrow. Certificate-based cryptosystem combines the advantage of both PKC and IBC as it avoids the usage of certificates and does not suffer from key escrow. Recently, Liu *et.al* proposed an efficient Certificate-based signature and showed that the scheme was secure in the random oracles. Unfortunately, this paper shows that the scheme is insecure and discusses the flaws in their security proof. Then the corresponding attacks are given. To overcome the flaws, an improved scheme is proposed and the result shows that the scheme is provable secure against two game attacks of certificate-based signature in the random oracle model. The security is closely related to the computational Diffie-Hellman problem.

Keywords: Security analysis, attack, improved scheme, the CDH problem, certificate-based signature.

1 Introduction

In traditional public key signature (PKS)[11], a user Alice signs a message using her private key. A verifier Bob verifies the signature using Alice's public key. However, generally speaking, Alice's public key is a random string selected from a given set. Therefore, it is infeasible to prove that a party is indeed the signer for a given signature. This problem can be solved by incorporating a certificate generated a trusted party called the Certification Authority (CA) which provides an unforgeable and trusted link between a public key and the identity of a signer. This hierarchical framework is referred to as the Public Key Infrastructure (PKI). In general, the signer registers its own public key with its identity in certificate server and anyone wishing to obtain the signer's public key needs to request it by sending the server the identity of the signer and gets it. When verifying a signature using the signer's public key, a verifier must obtain the signer's certification status of public key, hence in general the verifier makes a query on the signer's certificate status to the CA. In the point of view of a verifier, it takes two verification steps to verify a signature on a message. This approach seems to be inefficient, in particular, when the number of users is very large. And it also increases burden of the CA's certificate management.

To simplify certificate management of traditional PKI, Shamir [13] introduced Identity-based cryptology (IBC). It can solve the aforementioned problem by using Alice's identity or IP address as her public key while the corresponding private key is a result of some mathematical operation that takes as input the user's identity and the master secret key of a trusted authority, referred to as **Private Key Generator** (PKG). The main practical benefit of IBC lies in greatly reduction of need for public key certification. The PKG can generate the private key of all its users, so *Private Key Escrow* becomes an inherent problem in IBC. It results in unconditional trust to PKG for a user. Furthermore, the produced private key must be sent over secure channel, which makes secret key distribution a daunting task[2]. Certificate is implicitly provided in IBC and the explicit authentication of public key is no longer required.

To fill the gap between traditional cryptology and identity-based cryptology, Gentry[2] introduced the notion of certificate-based encryption (CBE), which combines public-key encryption (PKE) and IBE while preserving their merits. In CBE, each user generates his own private key and public key and request a certificate from the CA while the CA uses the key generation algorithm of an identity based encryption (IBE) scheme to the certificate. In this way, the certificate is implicitly used as the private key of the users as the signing key. Here CA is equivalent to PKG in the IBC. Although the CA knows the certificate of a user, it also cannot produce a signature in the name of user, since it does not have the user's private key. In addition to CBE, the notion of certificate based signature (CBS) was first suggested by Kang *et.al*[9]. Unfortunately, one of their schemes was found insecure against key replacement attack[11] which was pointed our by Li *et.al*[8]. And Li *et.al* also gave a novel certificate-based signature scheme. In ISPEC 2007, Au *et.al* [1] proposed a certificate-based ring signature scheme. In 2007, Wang *et.al*[14] proposed a certificate-based proxy cryptosystem with revocable proxy decryption power. Recently, J.K.Liu *et.al*[10] proposed an efficient certificate-based signature scheme as the scheme did not require any pairing operations, which is regards costly operations compared to other operation. And they claimed that the scheme was provably secure in the random oracle model and can be against two attack games of certificate-based signature.

Our contribution: In this paper, we first analyze the security of the first one of Liu *et.al*'s schemes which were proven secure against two attack games of certificate-based signature in [10]. Then we show that the scheme is insecure and it cannot be against either attack game of certificate-based signature. Furthermore, we give the corresponding two attacks on the schemes and pinpoint the reason accounting for the insecurity . Finally, to overcome the above attacks, we give an improved schemes and prove that the scheme is secure in the random oracle model.

The rest of this paper is organized as follows. We describe security and adversarial model of CBS in section 2. After we review Liu *et.al*'s signature scheme in section 3, we give the corresponding attack in section 4. In section 5, an improved scheme is given and the security of the improved scheme is analyzed. Finally, we conclude the paper in section 6.

2 Security Model

The certificate-based signature is a compromise between ID-based signature and PKI-based signature. It inherits merits of ID-based signature and PKI-based signature.

In the following, we reviews the definition of a certificate-based signature scheme. It consists of the six algorithms :

– *Setup* is a probabilistic algorithm taking a security parameter k as input. It outputs the certifier's master key msk and public parameters $param$. The algorithm is run by private key generator (PKG).
– *UserKeyGen* is probabilistic algorithm which takes as input $param$ and is run by the user. It outputs a public key PK and a secret key usk.
– *Certify* is probabilistic algorithm that takes as input master secret msk, public parameters $param$, the user's public key PK and identity ID and a time period t. It returns $Cert_t'$ to user. It is run by PKG. The aim is to issue a certificate in the phase.
– *Consolidate* is deterministic certificate consolidation algorithm taking as input $(param, t, Cert_t')$ and optionally $Cert_{t-1}$. It outputs certificate $Cert_t$ which is used by a user in time period t.
– *Sign* is a probabilistic algorithm which takes as input $(param, t, Cert_t, usk, m)$ and outputs a signature δ.
– *Verify* is deterministic algorithm which takes as input $(param, t, PK, ID, \delta)$. If δ is valid, it outputs 1. Otherwise, it outputs $\perp$.

Note that in a concrete certificate-based signature scheme, it may not involve certificate consolidate phase. In such situation, *Consolidate* algorithm will output $Cert_t = Cert_t'$. For simplicity, we will omit *Conslidate* and the time t in the rest of the paper.

2.1 Unforgeability

Unforgeability is a primitive property of a signature scheme. Intuitively, an adversary should not be able to forge a valid signature in a secure certificate-based signature (CBS) scheme which can be against adaptively chosen message attack [7]. Certificate-based signature is a combination of public-key signature and ID-based signature, where the signer needs both its personal secret key and a certificate from PKG. Thus, the security of a certificate-based signature requires that the signer can generate a valid signature under the public key PK and identity ID if and only if he has $Cert$ and usk. Only with one of those, the signer cannot generate a valid signature.

According to an adversary is in possession of the secret key of which entity, we divide the adversary into two types. One type is an uncertified entity which can randomly choose a pair key (PK, usk) which is not certified by PKG or replace a user's public key PK to produce a forgery. This attack scenarios is looked upon a malicious user. The other type is an adversary which is be in possession of the master key msk to attack a fixed entity's public key. Such attack scenarios is looked upon a malicious PKG. In the following, we use the enhanced model by Li *et.al* [8] which captures key replacement attack in the security of the first attack type.

CBS Game 1 Existential Unforgeability. The challenger runs $Setup$, gives $param$ to the adversary $\mathcal{A}$ and keeps msk secret. The adversary $\mathcal{A}$ issues the following queries:

- On user-key-gen query (ID), if ID has already been created, then PK_{ID} is returned. Otherwise, the challenger runs the algorithm **User-Key-Gen** to produce a secret/public key pair (usk_{ID}, PK_{ID}) and added (ID, usk_{ID}, PK_{ID}) to the list L. Then PK_{ID} is returned.
- On corruption query (ID). The challenger checks the list L and returns the secret key usk_{ID}.
- On certification query (ID). The challenger run $Certify$ algorithm and returns the certificate $Cert_{ID}$ to $\mathcal{A}$. Note that $Cert_{ID}$ is the certificate of the pair (ID, PK_{ID}) where PK_{ID} is the public key returned from **User-Key-Gen** query.
- On signing query (ID, PK_{ID}, m). The challenger runs the **Sign** algorithm and returns the signature δ.

Here, we can replace any user's public key with his own choice, but once it has replaced the public key, it cannot obtain the certificate of the false public key from the challenger.

- **Output.** Finally, $\mathcal{A}$ outputs a forge $(m^*, \delta^*, ID^*, PK^*)$. We say $\mathcal{A}$ wins the game if

  - δ^* is a valid signature on the message m^* under the public key PK_{ID^*} with identity ID^*. Here PK_{ID^*} is chosen by $\mathcal{A}$ and might not be returned from the **User-Key-Gen** oracle.
  - ID^* has never been submitted as one of Certify queries.
  - (ID^*, m^*) has never been submitted as one of Sign queries.

We define $\mathcal{A}$'s advantage in this game to be $Adv(\mathcal{A}) = Pr[\mathcal{A} \;\; wins]$.

In the second attack, the adversary $\mathcal{A}$ is in possession of the master key of PKG and wants to generate a valid signature under the public key PK^* without the knowledge of the corresponding secret key. The security of this type attack is denied by the following game between $\mathcal{A}$ and the challenger $\mathcal{C}$.

CBS Game 2 Existential Unforgeability. The challenger runs *Setup*, gives *param* and *msk* to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ can adaptively submit queries to User-Key-Gen oracle, corruption oracle and signing oracle as Game 1. But different from Game 1,the adversary is not allowed to replace any user's public key. At last, $\mathcal{A}$ outputs a signature δ^* on message m^* under public key PK_{ID^*} and identity ID^*. The adversary win the game if

- δ^* is a valid signature on message m^* under public key PK^* and the PKG's public key msk where PK_{ID^*} is the public key output from the User-Key-Gen query ID^*.
- ID^* has never been submitted to corruption oracle.
- (m^*, PK^*, ID^*) has never been submitted as one of Sign queries.

The adversary's advantage in the game is defined as $Adv(\mathcal{A})$=P[$\mathcal{A}$ *wins*]

3 Reviews of Liu *et.al*'s Certificate-Based Signature Scheme without Pairings

[**Setup of System Parameters:**] Let $\mathbb{G}$ be a multiplicative group with order q. The PKG randomly chooses a generator $g \in \mathbb{G}$ and $x \in Z_q$ to compute $X = g^x$ as public key. Let $H : \{0,1\}^* \to Z_q^*$ be a cryptographic hash function. The system parameters param and master secret key msk are as follows:

$$\mathbf{param} = (\mathbb{G}, q, g, X, H), msk = x$$

[**UserKeyGen:**] A user picks $u \in Z_q$ as his secret key usk at random, and computes the corresponding public key PK as (g^u, X^u, τ_u) where τ_u is the following non-interactive proof-of-knowledge (PoK)

$$PK\{(u) : U_1 = g^u \wedge U_2 = X^u\}$$

[**Certify:**] Let $\tilde{h} = H(PK, ID)$ for user with public key PK and binary string ID which is used to identity the user. To produce a certificate for the user the CA randomly selects $r \in_R Z_q^*$, compute

$$R = g^r \qquad s = r^{-1}(\tilde{h} - xR) \mod q$$

The certificate is (R, s). Note that a correctly generated certificate should fulfill the following equality:

$$R^s X^R = g^{\tilde{h}} \tag{1}$$

[**Sign:**] To produce a signature on message $m \in \{0,1\}^*$, the signer with public key PK, the certificate (R, s) and the secret key u computes as follows:

1. randomly choose $y \in_R Z_q^*$.

2. compute $Y = R^{-y}$, $h = H(Y, R, m)$ and $z = y + hsu \mod q$
3. the resultant signature is $\delta = (Y, R, z)$.

[Verify:] Given a signature $\delta = (Y, R, z)$ on message m under the public key PK, a verifier first checks whether τ_u is a valid PoK. If not, output $\perp$. Otherwise, compute $h = H(Y, R, m)$ and $\tilde{h} = H(PK, ID)$, then checks whether

$$(g^u)^{h\tilde{h}} \overset{?}{=} R^z Y (X^u)^{hR} \tag{2}$$

Output valid if it holds, otherwise, output $\perp$.

4 Attack on Liu *et.al*'s Certificate-Based Signature Scheme without Pairings

In [10],Liu *et.al* claimed that their scheme is secure against the certifier's attack in which an adversary in possession of the master key msk attacks a fixed entity's public key and a uncertified entity' attack in which an adversary can replace an user's public key while the master key is unknown for him. Then they give the corresponding security proof of the scheme in the random oracle models, respectively. Unfortunately, we show that the scheme is insecure by analyzing security of the scheme.

4.1 Attack 1

Given a signature $\delta = (Y, R, z, \tau_u)$ under the public key PK and the identity ID, a certifier with the master key x can attack as follows:

1. First, compute $\tilde{h}' = H(PK, ID)$.
2. then, compute $R' = x^{-1}\tilde{h}'$, where x is master secret key.
3. randomly choose $r \in Z_q^*$ to set $z' = r$ and compute $Y = (R'^{z'})^{-1}$.
4. the forged signature on message m' is $\delta' = (\tau_u = \tau_u', R', z', Y')$.

For an user, after certificated, the user's public key PK is fixed. Thus, the corresponding non-interactive proof-of-knowledge τ_u of public key is fixed. Obviously, the above forging $\delta' = (\tau_u = \tau_u', R', z', Y')$ can be accepted. Since

$$\begin{aligned} R'^{z'} Y' (X^u)^{hR'} &= R'^{z'} (R'^{z'})^{-1} (X^u)^{hR'} \\ &= (X^u)^{hR'} \\ &= (X^u)^{hx^{-1}\tilde{h}'} \\ &= (g^u)^{h\tilde{h}'} \\ \tilde{h}' &= H(PK, ID) \end{aligned}$$

According to the above verification procedure, we know the above equations hold; Hence, it indicates that our attack is successful. And in the whole forgery process, we find that, the signed message is not used in R', z' and Y'. It means that the forgery attack is valid for any message-signature .

4.2 Attack on Security Proof of the Scheme

In the following, we show that security proof of Liu *et.al*'s signature scheme in the **unforgeability against Game 2 Adversary** has the flaws. In the signing quering phase of security proof, when $\mathcal{A}$ queries the signing oracle for a message m and a public key (g^u, X^u, τ_u) with identity ID, the authors thought that the challenger $\mathcal{C}$ responded as follows:

1. first check whether proof-of-knowledge for (g^u, X^u) is valid.
2. then check whether PK, ID, has been queried for the $H()$ random oracle or extraction oracle before. If yes, it retrieves $(R, s, H(PK, ID), PK, ID)$. Let $\tilde{h} = H(PK, ID)$ and randomly chooses $h, z \in_R Z_q$ to set $Y = \frac{(g^u)^{h\tilde{h}}}{R^z(X^u)^{hR}}$
3. Finally, let $h = H(Y, R, m)$ as the response to the random oracle $H(Y, R, m)$.

But, in fact, the challenger $\mathcal{C}$ can respond not like the above steps, but as follow:

1. First, compute $\tilde{h} = H(PK, ID)$.
2. then, compute $R = x^{-1}\tilde{h}'$, where x is master secret key, it is known to the challenger $\mathcal{C}$.
3. randomly choose $r \in Z_q^*$ to set $z = r$ and compute $Y = (R^z)^{-1}$.
4. set $h = H(R, Y, m)$
5. the simulated signature on message m is $\delta = (\tau_u, R, z, Y)$.

After such signing query response is used, the challenger $\mathcal{C}$ cannot apply rewind technique to the point it just queries $H(Y, R, m)$. Hash function $H()$ cannot play a role of random oracle. Thus, the discrete logarithm problem in the security proof can be solved. It means that the security proof exits the flaws.

4.3 Attack3

In the following, we show that the scheme is not against an uncertified entity attack. That is to say, an uncertified entity can produce a forgeable signature without a certificate which is issued by CA. The attack is as follows:

1. randomly choose $r \in Z_q$ to compute $R = g^r$
2. randomly choose $a \in Z_q$ to compute $Y = X^{-aR}$.
3. randomly select a signed message m to compute $h = H(R, Y, m)$.
4. compute $u = \frac{a}{h}$ as the secret key of an user, and set $PK = (g^u, X^u)$ as the public key of the user.
5. compute $\tilde{h} = H(PK, ID)$ to set $z = \frac{a\tilde{h}}{r}$, where ID is the identity of the user.
6. produce a non-interactive proof-of-knowledge π_u by the secret key u.

$$\pi_u = PK\{(u) : U_1 = g^u \wedge U_2 = X^u\}$$

The forged signature on message m under the public key $PK = (g^u, X^u)$ and identity ID is $\delta = (R, Y, z)$.

To verify the validity of the forged signature, we only check whether the forged signature $\delta = (R, Y, z)$ satisfies verification equation.

$$\begin{aligned} R^z Y (X^u)^{hR} &= R^z X^{-aR} (X^u)^{hR} \\ &= R^z X^{-aR} (X)^{\frac{a}{h} hR} \\ &= (g^r)^{\frac{a\tilde{h}}{r}} \\ &= g^{a\tilde{h}} = (g^{\frac{a}{h}})^{h\tilde{h}} \\ &= (g^u)^{h\tilde{h}} \end{aligned}$$

According to the above verification, we find that the forged signature δ on message m satisfies verification equation. It denotes that our attack is successful. And it show that the scheme is not against an uncertified entity attack.

5 Its Improved Scheme

In the section, to overcome the attacks above, we give an improved certificate-based signature scheme. To clarify our scheme, we first review preliminaries.

5.1 Bilinear Maps

In the following, we recall the notations related to bilinear groups [4,5]. Let $\mathbb{G}_1, \mathbb{G}_2$ be bilinear groups as follows:

1. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic multiplicative groups of prime order p, where possible $\mathbb{G}_1 = \mathbb{G}_2$
2. g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$.
3. e is a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$
   (a) Bilinear: for all $u, v \in \mathbb{G}_1$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$;
   (b) Non-degenerate: $e(g, g) \neq 1$;
4. e and the group action in $\mathbb{G}_1, \mathbb{G}_2$ can been computed efficiently.

5.2 Security Assumption

Here we first review the definition of the strong Diffie-Hellman (SDH) assumption introduced in [4], on which the security of our signature is based, and then extend it into a new security assumption, the extended strong Diffie-Hellman assumption, on which the security of a variant of our signature scheme is based.

Strong Diffie-Hellman Assumption: Let $\mathbb{G}_1, \mathbb{G}_2$ be bilinear groups as shown the above section. The $q-$SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: given $g_1 \in \mathbb{G}_1$, and the $(q+1)-$tuple $(g_2, g_2^x, \cdots, g_2^{x^q}) \in \mathbb{G}_2^{q+1}$ as input, output a pair $(g_1^{\frac{1}{x+c}}, c)$ where $c \in Z_p$. Algorithm $\mathcal{A}$ has advantage, $Adv_{SDH}(q)$, in solving $q - SDH$ in $(\mathbb{G}_1, \mathbb{G}_2)$ if

$$Adv_{SDH}(q) \to Pr[\mathcal{A}(g_1, g_2, g_2^x, \cdots, g_2^{x^q})] = (g_1^{\frac{1}{x+c}}, c)$$

Where the probability is taken over the random choice of $g_2 \in \mathbb{G}_2$, $x \in Z_p^*$, and the coin tosses of $\mathcal{A}$.

Definition 1. *Adversary $\mathcal{A}$ $(t,\epsilon)-$breaks the $q-$SDH problem if $\mathcal{A}$ runs in time at most t and $Adv_{SDH}(q)$ is at leat ϵ. The $(q,t,\epsilon)-$SDH assumption holds if no adversary $\mathcal{A}$ $(t,\epsilon)-$breaks the $q-$SDH problem.*

Definition 2 (Computational Diffie-Hellman (CDH) Assumption). . *Let $\mathcal{G}$ be a CDH parameter generator. We say an algorithm $\mathcal{A}$ has advantage $\epsilon(k)$ in solving the CDH problem for $\mathbb{G}_1$ if for a sufficiently large k,*

$$Adv_{\mathcal{G},\mathcal{A}}(t) = Pr[\mathcal{A}(p,\mathbb{G}_1,g^x,g^y) = g^{xy} \mid (p,\mathbb{G}_1) \leftarrow \mathcal{G}^k, P \leftarrow \mathbb{G}_1, x,y \leftarrow \mathcal{Z}_p]$$

We say that $\mathbb{G}_1$ satisfies the CDH assumption if for any randomized polynomial time in t algorithm $\mathcal{A}$ we have the $Adv_{\mathbb{G}_1,\mathcal{A}}(\mathrm{t})$ is negligible function.
$q-$**SDH+CDH Assumption:** The $q-$SDH+CDH problem in $\mathbb{G}_1$ is defined as follows: give $q+1$-tuple $(g_1, g_1^{\alpha}, \cdots, g_1^{\alpha^q})$ and a random pair (g_1, g_1^r) of group $\mathbb{G}_1$ as inputs, output $(\rho \leftarrow (g_1)^{\frac{r}{(\alpha+c)}}, c)$, where g_1 is a generator of group $\mathbb{G}_1$, $c, r \in_R Z_p^*$. Note that α and r are unknown numbers. Algorithm $\mathcal{A}$ has advantage, Adv$_{SDH}(q)$, in solving q-SDH+CDH in $\mathbb{G}_1$ if

$$Adv_{q-SDH+CDH}(q) \leftarrow Pr[\mathcal{A}(g_1, g_1^{\alpha}, \cdots, g_1^{\alpha^q}, g_1^r) = (g_1^{\frac{r}{(\alpha+c)}}, c)]$$

Where the probability is taken over the random choices of $g_1 \in \mathbb{G}_1$, $\alpha, r \in Z_p^*$, and the coin tosses of $\mathcal{A}$.

Definition 3. *Adversary $\mathcal{A}$ $(t,\epsilon)-$breaks the $q-$SDH+CDH problem if $\mathcal{A}$ runs in time at most t and $Adv_{q-SDH+CDH}(q)$ is at least ϵ. The (q,t,ϵ)-SDH+CDH assumption holds if no adversary $\mathcal{A}$ (t,ϵ)-breaks the q-SDH+CDH problem.*

Definition 4. *(Discrete Logarithm Assumption).Given a group $\mathcal{G}$ of prime order q with generator g and elements $A \in \mathcal{G}$, the discrete logarithm problem in $\mathcal{G}$ is to output $x \in Z_q$ such that $A = g^x$.*

An adversary $\mathcal{B}$ has at least an ϵ advantage if

$$Pr[\mathcal{B}(g,A) = x | A = g^x] \geq \epsilon$$

We say that the (ϵ,t)-DL assumption holds in a group $\mathcal{G}$ if no algorithm running in time at most t can solve that DL problem in $\mathcal{G}$ with advantage at least ϵ.

5.3 Our Improved Scheme

In the following, our improved scheme is proposed, the detail steps are shown as follows:

[**Setup of System Parameters:**] Let $\mathbb{G}$ be a multiplicative group with order q, $g \in \mathbb{G}$ is a generator of group $\mathbb{G}$. The PKG randomly chooses $\alpha \in Z_q$ as master secret key and compute $X = g^{\alpha}$ as public key. Note that PKG is responsible for issuing certificate. $h_1, h_2 \in \mathbb{G}_1$ were randomly chosen and $H(\cdot), H_1(\cdot), H_0(\cdot)$

are three one-way hash functions which satisfy $H : \mathbb{G}_1^2 \times \{0,1\}^* \to Z_q, H_1 : \mathbb{G}_1 \times \{0,1\}^* \to Z_q$ and $H_0 : \mathbb{G}_1^2 \times \{0,1\}^* \to \mathbb{G}_1$.The system parameters param and master secret key msk are as follows:

$$\mathbf{param} = (g, h_1, h_2, e, H(\cdot), H_0(\cdot), H_1(\cdot), X), msk = x$$

[**UserKeyGen:**] A user picks $u \in Z_q$ as his secret key usk at random, and computes the corresponding public key PK as (g^u, τ_u) where τ_u is the following non-interactive proof-of-knowledge (PoK)

$$PK\{(u) : U_1 = g^u\}$$

[**Certify:**]To certify for user with public key PK and binary string ID which is used to identity the user. The PKG computes $h_0 = H_0(X, ID, U_1)$ and

$$d = (h_0)^{\frac{1}{\alpha - H_1(U_1, ID)}}$$

The certificate is d. Note that a correctly generated certificate should fulfill the following equality:

$$e(d, X \cdot g^{-H_1(U_1, ID)}) = e(h_0, g)$$

[**Sign:**] To produce a signature on message $M \in \{0,1\}^*$, the signer with public key PK, the certificate d and the secret key u computes as follows:

1. randomly choose $s \in_R Z_q^*$.
2. compute $\delta = (\delta_1, \delta_2)$, where

$$\delta_1 = d^u \cdot (h_2 h_1^{H_1(ID, U_1)})^{sm}, \delta_2 = (X g^{H_1(ID, U_1)})^s$$

   and $m = H(\delta_2, M, U_1)$.
3. the resultant signature on message M is $\delta = (\delta_1, \delta_2)$.

According to the above signing phase, we know that the size of our signature is more shorter than that of Liu *et.al*'s signature.

[**Verify:**] Upon receiving the signature $\delta = (\delta_1, \delta_2)$ on message M, a verifier computes as follows:

1. compute $m = H(\delta_2, M, U_1)$ and $h_0 = H_0(X, ID, U_1)$;
2. verify

$$e(X g^{-H_1(ID, U_1)}, \delta_1) = e(h_0, U_1) e(\delta_2, (h_2 h_1^{H_1(ID, U_1)})^m)$$

Correctness. It is easy to see that the improved signature scheme is correct. It is shown as follows: If a signature $\delta = (\delta_1, \delta_2)$ is valid, then the signature δ must pass the verification equation. Because

$$\begin{aligned}
&e(X g^{-H_1(ID, U_1)}, \delta_1) \\
&= e(X g^{-H_1(ID, U_1)}, d^u \cdot (h_2 h_1^{H_1(ID, U_1)})^{sm}) \\
&= e(X g^{-H_1(ID, U_1)}, (h_0^u)^{\frac{1}{\alpha - H_1(ID, U_1)}} \cdot (h_2 h_1^{H_1(ID, U_1)})^{sm}) \\
&= e(U_1, h_0) e(X g^{-H_1(ID, U_1)}, (h_2 h_1^{H_1(ID, U_1)})^{sm}) \\
&= e(U_1, h_0) e(\delta_2, (h_2 h_1^{H_1(ID, U_1)})^m)
\end{aligned}$$

where $m = H(\delta_2, M, U_1)$ and $h_0 = H_0(, U_1, ID)$.

6 Security Analysis

To demonstrate that our improved scheme is secure, we must prove that the scheme can be against game 1 adversary and game 2 adversary which are discussed.

Theorem 1 (**Unforgeability against Game 1 Adversary**)**.** If there exists a game1 adversary $\mathcal{A}$ which breaks our improved scheme with advantage at most ϵ and runs in times at most t, then the $q-$SDH+CDH problem can be (ϵ', t')-solved.

Theorem 2 (**Unforgeability against Game 2 Adversary**)**.** If there exists a game 2 adversary $\mathcal{A}$ which breaks our improved scheme with advantage at most ϵ and runs in times at most t, then the Computational Diffie-Hellman problem in group $\mathbb{G}$ can be (ϵ', t')-solved.

7 Conclusion

Certificate-based cryptosystem is an important public key cryptology system. Recently, Liu *et.al* proposed an efficient Certificate-based signature scheme and showed that the scheme was secure against two game attack of certificate-based signature scheme in the random oracles. Unfortunately, Our analysis show that the first one of the certificate-based signature schemes proposed by Liu *et.al* is insecure, namely, a malicious PKG can forge a signature on arbitrary message in name of any user's identity and a uncertified user is also able to forge a message-signature. And we discuss the flaw in their proof and the corresponding attack ways. To overcome the flaws, an improved scheme is proposed and the scheme is proven to secure against two game attacks of certificate-based signature. The security is closely related to the computational diffie-hellman problem.

Acknowledgement

The authors are grateful to the anonymous referees of ISPEC2009 for their helpful comments. This work is supported by Natural Science Foundation of China (NO:60703044, 90718001), The Beijing Nova Programma (NO:2007B001), the PHR, Program for New Century Excellent Talents in University(NCET-06-188)and The Beijing Natural Science Foundation Programm and Scientific Research Key Program of Beijing Municipal Commission of Education (NO:KZ2008 10009005).

References

1. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Certificate based (Linkable) ring signature. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 79–92. Springer, Heidelberg (2007)

2. Gentry, C.: Certificate-based Encryption and the Certificate Revocation Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
3. Geiselmann, W., Steinwandt, R.: A Key Substitution Attack on SFLASH, Cryptology ePrint Archive: Report 2004/245 (2004), http://eprint.iacr.org/2004/245
4. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. Journal of Cryptology 17(4), 297–319
5. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
7. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal of computing 17(2), 281–308 (1988)
8. Li, J., Huang, X., Mu, Y., Susilo, W., Wu, Q.: Certificate-based signature: Security model and efficient construction. In: López, J., Samarati, P., Ferrer, J.L. (eds.) EuroPKI 2007. LNCS, vol. 4582, pp. 110–125. Springer, Heidelberg (2007)
9. Kang, B.G., Park, J.H., Hahn, S.G.: A certificate-based signature scheme. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 99–111. Springer, Heidelberg (2004)
10. Liu, J.K., Baek, J., Susilo, W., Zhou, J.: Cettificate-based Signature Scheme without Pairings or Random oracles. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 285–297. Springer, Heidelberg (2008)
11. Nyberg, K., Rueppel, R.A.: Message recovery for signature schemes based on the discrete logarithm problem. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 182–193. Springer, Heidelberg (1995)
12. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
13. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
14. Wang, L.H., Shao, J., Cao, Z.-F., Mambo, M., Yamamura, A.: A certificate-based proxy cryptosystem with revocable proxy decryption power. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 297–311. Springer, Heidelberg (2007)
15. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
16. Zheng, Y.: Identification, Signature and Signcryption using High Order Residues Modulo an RSA Composite. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 48–63. Springer, Heidelberg (2001)
17. 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)

An Empirical Investigation into the Security of Phone Features in SIP-Based VoIP Systems

Ruishan Zhang,[1] Xinyuan Wang,[1] Xiaohui Yang,[1] Ryan Farley,[1] and Xuxian Jiang[2]

[1] George Mason University, Fairfax, VA 22030, USA
zhangruishan@gmail.edu, {xwangc,xyang3,rfarley3}@gmu.edu
[2] N.C. State University, Raleigh, NC 27606, USA
jiang@cs.ncsu.edu

Abstract. Phone features, e.g., *911 call*, *voicemail*, and *Do Not Disturb*, are critical and necessary for all deployed VoIP systems. In this paper, we empirically investigate the security of these phone features. We have implemented a number of attacks and experimented with VoIP services by leading VoIP service providers Vonage, AT&T and Gizmo. Our experimental results demonstrate that a man-in-the-middle or remote attacker could transparently 1) hijack selected E911 calls and impersonate the Public Safety Answering Point (PSAP); and 2) spoof the voicemail servers of both the caller and the callee of selected VoIP calls; and 3) make spam calls to VoIP subscribers even if *Do Not Disturb* is enabled. These empirical results confirm that leading deployed SIP-based VoIP systems have serious security vulnerabilities.

Keywords: VoIP security, SIP, voicemail fraud, 911 hijacking, voice spam.

1 Introduction

In addition to the basic function of making and receiving a call, VoIP systems generally offer many phone features, e.g., *911 call*, *voicemail*, and *Do Not Disturb*. Phone features are critical and necessary for all deployed Public Switched Telephone Network (PSTN) and VoIP systems.

Among all phone features, 911 emergency call is perhaps the most critical one. Recognizing that proper function of 911 call could impact the "life or death for millions of customers that subscribe to VoIP service" [1], the Federal Communications Commission (FCC) requires that all the interconnected VoIP services must support Enhanced 911 (E911) call and automatically report the caller ID and the registered location of E911 calls. On the other hand, voicemail is one of the most frequently used features of VoIP service. It is estimated [2] that 60∼70% of phone calls will be answered by voicemail rather than human. Given that a voice message often contains personal and sensitive information, any compromise of voicemail would violate the privacy of both the sender and the receiver of the voice message. *Do Not Disturb* allows VoIP users to temporarily block all incoming phone calls and have some quiet time. When *Do Not Disturb* fails to work, spam phone calls might continually annoy VoIP users.

Signaling channel and voice channel are two most important components of VoIP systems. In current deployed systems, the Session Initiation Protocol (SIP) [3] and the Real Time Transport Protocol (RTP) [4] are the most dominant signaling and voice transport protocol, and being widely used.

Although the VoIP features are intuitively expected as trustworthy and reliable as those in the traditional PSTN, the open architecture of VoIP has enabled many attacks on voice communication that were not possible in the traditional PSTN. In this paper, we empirically investigate and evaluate the security of phone features in currently deployed SIP-based VoIP systems in the U.S., e.g., Vonage [5], AT&T's CallVantage [6] and Gizmo [7]. Specifically, we focus on the trustworthiness of *E911 call*, *voicemail*, and *Do Not Disturb*.

Assuming there exists a man-in-the-middle (MITM) in between the VoIP phones and the VoIP servers or a remote attacker on the Internet, we implement a number of MITM and remote attacks on the investigated voice services. Specifically, the MITM can transparently hijack selected 911 emergence calls and impersonate the PSAP. The MITM could also launch various voicemail fraud attacks against selected VoIP subscribers. When a VoIP phone makes a call to a PSTN phone or receives a call from a PSTN phone, the MITM can impersonate the callee's voicemail server and ask the caller to leave a voice message or call another phone number. In addition, the MITM can impersonate the voicemail server when the caller accesses voicemail. This would allow the attacker to generate arbitrary fake voice messages to the caller. Finally, a remote attacker,not necessarily a MITM, can circumvent *Do Not Disturb* and make arbitrary calls to Vonage and AT&T VoIP phones. Our experiments confirm that currently deployed VoIP systems are far from secure and trustworthy.

The rest of this paper is organized as follows. Section 2 briefly overviews SIP and SIP security mechanisms. Section 3 introduces our exploitation methodology. Section 4 describes our experiments on E911, voicemail and *Do Not Disturb*. Section 5 discusses the root causes of the vulnerabilities in deployed VoIP systems, and proposes some approaches to mitigate potential threats. Section 6 introduces related work. Finally, section 7 concludes the paper.

2 SIP Overview

Session Initiation Protocol (SIP) [3], is a HTTP-like, application layer signaling protocol used for creating, modifying, and terminating multimedia sessions (e.g. VoIP calls) among Internet endpoints. SIP signaling involves various components: user agents (UA), proxy servers, redirect servers, registrar servers,location servers. An UA represents an endpoint of the communication (i.e., a SIP phone). The proxy server is the intermediate server that acts on behalf of UA to forward the SIP messages to its destination.

The SIP specification [3] recommends using TLS or IPSec to protect SIP signaling messages. It also suggests using S/MIME to protect the integrity and

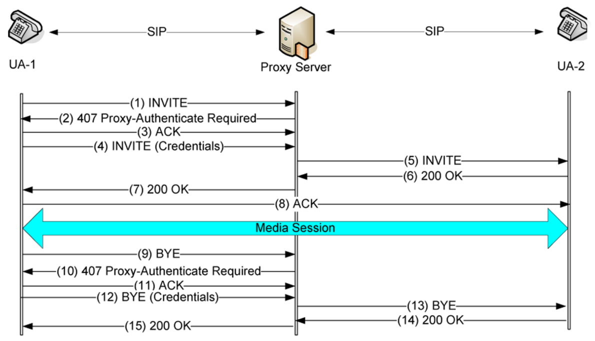


Fig. 1. An Example of Message Flow of SIP Authentication for INVITE and BYE Messages

confidentiality of SIP message bodies. However, most deployed SIP VoIP systems only utilize SIP authentication to protect SIP messages. Fig. 1 shows the typical SIP authentication of call setup and termination.

The SIP authentication of currently deployed VoIP systems has the following weaknesses [3][8]:

- It only protects a few important SIP messages.
- It only protects a few SIP fields.
- It only authenticates SIP messages from SIP phones to the SIP servers.

3 Investigation Approach

Our approach of the security investigation is from the perspective of VoIP customers rather than VoIP service providers. Consequently, we leave aside the attacks on the VoIP service provider's servers and instead focus on those attacks that directly target the VoIP users. We choose to experiment with the residential VoIP services by Vonage, AT&T, who are the No.1 and No.2 [9] in U.S. VoIP market share. In addition, we experiment with Gizmo's popular softphone that has been used by millions of people. Note all the attacks we experiment are against our own accounts and phones only.

The key technique used in our empirical investigation is the MITM who can monitor, modify and forge VoIP traffic to or from selected VoIP users. Since most VoIP phones are many hops away from the VoIP servers, attackers have many opportunities to play MITM attack on existing deployed VoIP services. In fact, Zhang et al [10] have shown that a remote attacker from anywhere on the Internet, who is not originally in between the targeted VoIP phone and its VoIP servers, can become a MITM within a few minutes by exploiting some implementation flaws in the VoIP phone.

The MITM is used to verify the weaknesses in E911 and voicemail. To circumvent *Do Not Disturb*, the MITM is not required. A remote attacker can successfully make spam calls to defeat this feature.

4 Experiments with Deployed VoIP Services

To empirically investigate the security of deployed VoIP systems, we build a testbed that consists of the MITM, Vonage, AT&T SIP phones and Gizmo's softphone. Fig. 2 illustrates the network setup of our testbed. All the SIP phones are within a private network 192.168.1.x, where the Gizmo softphone runs on a Windows XP virtual machine. The MITM runs on a FreeBSD 5.4 virtual machine, which acts as the Network Address Translation (NAT) router for the private network. Specifically, the natd runs on the external interface `lnc0` of the FreeBSD virtual machine, and our MITM intercepts, manipulates the network traffic at the internal interface `lnc1` via divert socket. Note the MITM does not need to be directly connected to the VoIP phones, and it could be at anywhere along the path of VoIP traffic.

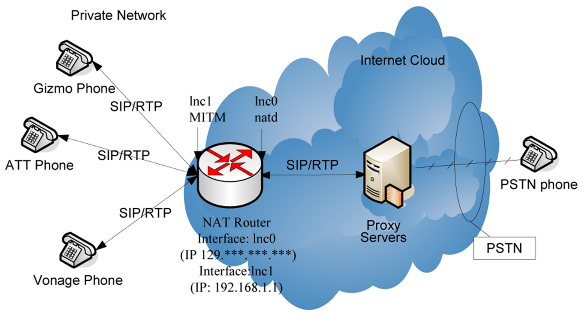


Fig. 2. Testbed Setup of MITM Attacks

4.1 911 Call Hijacking

When a VoIP user dials 911, the call is supposed to be routed to a geographically appropriate PSAP by the VoIP service provider. However, our experiments show that the MITM could hijack the selected 911 call from either Vonage or AT&T SIP phone, divert it to any third party and let the third party impersonate the PSAP. In this case, the 911 call is never routed to the VoIP service provider, and yet it appears to the 911 caller that the 911 call is successfully connected to the appropriate PSAP.

Fig. 3 illustrates the message flows of the 911 call hijacking experiments. Specifically, the left and the right parts show the message flows of 911 calls from the Vonage phone and the AT&T phone respectively. Depending on the service provider's implementation, the signaling path and the RTP stream path could be different. All SIP and RTP packets are transferred on UDP. We use SIP/RTP server(s) to denote the SIP server and the RTP server which handle the signaling messages and the RTP streams respectively.

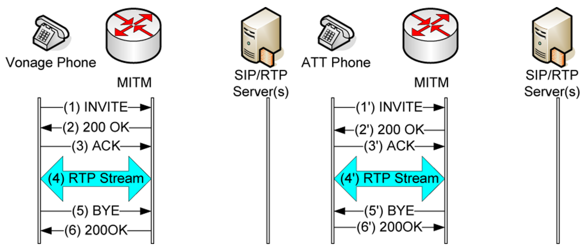


Fig. 3. Message Flow of Hijacking 911 Calls

When we dial 911 from the Vonage (or the AT&T) SIP phone[1], the caller's SIP phone sends an `INVITE sip:911` message to the SIP server in step (1) or (1'). Our MITM intercepts the `INVITE` message, pretends to be the SIP server and responds with a spoofed `200 OK` message in step (2) or (2'). In the spoofed `200 OK` message, the MITM sets its own IP address and port number (e.g., 12345) as the RTP stream termination point, which asks the caller's SIP phone to establish the voice stream to the MITM instead of the service provider's server. Although Vonage and AT&T's SIP server normally challenges the `INVITE` message with `407 proxy-authentication required` as shown in Fig. 1, we find that both Vonage and AT&T SIP phones actually accept the spoofed `200 OK` message directly, and they respond with an `ACK` message to the SIP server in step (3) or (3'). The MITM intercepts the `ACK` message so that it won't reach the service provider's SIP server. At this point, the three way handshake for the VoIP call setup is finished, and the 911 call between the caller and the MITM has been established. Then at step (4) or (4'), the caller talks to the MITM, who pretends to be the PSAP.

Traditionally, 911 calls can only be terminated by the PSAP. We find that the Vonage SIP phone actually allows the 911 caller to terminate the 911 call. Specifically, the Vonage SIP phone sends out a `BYE` message to the Vonage's SIP server once the 911 caller hangs up in step (5). Then the MITM simply responds with a fake `200 OK` message. On the other hand, the AT&T SIP phone prevents the 911 caller from terminating the call until it is reset. Specifically, if the 911 caller hangs up the AT&T SIP phone, it starts and keeps ringing until the handset is picked up or the phone adapter is reset. This behavior conforms to the specification of traditional 911 call in the PSTN. At step (5'), the MITM pretends the PSAP and sends the AT&T SIP phone a fake `BYE` message. The AT&T SIP phone responds with a `200 OK` message which would terminate the 911 call completely.

Despite the 911 call implementation differences between Vonage and AT&T SIP phones, our MITM is able to transparently hijack selected 911 calls and pretend to be the PSAP. Our experiments demonstrate that 911 calls from existing deployed Vonage and AT&T VoIP services are not trustworthy.

[1] Note we block our experimental 911 calls at our border router so that they will not interfere with any real 911 calls.

4.2 Fake Voicemail Attacks

Voicemail is intended for the caller to leave a voice message when the callee is not available. Once the caller hears the voicemail prompt after dialing the callee's phone number, he would expect it is the callee's voicemail. In addition, when someone wants to check his voicemail and dials his voicemail access number, he would expect to reach his own voicemail. In this section, we show that the MITM can compromise the trust of voicemail by spoofing both the caller's voicemail and callee's voicemail.

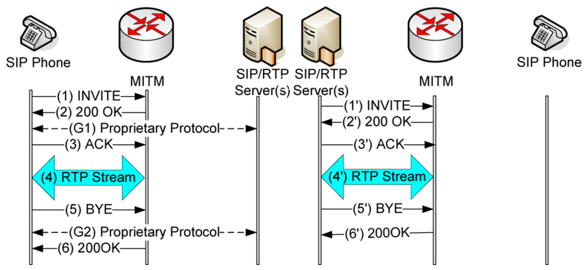


Fig. 4. Message Flows of Fake Callee's Voicemail at Caller's and Callee's Sides

Fake Callee's Voicemail Attack. When the caller places a call, the MITM intercepts the call and spoofs the callee's voicemail to prompt the caller to leave a voice message or call some other phone number. In this case, the caller is tricked into believing that the callee is not available even if the callee is actually available. Note the fake callee's voicemail attack can be launched by the MITM at either the caller's or callee's side. Therefore, the fake callee's voicemail attack is possible even if one communication party (caller or callee) uses a PSTN phone.

Fake callee's voicemail at the caller's side. We use our Vonage and AT&T SIP phones to call a PSTN phone. The left part of Fig. 4 illustrates the corresponding message flow. After dialing the callee's phone number, the caller's SIP phone sends the corresponding `INVITE` message to the SIP server of its service provider. The MITM intercepts the `INVITE` message in step (1), and replies with a fake `200 OK` message in step (2). The caller's SIP phone is tricked by the fake `200 OK` message and sends back an `ACK` message in step (3). This establishes the voice RTP session between the caller's SIP phone and the MITM. In step (4), the MITM sends the crafted RTP stream to the caller's SIP phone and the caller hears bogus voice greeting "xxx can not take your call, please leave a brief message or call 1-xxx-xxx-xxx". Then the MITM starts to record the RTP stream from the caller's SIP phone. Once the caller hangs up, his SIP phone sends a `BYE` message to the MITM in step (5), the MITM replies with a `200 OK` message to the caller's SIP phone in step (6), which terminates the call.

When we use the Gizmo softphone to call the PSTN phone, the Gizmo softphone does not respond with an `ACK` message to the MITM after receiving the

`200 OK` message from the MITM in step (2). After further investigation, we find that the Gizmo phone does not follow the SIP specification [3] exactly:

– The IP address and the port number of the RTP server are specified by the Gizmo softphone in the `INVITE` message in step (1), rather than by the SIP server in the `200 OK` message in step (3).
– Gizmo softphone uses some proprietary protocol, running on TCP port 443, to exchange some signaling information with Gizmo RTP server in step (G1) and (G2). Step (G1) is necessary for establishing the SIP call, and if packets in step (G1) are dropped by the MITM, the Gizmo softphone will not proceed to step (3). Once the MITM allows the traffic in step (G1), the the Gizmo softphone proceeds to establish the call with its SIP server. Step (G2) is used to terminate the SIP call.

After implementing these special handling, the MITM is able to spoof the callee's voicemail to calls from the Gizmo softphone.

Fake callee's voicemail at the callee's side. In this experiment, we use a PSTN phone to call our Vonage and AT&T SIP phones. The right part of Fig. 4 illustrates the corresponding message flow. After the caller dials the phone number of a SIP phone from a PSTN phone, the SIP server sends an `INVITE` message to the SIP phone in step (1'). The MITM intercept the `INVITE` message and responds with a `200 OK` message in step (2'). The SIP server thinks it is from the SIP phone and sends an `ACK` message in step (3'). Now the voice RTP session between the SIP server and the MITM is established, and the MITM sends the RTP server crafted RTP stream in step (4'). As a result, the caller from the PSTN phone hears the bogus voice greeting "XXX can not take your call, please leave a brief message or call 1-xxx-xxx-xxx". Then the MITM starts to record the RTP stream from the RTP server. Once the caller hangs up, the SIP server sends a `BYE` message to the MITM in step (5'). Finally, the MITM replies with a `200 OK` message to the SIP server in step (6'), which terminates the call.

Fake Caller's Voicemail Attack. When a caller wants to check his voicemail, he usually dials some voicemail access number (e.g., *123 for Vonage, *** for AT&T, 611 for Gizmo) and authenticates himself with a voicemail PIN. Again, the MITM can intercept the call and spoof the caller's voicemail. This not only allows the attacker to trick the caller with bogus voicemail messages but also let the attacker capture the caller's voicemail PIN.

Table 1. Differences between the Voicemail Services by Vonage, AT&T and Gizmo

Service Provider	Voicemail Access Number	Codec for voice	Payload Type for Event
Vonage	*123	G.711 PCMU(0)	101
AT&T	***	G.721(2)	100
Gizmo	611	iLBC(102)	106

We have experimented the fake caller's voicemail attack with Vonage, AT&T and Gizmo SIP phones. The SIP message flow is similar to the left part of Fig. 4. The MITM first spoofs the SIP server and establishes a RTP voice session with the voicemail caller. Then the MITM spoofs the voicemail service and interacts with the voicemail caller. In our implementation, we record the RTP stream from the real voicemail server and replay the recorded RTP stream to the voicemail caller. Therefore, what the voicemail caller hears is exactly the same as that from the real voicemail server. After the MITM prompts the voicemail caller, it waits for responses (e.g., PIN, functional choice) from the caller and responds accordingly.

We find that Vonage, AT&T and Gizmo use different codecs and RTP payload types for their voicemail traffic. Table 1 summarizes their differences in the codec and RTP payload type used. Specifically, the codecs for the RTP voice are G.711 PCMU, G.721 and iLBC respectively, and the payload types for the RTP event are 101, 100 and 106 respectively. Despite these differences in the voicemail implementation by Vonage, AT&T and Gizmo, our MITM is able to spoof the voicemail for all of them.

In summary, the MITM can spoof both the caller's and the callee's voicemail even if one side of the voice call uses a PSTN phone. Eventually, the caller is deceived to leave a voicemail or call another number while the callee does not even know he has been called. Meanwhile, the MITM can read the voicemail left by the caller and trick the voicemail caller with bogus voicemail messages. Furthermore, the MITM could obtain the voicemail PIN entered by the caller.

4.3 Circumventing Do Not Disturb

Do Not Disturb enables VoIP users to block all incoming phone calls during a short time, e.g., 30 minutes. When we used a PSTN phone to call a Vonage or an AT&T VoIP phone with Do Not Disturb enabled, the call was forwarded to the voicemail system. This indicates *Do Not Disturb* is effective if the call goes through SIP servers. However, we can circumvent *Do Not Disturb* by calling the Vonage or the AT&T phone directly.

Fig. 5 shows the network setup of circumventing *Do Not Disturb*. Note unlike previous experiment, this experiment only requires a remote attacker. Fig. 6 depicts the message flow of circumventing *Do Not Disturb* attacks on the Vonage phone and AT&T phone.

In step (1) and (1'), the remote attacker sends an `INVITE` message to the SIP phone. Note to make the `INVITE` message accepted by the SIP phone, the IP address should be spoofed as that of a **real SIP server**. Otherwise, the SIP phone would omit this `INVITE` message. In the `INVITE` message, the IP address and port number for the RTP stream on the caller side are set to the remote attacker's IP address and 12345 respectively. Since the SIP specification does not require to authenticate SIP messages from SIP servers, the SIP phone will accept this `INVITE` message. Then in step (2-3) or (2'-3'), the SIP phone sends back `Trying,Ringing` messages to the real SIP server, and begins to ring. At

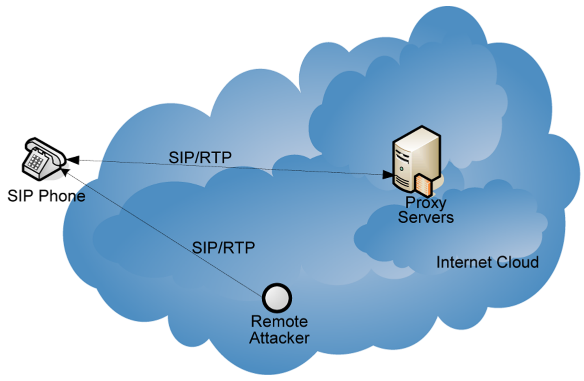


Fig. 5. Testbed Setup of Circumventing Do Not Disturb

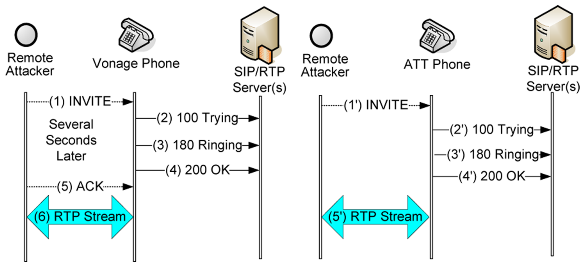


Fig. 6. Message Flow of Circumventing Do Not Disturb

this point, the called party begins to hear annoying ring tone. Consequently, *Do Not Disturb* is bypassed.

Actually, the attacker can even successfully establish a SIP call with these phones and send voice spam by exploiting some implementation flaws. After the receiver picks up the phone, the SIP phone responds with a `200 OK` message to its real SIP server. According to the SIP specification, to establish a SIP call, the remote attacker needs to send an `ACK` message to complete an `INVITE/200OK/ACK` three-way handshake. Since the `200 OK` message is sent to the real SIP server, the remote attacker does not know the time when the receiver picks up the phone. To solve this problem, The remote attacker can guess the time interval between the `INVITE` message and the `200 OK` message, e.g., 5 seconds. Then 5 seconds after sending the `INVITE` message, the remote attacker sends an `ACK` message with a spoofed source IP address to the SIP phone. In addition, to ensure the correctness, the callee should check whether the Tag value in the To field of the `ACK` message is the exactly same as that in the `200 OK` message. Since the remote attacker can not see the `200 OK` sent to the SIP server, the remote attacker can not craft an `ACK` SIP message with an correct Tag value. After further investigation, we find:

- The Vonage phone does not check the Tag value, and accepts an `ACK` message with a random Tag value.
- The AT&T phone does not need a complete `INVITE/200OK/ACK` three-way handshake before sending RTP stream. As soon as the receiver picks up the phone, the AT&T phone begins to send RTP stream to the remote attacker.

Consequently, the remote attacker can exploit these implementation flaws to talk to both the Vonage and AT&T phone.

5 Discussion

We have demonstrated that a MITM or remote attacker, could successfully launch many attacks on phone features in deployed SIP-based VoIP systems. These attacks exploit the inherent vulnerabilities in current deployed VoIP systems.

- Only SIP authentication mechanism is employed to protect SIP messages. Due to the weaknesses in SIP authentication, an attacker can freely craft unprotected SIP messages (e.g., Trying, Ringing, 200 OK, ACK), and SIP messages from a SIP server to a SIP phone. In addition, the attacker can also modify unprotected SIP fields (e.g., SIP message body, From, To).
- Since RTP voice stream is unencrypted and unauthenticated, the attack can easily capture voice traffic, generate bogus RTP stream and talk to VoIP phones.

Leveraging these weaknesses in SIP and RTP, the attacker can seamless spoof the SIP and RTP server when talking to the caller, and seamless spoof the callee when talking to the SIP server and RTP server. Accordingly, the attacker can readily hijack 911 calls, spoof the caller's and the callee's voicemail, and make a call to the selected phone number. Note during an attack, both a VoIP phone and the PSTN or cell phone communicating with the VoIP phone are potential victims. In addition to the features that we have investigated in this paper, we believe other phone features, e.g., 411, all have similar exploitable vulnerabilities.

Since our attacks target VoIP end users, intrusion defense measures deployed on VoIP servers side are ineffective to mitigate the attacks. To defeat or minimize these attacks, the best solution is to provide privacy, integrity and authentication protection for SIP signaling messages and RTP voice streams. For example, SIP over TLS and Secure Real-time Transport Protocol (SRTP) could be deployed to protect SIP messages and RTP voice streams. Unfortunately, we still have not widely seen such a deployment. Considering Vonage and AT&T Callvangtage's VoIP phones are being utilized to replace traditional PSTN phones, and millions of VoIP subscribers are using their VoIP phones to perform many security critical activities, e.g, phone banking and 911 calls, the current state of VoIP security is not optimistic.

6 Related Work

Most previous work is on the defense side. Arkko et al [11] proposed a scheme to negotiate the security mechanism used between a SIP phone and and its next-hop SIP entity. Baugher et al [12] proposed SRTP to protect the RTP traffic. However, none of them are widely being used. Reynolds et al [13] proposed multi-protocol protection against flood-based DoS attacks on VoIP networks. Wu et al [14] described a cross protocol intrusion detection architecture for VoIP environments. Sengar et al [15] proposed an intrusion detection system based on interactive protocol state machines. The above intrusion detection systems or methods are deployed on VoIP servers side, and ineffective to defend against the attacks we proposed.

Mintz-Habib et al [16] proposed a VoIP emergence service architecture and developed a prototype system. Zhang et al [8] [10] implemented four billing attacks on deployed VoIP systems, and demonstrated a remote attacker can become a MITM by exploiting some implementation flaws in a VoIP phone. Wang et al [17] systematically investigated the trust issue of SIP-based VoIP and identified the voice pharming attack on VoIP systems. McGann and Sicker [18] analyzed detection capability of several VoIP security tools: SiVuS,PROTOCOS [19], SIP Forum Test Framework [20], and some commercial products. They showed that there exists a large gap between known VoIP security vulnerabilities and the tool's detection capability.

7 Conclusion

In this paper, we empirically investigated the trustworthiness of phone features in leading deployed SIP-based VoIP systems. We demonstrated that the MITM could transparently hijack E911 calls and spoof the PSAP. In addition, it can spoof voicemail and use it for many potential voicemail related frauds. Finally, a remote attacker can circumvent *Do Not Disturb* and make annoying phone calls.

We hope our work can raise the awareness of millions of VoIP subscribers that the currently deployed VoIP phone features are not as trustworthy and reliable as expected. Before VoIP service providers employ SIP over SSL and SRTP to protect SIP signaling messages and RTP voice streams, a VoIP subscriber should be aware of the risk associated with current VoIP services.

Acknowledgments. This work was partially supported by NSF Grants CNS-0524286 and CCF-0728771.

References

1. First Report and Order and Notice of Proposed RuleMaking, `http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-05-116A1.pdf`
2. How To Deal With Voicemail When Prospecting, `http://www.content4reprint.com/business/network-marketing/how-to-deal-with-voicemail-when-prospecting.htm`

3. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol. RFC 3261, IETF (June 2002)
4. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 1889, IETF (January 1996)
5. Vonage, http://www.vonage.com/
6. AT&T's CallVantage, https://www.callvantage.att.com/
7. Gizmo, http://www.gizmoproject.com/
8. Zhang, R., Wang, X., Yang, X., Jiang, X.: Billing Attacks on SIP-Based VoIP Systems. In: lst USENIX Workshop on Offensive Technologies (WOOT 2007) (August 2007)
9. US VOIP market shares, http://blogs.zdnet.com/ITFacts/?p=11425
10. Zhang, R., Wang, X., Farley, R., Yang, X., Jiang, X.: On the Feasibility of Launching the Man-In-The-Middle Attacks on VoIP from Remote Attackers. In: 4th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009), Sydney, Australia (March 2009)
11. Arkko, J., Torvinen, V., Camarillo, G., Niemi, A., Haukka, T.: Security Mechanism Agreement for the Session Initiation Protocol (SIP). RFC 3329, IETF (January 2003)
12. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: The Secure Real-time Transport Protocol (SRTP). RFC 3711, IETF (March 2004)
13. Reynolds, B., Ghosal, D.: Secure IP Telephony Using Multi-layered Protection. In: 10th Network and Distributed System Security Symposium (NDSS 2003) (Feburary 2003)
14. Wu, Y., Bagchi, S., Garg, S., Singh, N.: SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments. In: 34th International Conference on Dependable Systems and Networks (DSN 2004), pp. 433–442 (July 2004)
15. Sengar, H., Wijesekera, D., Wang, H., Jajodia, S.: VoIP Intrusion Detection Through Interacting Protocol State Machines. In: 36th International Conference on Dependable Systems and Networks (DSN 2006) (June 2006)
16. Mintz-Habib, M., Rawat, A., Schulzrinne, H., Wu, X.: A VoIP Emergency Services Architecture and Prototype. In: 14th International Conference on Computer Communications and Networks (ICCCN 2005) (October 2005)
17. Wang, X., Zhang, R., Yang, X., Jiang, X., Wijesekera, D.: Voice Pharming Attack and the Trust of VoIP. In: 4th International Conference on Security and Privacy in Communication Networks (SecureComm 2008) (September 2008)
18. McGann, S., Sicker, D.C.: An analysis of Security Threats and Tools in SIP-Based VoIP Systems. In: Second VoIP Security Workshop (2005)
19. PROTOS SIP Fuzzer, http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/
20. SIP Forum Test Framework, http://www.sipfoundry.org/sip-forum-test-framework/sip-forum-test-framework-sftf.html

Reconstructing a Packed DLL Binary for Static Analysis*

Xianggen Wang[1,2], Dengguo Feng[2], and Purui Su[2]

[1] University of Science and Technology of China
[2] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences

Abstract. DLLs (Dynamic Link Libraries) are usually protected by various anti-reversing engineering techniques. One technique commonly used is code packing as packed DLLs hinder static code analysis such as disassembly. In this paper, we propose a technique to reconstruct a binary file for static analysis by loading a DLL and triggering and monitoring the execution of the entry-point function and exported functions of packed DLLs. By monitoring all memory operations and control transfer instructions, our approach extracts the original hidden code which is written into the memory at run-time and constructs a binary based on the original DLL, the codes extracted and the records of control transfers. To demonstrate its effectiveness, we implemented our prototype ReconPD based on QEMU. The experiments show that ReconPD is able to analyze the packed DLLs, yet practical in terms of performance. Moreover, the reconstructed binary files can be successfully analyzed by static analysis tools, such as IDA Pro.

Keywords: Security Analysis, Malware Analysis, Binary Reconstructing, Dynamic Analysis, Static Analysis.

1 Introduction

Reverse engineering becomes a prevalent technique to analyze malware. To make the analysis more difficult, malware writers usually protect their programs including executables and DLLs by various anti-reverse engineering techniques. One of the most popular anti-reverse engineering methods is binary code packing [18,20]. It is impossible to do static analysis on packed malware because of the inaccuracy of disassembler. This paper focuses on identifying and extracting the hidden code generated using binary code packing and reconstructing a binary for static analysis.

To identify and extract the hidden code in packed binary programs various tools have been developed. However, most of commonly known tools such as PEiD [4]

* Supported by National Natural Science Foundation of China (No.60703076) and National High Technology Research and Development Program of China (No.2006AA01Z412 and No.2007AA01Z451)

are only valid for some special known packing algorithms applied on the packed executables; what is more, they are usually unable to analyze packed DLLs.

DLL is commonly used in the malware. Windows makes certain features available only to DLLs. For example, you can install certain hooks (set using **SetWindowsHookEx** and **SetWinEventHook**) only if the hook notification function is contained in a DLL. As a number of critical technologies and codes are packed in DLL, it is becoming one important target of anti-reverse engineering protection.

By now, the difficulty of statically analyzing a packed binary program is how to correctly disassemble them. One of approaches to solve this problem is to reversely analyze the packing algorithm and develop specific automatic unpacking tools. But this method is only valid for known packing tools(such as UPX). In fact, it is easy for malware writers to implement new anti-reverse engineering algorithms and tricks. Dynamic analysis is a promising solution to the problem of hidden code extraction because it does not depend on signatures. The original code or its equivalent must eventually be restored in memory and get executed at some point at run-time regardless of what packing algorithms might be applied [20]. By taking advantage of this intrinsic nature of packed binary programs, we could potentially extract the original hidden binary code or its equivalent and reconstruct a binary for static analysis by patching the extracted code on the original packed binary program. Most of the work to date on identification and extraction of the original hidden code has focused on executables. This paper, by contrast, focuses on the DLL.

In this paper, we present a fully dynamic approach for automatically identifying packed DLLs, and extracting the original hidden code which is dynamically generated and executed at run-time and additional information useful for reconstructing of the packed binary. Finally we reconstruct a binary which can be directly analyzed by static analysis tools, such as IDA Pro. This method is valid for unknown packing algorithms without the complex reverse analysis. In summary, the contributions of this paper are as follows:

1. **Propose a general and fully dynamic approach for identifying and extracting the original hidden code of packed DLLs:** Previous work relies on either heuristics of known packing tools or the accuracy of the disassembler and is only valid for packed executables. In this paper, we present a binary extraction technique which is applicable to packed DLLs. This method is fully dynamic and thus does not depend on the program disassembly or the known signatures of packing algorithms. It triggers the execution of entry-point function and exported functions of a DLL, monitors the execution and records related memory operations by shadow memory and extracts the content of the memory which is dynamically generated and executed.
2. **Provide a method to reconstruct a DLL binary file based on the execution trace:** By patching the extracted code on an original packed DLL binary file, we reconstruct a binary file which can be directly disassembled.
3. **Implement and evaluate ReconPD, an automated framework for extracting hidden code and reconstructing binary files.** Applying

our proposed technique, we build a framework for automatically examining DLL binaries, extracting their original hidden code and reconstructing a binary file based on the extracted code and additional information. The reconstructed binary file can be successfully analyzed by static techniques. Based on the prototype, we have successfully done a series of experiments on the analyses of packed DLL binary files. We also present the evaluation results of ReconPD, demonstrating that it is applicable to analyze packed DLL binary.

The rest of the paper is organized as follows: In Section 2 we review the related work. Section 3 presents the problem definition and describes the proposed method. In Section 4 we provide details of the technique and the prototype implementation. Section 5 shows the experimental results. Finally, Section 6 concludes the paper.

2 Related Work

Static analysis is one of the most popular software analysis methods nowadays, it is widely used in software reverse engineering [1] and software security analysis [2,3]. Static analysis tools can analyze intermediate binary code that is unable to run, since they don't need to really execute the code at all. And they have a good global view sight of the whole binary code and can figure out the entire program logic before running it.

To protect the privacy of software and prevent reverse engineering, software writers usually have their programs heavy-armored with various anti-reverse engineering techniques [7,8]. Such techniques include SMC (Self-Modifying Code) [9,10], encryption [11,12], binary and source code obfuscation [14,15], control-flow obfuscation [13], instruction virtualization [5,6], and binary code packing [18]. The development of anti-reverse technology makes static analysis more and more difficult. Although dynamic analysis can compensate static analysis for its effectiveness and accuracy to a certain extent, static analysis has some advantages that dynamic analysis doesn't have such as the more comprehensive view of a program's behavior.

Currently, extracting and re-building the original program from a protected binary has been one of the major challenges for software reverse engineers and security community.

Identifying and extracting the original hidden code
PolyUnpack [18] is a general approach for extracting the original hidden code without any heuristic assumptions. As a pre-analysis step, PolyUnpack disassembles the packed executable to partition it into the code and data sections, and then it executes the binary instruction by instruction, checking whether the instructions are in the code section or not. However, in terms of performance, disassembling binary executables as being done in [18] and [19] is an arduous task and correctly disassembling a binary program is challenging and error-prone, as demonstrated in [19]. In Addition, PolyUnpack can not extract the hidden code inside the DLL.

Min Gyung Kang et al. proposed a fully dynamic method Renovo [20] which monitors currently-executed instructions and memory writes at run-time. This approach inserts a kernel module into the emulated system and registers several call-back functions, by observing the program execution. Compared with it, our approach doesn't need touch anything in GusetOS and is completely transparent to analysis target; moreover, our method is applicable to a packed DLL binary and we can reconstruct a binary file for static analysis by monitoring the control transfers.

Re-building the original program
PEiD [4] is a tool for identifying compressed Windows PE binaries. Using the database of the signatures for known compression and encryption techniques, it identifies the packing method employed and suggests which unpacker tool can be applied. However, despite their ability to perfectly restore the original program, executables packed with unknown or modified methods are beyond the scope of this approach.

Christodorescu et al. proposed several normalization techniques that transform obfuscated malware into a normalized form to help malware analysis [16]. Their unpacking normalization is similar to our approach. Its basic idea is to detect the execution of newly-generated code by monitoring memory writes after the program starts. Our approach goes further; we identify and monitor memory modification and execution in all code, data, stack and heaps, and also our approach can tackle dynamic code generation inside the packed DLL binary.

3 Problem Statement

One of the most popular anti-reverse engineering methods is binary code packing which is usually used to protect binary program against reverse engineering and other attacks. Code packing transforms a program into a packed program by encrypting or compressing the original code and data into packed data and associating it with a restoration routine [20]. It is impossible to analyze the packed binary program using static techniques, thus, the target of our work is to identify them and extract the original hidden code in packed binary files. We generate a binary file which can be successfully analyzed by static techniques by patching a packed binary file with extracted code and restoring its control transfers.

The protecting procedure can be expressed as below:

$$dummy_i = HR_i(target_i, key_i)$$

A *hiding routine* HR_i is a set of instructions which camouflage a target instruction $target_i$ with a dummy instruction $dummy_i$ based on encrypt key key_i. The key may be null or an invariant if the hiding routine is just a transformer.

A *restoring routine* RR_i is a set of instructions which un-camouflage an original target instruction hidden by a dummy instruction based on the decrypt key key_i. RR_i can be viewed as a set of instructions that replaces $dummy_i$ with $target_i$ at run-time [9]. Fig.1 shows how a packed program works [20].

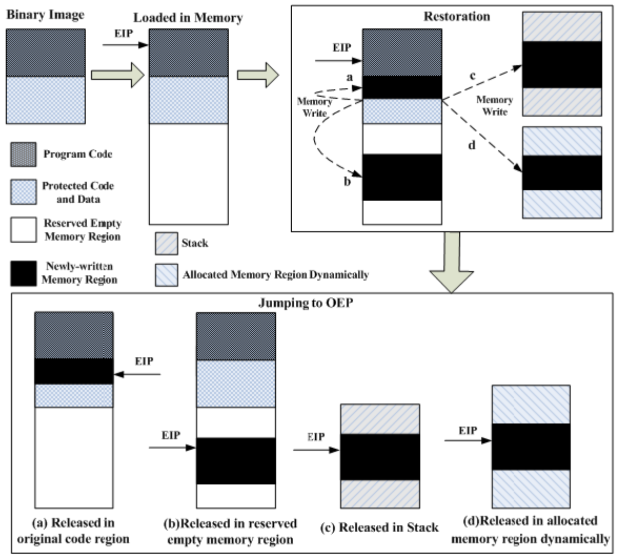

Binary Image
Loaded in Memory
Restoration
EIP
a
b
c
d
Memory Write
Memory Write
Program Code
Protected Code and Data
Reserved Empty Memory Region
Newly-written Memory Region
Stack
Allocated Memory Region Dynamically
Jumping to OEP
(a) Released in original code region
(b)Released in reserved empty memory region
(c) Released in Stack
(d)Released in allocated memory region dynamically

Fig. 1. How a packed program works

The restoring procedure during execution can be described as below:

$$target_i = RR_i(dummy_i, key_i)$$

RR_i is a reverse procedure of HR_i. In an analysis, HR_i is unavailable usually, so we can only analyze RR_i. But it's difficult and time-consuming to analyze RR_i directly, so we observe the execution of RR_i in order to identify and extract the hidden code it releases.

In the following discussion, we assume there is a valid key, and also we assume that at least a part of dynamically released code will be executed at run-time.

RR_i may be only used to release the code, or used to release both code and data. In the procedure of reconstructing a binary, we must disable the RR_i code generation but retain its data generation. Otherwise, it may disturb the reconstructed binary's execution.

In the procedure of restoring code, the memory used to store code may be used to store one block of code or blocks of code iteratively. That is, there may be several $dummy_i$. We suppose that one of them is $dummy_i$ whose address is A_d^i, and the related code released is $target_i$, whose address is A_t^i. There are two different models:

1. for any $i, j > 0$, and $A_d^i \neq A_d^j$, there is $A_t^i \neq A_t^j$;
2. there exist $0 < i < j$, making $A_t^i = A_t^j$,but $A_d^i \neq A_d^j$;

In the first model, each restored code block is stored in different memory regions, that is, they do not overlap with each other. In the second, there are at least two released blocks stored in the same memory region, resulting in some newly-generated code overlapping with each other. For the second model, we need to transform the addresses because of the address confliction when we reconstruct the binary. The details will be discussed in Section 4.

4 System Design and Implementation

To demonstrate the effectiveness of our approach, we design and implement a system, ReconPD, to automatically identify packed DLLs and extract their original hidden code. Fig.2 depicts an overview of ReconPD. The prototype ReconPD is a subsystem of WooKoo, a malware dynamic binary analysis platform, which was developed by us based on QEMU [22]. We modified QEMU to monitor the dynamic execution of a target executed in the Guest OS.

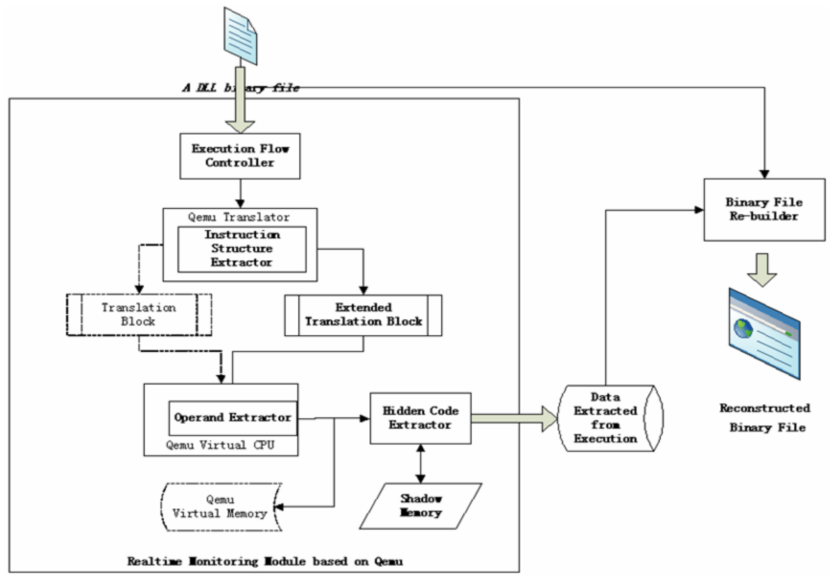

A DLL binary file
Execution Flow Controller
Qemu Translator
Instruction Structure Extractor
Translation Block
Extended Translation Block
Operand Extractor
Qemu Virtual CPU
Hidden Code Extractor
Qemu Virtual Memory
Shadow Memory
Realtime Monitoring Module based on Qemu
Data Extracted from Execution
Binary File Re-builder
Reconstructed Binary File

Fig. 2. The Framework of ReconPD

ReconPD extended QEMU Translator by adding the module of Instruction Structure Extractor and extended QEMU Virtual CPU by adding the module of Operand Extractor. It also extended the data structure of TB (Translation Block) to store the information of the instruction structure and added a new data structure of shadow memory. When analyzing a DLL, we run it in an *emulated environment. Execution Flow Controller* loads a DLL and triggers the entry-point function and exported functions. *Hidden Code Extractor* observes its execution, consults the *shadow memory*, and determines if any hidden code is currently executed. If so, it extracts the hidden code and other data. Finally, *Binary File Re-builder* reconstructs a binary file based on the original DLL and the data collected by Hidden Code Extractor. We will present these components and provide some implementation details in turn, and discuss the limitations of the current implementation of ReconPD.

4.1 Recognizing and Extracting the Hidden Code

No matter what packing methods are applied and where the hidden code is restored, the original program code and data should eventually be present in memory to be executed, and also the instruction pointer should jump to the OEP of the restored program code which has been written in memory at run-time [20]. Taking advantage of this inevitable nature of packed DLLs, we propose a technique to dynamically extract the hidden original code from the packed DLLs by examining whether the current instruction is newly-generated after the DLL is loaded.

Execution Flow Controller. Before an application (or another DLL) can call functions in a DLL, the DLL's file image must be mapped into the calling process' address space. Normally, when a DLL is mapped into a process' address space, the system calls a special function in the DLL (we call it the entry-point function, usually **DllMain**) which is used to initialize the DLL. Usually the restoration of the hidden code and data is a part of the initialization process for packed DLL.

Windows is unable to launch DLL directly, so in this paper we make use of a DLL loader named LoadDLL.exe for loading a DLL by calling Windows API **LoadLibraryEx**. In order to monitor the execution of the entry-point function of a DLL, we set *dwFlags* to **DONT_RESOLVE_DLL_REFERENCES** to prevent the operation system from calling the entry-point function to initialize the DLL. To obtain a comprehensive analysis of the DLL as the entry-point function is executed, we control the execution flow and trigger all exported functions of the DLL. As the return instruction of the entry-point function is "RET 0C", we take it as a sign of the end of the execution of the entry-point function. An exported function is typically allowed to run until it exits normally or crashes.

Considering that determining whether a program has hidden code or not is an *un-decidable* problem [18], we introduce a configurable time-out parameter into the system. We terminate the extraction procedure if we do not observe any hidden code being executed within this time-out. In the experiments, we set this parameter to be 30 minutes.

Shadow Memory. Based on the above inevitable instinct nature of packed DLLs, shadow memory is introduced to record every memory modification. According to whether the written content is executed or not, we can determinate whether it is code or data. Thus, we need to record the following information for each byte of memory:

1. EIP_M: The instruction modifying this byte.

2. *EIP_C*: The instruction referring this byte, such as directly jumping to this memory by JMP and JNZ, or calling the function whose entry point is this memory.
3. *State*: The state of this byte, there are three possible states, *clean*, *dirty*, and *executed.* Fig.3 shows how the states transfer.

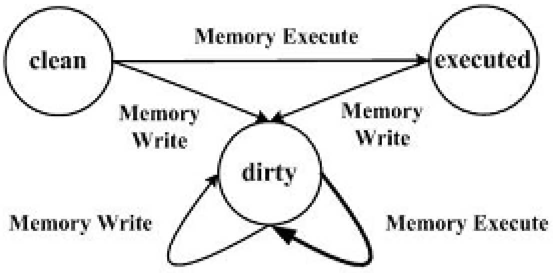


Fig. 3. How a memory state transfers

We just need to pay attention to the state *dirty* and *clean* in order to recognize and extract the original hidden code. In ReconPD, we assign a block of shadow memory for each monitored memory region (including static region and dynamic region such as stack and heaps allocated during execution) of the DLL-calling process (LoadDll.exe).

In order to reconstruct the control flow, we record *control transfer instructions*, such as JNZ and CALL, which will change the execution flow. During the reconstruction of a binary file, we modify their destination address if the target code is reassigned.

Single-Step Monitoring. Our monitoring on the execution needs to intercept each instruction to get the real values of operands, so we make the monitored code execute in single-step mode and each TB contain one instruction in QEMU.

In QEMU, there is a flag of virtual CPU for single-step execution mode. If the flag is set, the code of both operation system and applications is executed in single-step mode. Furthermore, the flag is perceptible for applications in the GuestOS, and the analyzed object may detect it and destroy itself or stay inactive if it found the flag is set. In ReconPD, an additional flag, *TSingleStep*, was introduced, which is transparent for applications. When *TSingleStep* is set, the translator makes sure that there is only one instruction in each TB of the target process, and instructions of other process are executed as normal.

By the method above, we not only keep the analysis transparent for the target process but also improve the analysis efficiency by avoiding the whole system running in single-step mode.

Instruction Interception. During monitoring the execution, we intercept the following two classes of instructions:

Memory Modifications: including MOV, MOVS, STOSB and so on. These operations are used to write dynamic code into memory. The stack operations such as POP are not included, which are rarely used to generate code.

Control Transfers: including direct jumps/calls, indirect jumps/calls, and return instructions, which determine the control flow of the execution.

ReconPD firstly records the structure of instruction during the translation and extracts the real values of operands during run-time.

4.2 Reconstructing Binary Files

In this phrase, we patch the hidden code exacted on the packed DLL binary file and restore the control transfers to generate a binary file for static analysis. There are two parts of work during this phrase.

Patch the original hidden code extracted on the packed DLL binary file

1. The code is released in code region and covers the original protected code just as showed in Fig2(a). We write the dynamic code into the binary file where this block is mapped from the binary file.
2. The code is released in reserved empty memory of code region without covering the original code just as showed in Fig2(b), we set $SizeOfRawData$ of the code section to $VirtualSize$, then we write the dynamic code into the revised binary file where this block is mapped from the binary file.
3. The code is released in data region, such as stack and heap just as showed in Fig2(c,d), we expand a new section into the PE binary and the code is written into the expanded section. It's a mature technique to expand the PE binaries, thus, we do not discuss it in this paper.
4. There are two or blocks written into the same memory being covered each other, the first block write into the binary file where this block is mapped from, and the rest are written into the expanded section in the binary file.

In the restoration process, we should modify *dest* of the control transfer to the correct address where the instruction is reassigned in a binary file. New value of *dest* can be easily got by address mapping. But sometimes, we can not directly update it. For example, some branch instructions like "jmp ebx;" may transfer control to different dynamic generated code blocks at different conditions. For this case, we make the branch instruction transfer control to a dispatch routine, which will mimic the original branch conditions to call different dynamic generated code blocks at different time. By this way, though we can not completely rebuild the original logic of the execution, we can still provide all possible control transfers, which is much helpful for static analysis.

Disable the restoration, preventing it from modifying the patched code. When releaser's code generation behavior is disabled, we should consider whether the releaser dynamically modifies code only or both code and data. If a releaser only modifies code, we could disable it simply by replacing it with NOP instruction. Or else, to keep the original program logic, we have to reserve its function of modifying data and prevent it from modifying code. So we replace the releaser with a unconditional jump, which transfers control to a judge function in the new code section, and the judge function is an encapsulation of the releaser, which behaves just like the releaser if the modification's target is data but does nothing if the target is code.

5 Evaluation

The experiments are finished on the WooKoo Platform, based on Fedora Core 3 Linux 2.6.9-1.667smp and QEMU 0.8.2. The experiments are all finished with GuestOS of Windows XP Professional SP2. In this section, we describe two experiments and present the evaluation results, demonstrating that ReconPD is an accurate and practical solution for extracting the original hidden code of packed DLLs and the reconstructed binaries can be successfully analyzed by static analysis tools such as IDA pro.

5.1 Synthetic Samples

Because of the difficulty of collecting wild samples, we have to develop some samples. To verify that ReconPD generates accurate results, we have tested ReconPD against the synthetic sample programs generated by using a packing tool developed by us. The packing tool applies different packing techniques as well as encryption, code obfuscation to thwart reverse engineering.

We use some DLLs belong to Microsoft Windows Operating System as the original binary to generate synthetic packed program samples. With the knowledge that the packing tool usually restore and execute the original binary instructions at run-time, we could verify the correctness of our extraction technique by comparing the extracted hidden code regions with the .text section of the original binary.

Table 1. The Experimental Results of Packed DLLs

Files	Size of Packed Files(KB)	Size of Shadow Memory(KB)	Time without monitor(Seconds)	Time under monitor (Seconds)	Reconstructed Binary
rasapi32.dll	234	2928	54	62	Loadable
jscript.dll	444	5376	38	45	Loadable
mfc42.dll	1008	12240	32	39	Loadable
msi.dll	2792	33744	22	25	Loadable
shell32.dll	8112	97728	674	681	Loadable

As shown in Table 1, ReconPD fully extracted the original binaries processed by our packing tools. And the reconstructed DLL binary files could be loaded and disassembled by IDA Pro properly. From the experimental results, we found the size of shadow memory is almost 12 times of DLLs.

5.2 Performance Overhead

We evaluated the performance by running the samples with and without monitor in single step mode. From the results shown in Table 1, we found that the monitor decreased the performance about 18.9%. Without the monitor, we also compared the performance of running in single step mode with that of running in normal mode. We found that the performance decreased greatly by single step mode even we have optimized it by only keeping the target process in single-step mode but not the whole system as QEMU. Considering that ReconPD is aiming to provide hidden code extraction environment for malware analysis which usually takes several hours to days, this degree of slowdown in initial execution time is tolerable.

5.3 Limitations

Just as discussed above, there are some limitations of ReconPD.

Some exported functions crash at run-time because of the improper parameters provided by us. We are able to analyze the number of parameters of exported functions but not recognize the data type of parameters. We may patch both the data and code to reconstruct complete binary while there may be no comprehensive solution to this problem. We have intercepted typical memory write instructions. However, there is a method to evade the monitor by replacing some instructions. For example, *"XOR EAX,EAX; ADD EAX,EBX"* can replace the instruction *"MOV EAX,EBX;"*. In the future, we will make it support much more instructions.

Finally, some of the reconstructed binary are unable to be loaded because ReconPD does not patch the data. It just monitors and restores both the original hidden code and control transfers.

6 Conclusion

We proposed a technique to reconstruct a binary file for static analysis by monitoring the executions of packed DLLs. This method is valid for unknown protection algorithms without the complex reverse analysis. To demonstrate the idea, we have implemented ReconPD based on QEMU. By triggering the execution of entry-point function and exported functions of a DLL, and monitoring all the memory operations and control transfer instructions, it identifies and extracts the code which is written into the memory during execution and constructs a binary file based on the original binary, the codes extracted and the records of control transfers. We show that ReconPD can successfully analyze packed DLLs and the reconstructed binary can be successfully analyzed by static analysis tools, such as IDA Pro.

References

1. Balakrishnan, G., Gruian, R., Reps, T., Teitelbaum, T.: CodeSurfer/x86—A platform for analyzing x86 executables. In: Bodik, R. (ed.) CC 2005. LNCS, vol. 3443, pp. 250–254. Springer, Heidelberg (2005)

2. Christodorescu, M., Jha, S.: Static Analysis of Executables to Detect Malicious Patterns. In: Usenix Security Symposium (2003)
3. Christodorescu, M., Jha, S., Seshia, S., Song, D., Bryant, R.: Semantics-aware Malware Detection. In: IEEE Symposium on Security and Privacy (2005)
4. PEiD, http://www.secretashell.com/codomain/peid/
5. Themida, http://www.oreans.com/
6. Yoda Protector, http://sourceforge.net/projects/yodap/
7. van Oorschot, P.C.: Revisiting software protection. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 1–13. Springer, Heidelberg (2003)
8. Wang, P.: Tamper Resistance for Software Protection, Master Thesis, Information and Communications University, Korea (2005)
9. Kanzaki, Y., Monden, A., Nakamura, M., Matsumoto, K.: Exploiting self-modification mechanism for program protection. In: Proc. of the 27th Annual International Computer Software and Applications Conference, pp. 170–181 (2003)
10. Giffin, J.T., Christodorescu, M., Kruger, L.: Strengthening Software Self-Checksumming via Self-Modifying Code. In: 21st Annual Computer Security Applications Conference, pp. 23–32 (2005)
11. Albert, D.J., Morse, S.P.: Combating Software Piracy by Encryption and Key Management. Computer (1984)
12. Lee, J.-W., Kim, H., Yoon, H.: Tamper resistant software by integrity-based encryption. In: Liew, K.-M., Shen, H., See, S., Cai, W. (eds.) PDCAT 2004. LNCS, vol. 3320, pp. 608–612. Springer, Heidelberg (2004)
13. Huang, Y.L., Ho, F.S., Tsai, H.Y., Kao, H.M.: A control flow obfuscation method to discourage malicious tampering of software codes. In: ASIACCS 2006, computer and communications security, New York, NY, USA, p. 362 (2006)
14. Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassembly. In: CCS 2003, New York, NY, USA, pp. 290–299 (2003)
15. Wroblewski, G.: General method of program code obfuscation. In: Proc. Int. Conf. on Software Engineering Research and Practice (SERP) (2002)
16. Christodorescu, M., Kinder, J., Jha, S., Katzenbeisser, S., Veith, H.: Malware normalization. Technical Report 1539, University of Wisconsin, USA (2005)
17. DataRescue SA. IDA Pro disassembler: Multi-processor, Windows hosted disassembler and debugger, http://www.datarescue.com/idabase/
18. Royal, P., Halpin, M., Dagon, D., Edmonds, R., Lee, W.: PolyUnpack: Automating the hidden-code extraction of unpack-executing malware. In: ACSAC 2006, USA, pp. 289–300 (2006)
19. Nanda, S., Li, W., Lam, L., Chiueh, T.: BIRD: Binary interpretation using runtime disassembly. In: CGO 2006, USA, pp. 358–370 (2006)
20. Kang, M.G., Poosankam, P., Yin, H.: Renovo: A Hidden Code Extractor for Packed Executables. In: The 5th ACM Workshop on Recurring Malcode (WORM) (2007)
21. Moser, A., Kruegel, C., Kirda, E.: Exploring Multiple Execution Paths for Malware Analysis. In: SP 2007, pp. 231–245 (2007)
22. Bellard, F.: QEMU, a Fast and Portable Dynamic Translator. In: FREENIX Track: 2005 USENIX Annual Technical Conference (2005)

Static Analysis of a Class of Memory Leaks in TrustedBSD MAC Framework*

Xinsong Wu[1,2], Zhouyi Zhou[1,2], Yeping He[1], and Hongliang Liang[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing China
[2] Graduate School, Chinese Academy of Sciences, Beijing China
xinsong@nfschina.com, {zhouyi,yphe}@ericist.ios.ac.cn, hongliang@ios.ac.cn

Abstract. Security labels of subjects and objects are crucial for some security policies and are an essential part of the TrustedBSD MAC framework. We find that security labels not being destroyed properly will result in memory leaks. This paper analyzes the security labels management of the TrustedBSD MAC framework and presents a path-sensitive static analysis approach to detect potential memory leaks caused by the security label management. This approach verifies complete destruction of security labels through compiler-integrated checking rules at compile-time. It achieves complete coverage of execution paths and has low false positive rate.

Keywords: static analysis, memory leak, TrustedBSD MAC framework, security label, mygcc.

1 Introduction

The TrustedBSD MAC (Mandatory Access Control) framework (hereinafter referred to as the MAC framework) is a group of hooks inserted in OS (Operating System) kernel to support different security policies [1]. While some security policies employ existing subject and object information (such as UNIX credential data and file permissions) most of the mandatory access control policies require additional information. For example, the BLP [2] confidentiality security policy makes decision based on subject and object sensitivity labels: subjects are assigned clearances, and objects are assigned classifications. Consequently, the MAC framework includes a number of hooks to manage security labels of subjects and objects.

At present, the hooks of the MAC framework are manually inserted into the OS kernel. Although the developers of the MAC framework are highly skilled and experienced hackers, errors are unavoidable when they deal with the highly complex OS kernel (millions of code lines) manually. For example, Figure 2 depicts an improper placement of the security label hooks found in our work on implementing a FreeBSD based trusted operating system. Consequently, the correctness of the hooks placement

* Supported by the National Natural Science Foundation of China under Grant No.90818012 and the National High-Tech Research and Development Plan of China under Grant No. 2007AA010601.

needs to be carefully analyzed. Furthermore, most of the OS kernels are still in the course of development, and more components are being added to them. So, the MAC framework needs to add new hooks or/and revise existing hooks to address the newly added components. This is an error-prone process because the framework developer may not fully understand the impact of the newly added components upon the OS kernel and the MAC framework. Therefore, we need an automatic approach to analyze the correctness of the hooks placement.

The first study on the correctness of the hooks placement is taken by the researchers of the IBM T.J.Waston Research Center. They analyze the authorization hooks placement of the LSM (Linux Security Module) framework [3] using both the dynamic approach and the static approach and successfully find some placement errors [4] [5]. While the efficiency of the dynamic approaches is higher than that of the static approaches, the execution paths coverage rate of the dynamic approach is poor. So, we decide to use the static approach. Zhang et al use the CQUAL [6] to implement the static analysis of hooks placement. However, the CQUAL is flow-insensitive, and cannot implement path-sensitive analysis. Therefore, using the CQUAL cannot find the error in Figure 1. This paper presents a path-sensitive static analysis approach to the correctness verification of the hooks placement related to security labels. This approach is based on Mygcc [7] which can execute user-defined checking rules at compile-time automatically.

The rest of the paper is organized as follows. Section 2 introduces the security label management of the MAC framework. Section 3 describes the extensions to the Mygcc. Section 4 gives result of the static analysis of the MAC framework. Section 5 discusses the related work, and Section 6 concludes the paper.

2 Security Label Management

First of all, we give the definition of controlled objects and controlled operations. Then, we analyze the security label management of the MAC framework and depict the memory leak problem resulting from the security label management.

2.1 Controlled Objects and Controlled Operations

A controlled object, also called a protected object, is the object that needs the protection provided by the MAC framework, such as *file*, *inode*, *task*, *socket* and *shared memory* etc.

A controlled operation is the system operation that performs on the controlled objects and may result in security-sensitive events, such as the read operation and the write operation. Though some operations perform on the controlled objects, they are not the controlled operation if they do not cause an information-flow, such as the lock operation and unlock operation of the *inode*.

2.2 Hooks Related to Security Labels

The MAC framework supports security labels through a policy-agnostic labeling primitive, which permits security policies to tag controlled objects with security labels for decision-making. Security labels are initialized, allocated, and destroyed along with their objects [8]:

Security label initialization occurs when the data structure for a controlled object is initialized, which permits security policies to allocate and initialize memory for the object.

Security label association or creation occurs when an initialized kernel structure is associated with an actual controlled object, such as a file or a process.

Security label destruction occurs when the controlled object is released by the kernel service implementing it. The MAC framework is given the opportunity to release storage for the security label, which permits security policies to free any allocated storage or references associated with that security label.

The security label structure stored in kernel data structures is maintained by the MAC framework: based on the life cycle of the data structure, the MAC framework provides per-object hooks for memory initialization, object allocation, and object destruction, as shown in Figure 1.

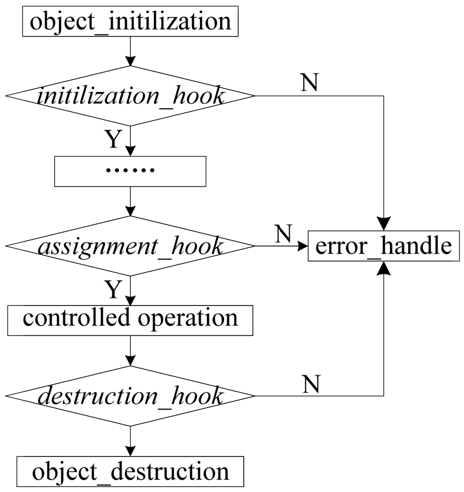


Fig. 1. Hooks related to security labels

2.3 Complete Destruction of Security Labels

As shown in Figure 1, the memory space of a security label must be freed before its host object is freed. Otherwise, there exists a potential memory leak. In the OS kernel, for a specific controlled object there may be many paths to its destruction point. Therefore, we need to ensure that at each path to the controlled object destruction point there is a security label destruction hook. We call this the complete destruction problem. For example, in Figure 2, if the execution follows the path from Line 201 to 227, the *inp* will be freed without its security label being freed. This violates the complete destruction rule of security labels and results in a memory leak.

```
/*$FreeBSD:src/sys/netinet/in_pcb.c,v1.197 $*/
int
in_pcballoc(struct socket *so, ...)
{
...
#ifdef MAC
error = mac_inpcb_init(inp, M_NOWAIT);
if (error != 0)
    goto out;
SOCK_LOCK(so);
mac_inpcb_create(so, inp);
SOCK_UNLOCK(so);
#endif

#ifdef IPSEC
error = ipsec_init_policy(so, &inp->inp_sp);
if (error != 0)
   goto out;
...
#if defined(IPSEC) || defined(MAC)
out:
if (error != 0)
   uma_zfree(pcbinfo->ipi_zone, inp);
#endif
return (error);
}
```

Fig. 2. Example of a memory leak

3 Extensions to Mygcc

The original Mygcc mainly focuses on the general errors in the OS kernel, so we need to extend it to perform the static analysis of the memory leaks related to security labels. First, we introduce the features of Mygcc. Then, we depict the extensions to Mygcc.

3.1 Features of Mygcc

After evaluating various analysis tools that may be suitable for the complete destruction analysis of the MAC framework, we choose the GCC compiler based Mygcc as our basic tool for further extension. Because Mygcc has the following features:

(1) Mygcc supports path-sensitive checking rules.

The complete destruction analysis depends on the path information. For example, though there exists the complete destruction problem in the code fragment in Figure 2, a path-insensitive tool can't find this error.

(2) Mygcc is an intra-procedure analysis tool.

In the OS kernels equipped with the MAC framework, most hooks and controlled operations are in the same function, which is suitable for Mygcc.

(3) Mygcc uses a configuration file to guide checking.

Tools like the CQUAL and SPLINT [9] are based on code annotation, which gives heavy burden to the developers. Furthermore, the annotation process may introduce new errors. However, Mygcc uses a single configuration file to guide checking, and the checking is executed in the compile process.

However, the original Mygcc mainly focuses on the general errors in the OS kernel, such as deadlock and null pointer reference etc. So, we need to extend it to address specific requirements of the complete destruction analysis.

3.2 Extensions to Mygcc

Based on the analysis of the MAC framework, we extend Mygcc to address three requirements: to execute checks from the function entry, to match the types of placeholders and to relate one placeholder to another one.

(1) Executing checks from the function entry

The original Mygcc must have a specific matching node as the starting point, and then traverse the abstract syntax tree until it encounters the *avoid* pattern (success), or another controlled operation (warning). However, the MAC framework needs only to match one controlled operation. So, we extend Mygcc to implement executing checks from the function entry.

The algorithm of the function entry checking is as follows:

```
proc check_entry(CFG, condate)
 substs ← ∅;
 foreach node t ∈ CFG do
  global_store ← ∅;
  if condate.from.format spec = “entry”
  then if match(t, condate.to)
   then
    substs ← substs ∪ {global_store};
     fi // match(t, condate.to)
  fi //condate.from.format spec = “entry”
 end //for
 for subst ← substs do
  global_store ← subst;
  list ← [];
  if condate.from.format_spec = “entry”
  then
     foreach node t ∈ CFG.entry.succs do
      list ← [t|list];
     end
  fi //condate.from.format_spec = “entry”
      Do depth first check starting from the element of list
      along the edges of control flow edge;
 end //for
end //proc
```

The basic analysis procedure is shown in Figure 3.

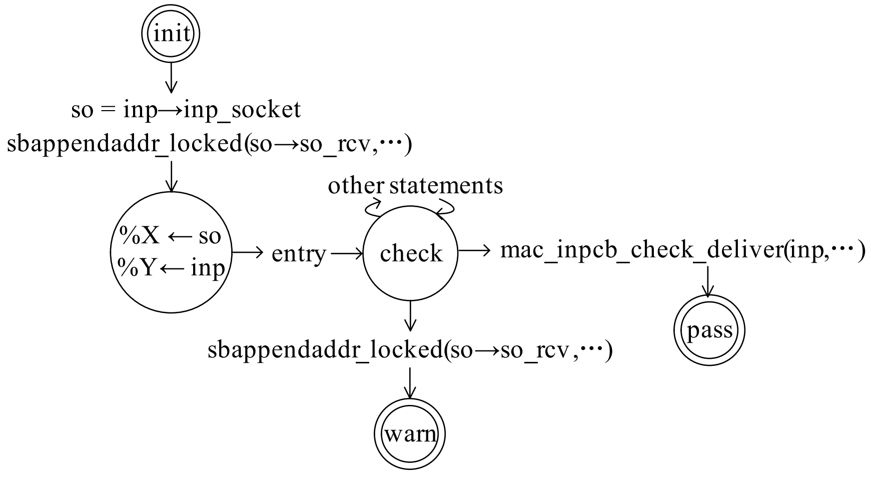

init
so = inp→inp_socket
sbappendaddr_locked(so→so_rcv,⋯)
%X ← so
%Y← inp
entry
other statements
check
mac_inpcb_check_deliver(inp,⋯)
pass
sbappendaddr_locked(so→so_rcv,⋯)
warn

Fig. 3. Basic analysis procedure

(2) Matching the type of the placeholder

The original Mygcc doesn't match the type of the placeholder. However, in the MAC framework, static analysis needs to match the types. For example, in Figure 2, as to the function *uma_zfree*, identifying the controlled operation depends on the second parameter. Namely, if the type of the second parameter is a structure pointer of type *inpcb*, it matches the controlled operations related to *inpcb*. The algorithm is as follows:

```
proc match(t,pattern)
  arglist ← ∅
  parmlist ← ∅
  foreach arg ∈ t.args do
    arglist ← [arg|arglist]
  end //for
  foreach parm ∈ pattern.parms do
    arglist = [arg|rest]
    arglist ← rest
    if parm.type = arg.type
    then global_store ← [global_store|parm←arg]
    else goto fail;
    fi
  end //for
  goto out;
 fail:
   global_store ← ∅;
  out:
end //proc
```

(3) Relating one placeholder to another

In the MAC framework, the parameters of some access authorization are not the controlled objects themselves. Therefore, in order to match access authorization contexts and controlled objects, we need to create the relationship between them. The following is the depth-first checking algorithm based on placeholders matching.

```
proc check_match(CFG, condate)
  for subst ← substs do
    global_store ← subst
    match_store ← global_to_match[subst]
    list ← construct the depth first check starting node list
    while list = [t|rest] do
      list ← rest
      if ¬ visited(t)
      then
        visited(t) ← true
        if matched(t,condate.to, global_store)
        then warning "reached t"
        else foreach edge e = t← t'  do
            if ¬ matched(e,avoid, match_store)
            then list ←  [t'|list]
            fi
          end//for
        fi // matched(t,condate.to, global_store)
      fi // ¬ visited(t)
    end //while
  end //for
end //proc
```

Besides the extensions described above, we also accomplish some other extensions to Mygcc to satisfy the static analysis requirements of the MAC framework, and we have issued them as a patch of Mygcc-1.0.0[1].

4 Static Analysis of Memory Leaks

In addition to extending the Mygcc, another important work is to write checking rules of the static analysis. First, we describe some typical checking rules of the static analysis of the MAC framework. Then, we give the analysis results.

4.1 Rules of Static Analysis

To enforce static analysis of the MAC framework, we write 32 checking rules based on the extended Mygcc *condate*, and they cover the major modules of the FreeBSD kernel. Here, we give two examples of the *condate* rules.

[1] http://wiki.freebsd.org/ZhouyiZHOU?action=AttachFile&do=get&target= mygcc.patch

Figure 4 gives the checking rule of the *inpcb* destruction. Line 1 is the name of the rule, Line 2 matches the type of the placeholder and Line 3 indicates the depth-first traverse starting from the function entry. Line 4 gives conditions of warnings and indicates to execute function *uma_zfree* for the controlled destruction operation. Line 5 gives the conditions that the rule doesn't throw warnings as follows:

• In the execution path, the controlled operation *uma_zfree* is ahead of the destruction hook *mac_inpcb_destroy*.

• The initialization hook of the MAC framework fails. Because the security label hasn't been allocated, it needs not to be destroyed.

```
1  condate inpcb_destory {
2  types "%X struct_inpcb *"
3  from "entry"
4  to "uma_zfree(%_,%X)"
5  avoid "mac_inpcb_destroy(%X)" or
6  +"%w = mac_inpcb_init (%X,%_);%w != 0"
7  } warning("mac_inpcb_destory needed");
```

Fig. 4. Example of destruction hook checking rules

Figure 5 gives the checking rule of the security label initialization. After the controlled operation *m_gethdr* or *m_getcl* creates an *mbuf* (Line 3 and 4), on the execution path there is a security label initialization hook *mac_inpcb_create_mbuf* (Line 7) ahead of a controlled operation on this *mbuf* (Line 5 and 6).

```
1  condate mbuf_create {
2  types "%X struct mbuf *"
3  from "%X = m_gethdr(%_,%_)" or
4  "%X = m_getcl(%_,%_,%_)"
5  to "ip_output(%X,%_,%_,%_,%_,%_)" or
6  "ip6_output(%X,%_,%_,%_,%_,%_)"
7  avoid "mac_inpcb_create_mbuf(%_,%X)"
8  }warning("uninitialized mbuf send");
```

Fig. 5. Example of initialization hook checking rules

4.2 Result of Static Analysis

We have analyzed 320 C files based on the standard kernel configuration, and the tool throws 27 warnings. After these warnings are evaluated manually and discussed in the TrustedBSD mail list, 8 of them are confirmed by the FreeBSD community and the other 19 are false positive. 5 of the confirmed warnings are related to *NFS* and *FIFOFS*, and they may be used to guide the improvement of these subsystems. 3 of the confirmed warnings are related to *IPv4* and *IPv6* network protocol stack, and they have been corrected in the FreeBSD CVS.

Our experiment shows the false positive rate of the verification is 70.4%, which partially results from the imperfect of the checking rules and the algorithms. However, compared to other tools (e.g. in [10] it is 95%), the false positive rate is relatively low.

5 Related Work

[4] proposes a static analysis approach, which leads off the researches on the static analysis of hooks placement. At first, they use the modified GCC to output the locations of controlled operations. Then they use a perl script to annotate the kernel source code with the type information needed by CQUAL. At last, they use CQUAL to perform inter-procedural analysis. Because CQUAL is flow-insensitive, they cannot enforce path-sensitive analysis.

[5] presents a run-time verification tool, which is simpler than that in [4], to verify LSM hooks placement. Their approach begins with logging system call entry/exit/arguments, function entry/exits, controlled operations and authorizations. Then they filter the logging results using filtering rules defined by a rule language. Finally they manually examine authorization graph generated from the filtered results. [5] is not suitable for complete destruction verification because its coverage rate is low and cannot enforce verification automatically.

[11] puts forward a novel approach to automatically inserting authorization hooks into the Linux kernel. Their approach is not suitable for analyzing the MAC framework because of its path-insensitive nature.

There are also a number of general purpose static analysis tools for C programs [12]. After thorough investigation, we find that Mygcc is more suitable for the analysis of the MAC Framework, because it is open-source, path-sensitive and has parsing power of an integrated compiler.

6 Conclusion

We analyze a class of memory leaks resulting from the security label management in the MAC framework. In order to detect potential memory leaks in the MAC framework, we design and implement a static analysis approach based on the extended Mygcc. This approach can cover all execution paths of a specific controlled object to check whether the destruction hook is properly placed. Our work can help the framework developers to improve the correctness of the hooks placement.

Acknowledgements. We would like to thank Nic Volanschi, the author of mygcc, for his valuable advices on how to extend mygcc. We also thank the anonymous reviewers for their comments.

References

1. `http://www.trustedbsd.org/`
2. Bell, D.E., LaPadula, L.J.: Secure Computer System: Unified Exposition and MULTICS Interpretation. MTR-2997, MITRE Corporation, Bedford, MA (1976)

3. Wright, C., Cowan, C., Smalley, S., Morris, J., Kroah-Hartman, G.: Linux Security Modules: General Security Support for the Linux Kernel. In: Usenix Security Symp., Usenix Assoc, pp. 17–31 (2002)
4. Zhang, X., Edwards, A., Jaeger, T.: Using CQUAL for Static Analysis of Authorization Hook Placement. In: Proceedings of the 11th Usenix Security Symposium, San Francisco, California (August 2002)
5. Edwards, A., Jaeger, T., Zhang, X.: Runtime Verification of Authorization Hook Placement for the Linux Security Modules Framework. In: ACM Conference on Computer and Communications Security (November 2002)
6. Foster, J.S., Fahndrich, M., Aiken, A.: A Theory of Type Qualifiers. In: ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 1999). Atlanta, Georgia (May 1999)
7. Volanschi, N.: A Portable Compiler-Integrated Approach to Permanent Checking. In: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering, Tokyo, Japan (September 2006)
8. Watson, R., Morrison, W., Vance, C., Feldman, B.: The TrustedBSD MAC Framework: Extensible Kernel Access Control for FreeBSD 5.0. In: USENIX Annual Technical Conference, San Antonio, TX (June 2003)
9. Larochelle, D., Evans, D.: Statically Detecting Likely Buffer Overflow Vulnerabilities. In: 10th USENIX Security Symposium (August 2001)
10. Meng, C., He, Y., Luo, Y.: Value Equality Analysis in C Program API Conformance Validation. Journal of Software 19(10), 2550–2561 (2008) (in Chinese)
11. Ganapathy, V., Jaeger, T., Jha, S.: Automatic Placement of Authorization Hooks in the Linux Security Modules Framework. In: Proceedings of the 12th ACM conference on Computer and communications security (November 2005)
12. `http://spinroot.com/static/`

Efficient Concurrent $n^{poly(\log n)}$-Simulatable Argument of Knowledge[⋆]

Guifang Huang[1,3], Dongdai Lin[1,3], and Yanshuo Zhang[2]

[1] The State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing, 100190, P.R. China
{hgf,ddlin}@iscas.ac.cn

[2] Beijing Institute of Electronic Science and Technology, Beijing, 100070, P.R. China
zhangyanshuo@amss.ac.cn

[3] Graduate University of Chinese Academy of Sciences, P.R. China, 100049

Abstract. In [16], Pass generalized the definition of zero knowledge proof and defined $n^{O(\sigma(n))}$-simulatable proof which can be simulated by a simulator in $n^{O(\sigma(n))}$ time. Assuming the existence of one-way permutation secure against sub-exponential circuits and 2-round perfect hiding commitment scheme, an efficient 4-round perfect $n^{poly(\log n)}$-simulatable argument of knowledge was presented there.

In this paper, we construct an efficient concurrent $n^{poly(\log n)}$-simulatable argument of knowledge under more general assumption. The new scheme is 5-round and is based on the existence of one-way permutation secure against sub-exponential circuits. However, for the scheme in [16], if using ordinary Σ-protocol for the corresponding statement as sub-protocol, instead of Σ-protocol with honest verifier perfect zero knowledge, the resulting protocol is not necessarily closed under concurrent composition.

Keywords: straight-line $n^{poly(\log n)}$-simulatable, argument of knowledge, Σ-protocol.

1 Introduction

Zero knowledge (short for ZK) proof, brought out by Goldwasser, Micali and Rackoff [11], is a protocol between two parties: prover and verifier, in which prover can convince verifier the validity of a statement but reveals nothing. Since its invention, ZK proof has attracted much attention and a great deal of work has been done on it. One of the most fundamental results achieved is that every language in $\mathcal{NP}$ has a ZK proof [12]. Nowadays, ZK proof has played a central role in study of cryptographic protocols. For example, it can be used in multi-party computation [8,9] to have the parties prove the correctness of their computations.

The definition of ZK proof is formalized through simulation paradigm. That is, for every probabilistic polynomial time verifier, there exists a probabilistic

⋆ This work is supported by NSFC under grant No. 60673069, 60803128, and the National 863 Program under grant No. 2007AA01Z447.

polynomial time simulator such that what the simulator outputs is indistinguishable from the real interaction. Although ZK proof is very useful in the design of increasingly complex cryptographic tasks, many limitations are known [10,13,14]. As a result, general protocols, whose security requirements are formalized by simulation paradigm of ZK proof, inherit these limitations. For example, recently several limitations have been shown concerning the concurrent composition [6] of secure two-party and multi-party protocols [14,15]. In order to overcome some of these limitations, some relaxed notions which are good enough for applications are brought out. A first step in this direction is the definition of witness indistinguishability [7]. Recently, Pass [16] defined another relaxed notion—$n^{O(\sigma(n))}$-simulatable proof (argument). It was pointed out that $n^{O(\sigma(n))}$-simulatable argument can be used as a sub-protocol to construct universal composable secure protocols.

$n^{O(\sigma(n))}$-simulatable argument is that for every probabilistic polynomial time verifier, there exists a simulator with running time $n^{O(\sigma(n))}$ which can simulate the real conversation. From definition of ZK argument, it is concluded that ZK argument is $n^{O(1)}$-simulatable argument. Therefore, $n^{O(\sigma(n))}$-simulatable argument is a generalization of definition of ZK argument. Furthermore, $n^{O(\sigma(n))}$-simulatable argument is witness indistinguishable. Therefore, $n^{O(\sigma(n))}$-simulatable argument is a notion between ZK and witness indistinguishability.

In [16], based on the assumption that there exist one-way permutation secure against sub-exponential circuits and 2-round perfect hiding commitment scheme, an efficient 4-round concurrent perfect $n^{poly(\log n)}$-simulatable argument of knowledge was constructed. Honest verifier perfect ZK property of Σ-protocol contributes to the "perfect" property of the construction, which in turn results in its closeness under concurrent composition. However, if using ordinary Σ-protocol instead of Σ-protocol with honest verifier perfect ZK property as a tool, the resulting scheme is not necessarily closed under concurrent composition.

In this paper, using ordinary Σ-protocol as a tool, we give an efficient concurrent $n^{poly(\log n)}$-simulatable argument of knowledge under more general assumption. The scheme is 5-round and is based on the existence of one-way permutation secure against sub-exponential circuits.

It is organized as follows. In section 2, some related notions and definitions are given. In section 3, we present the 5-round concurrent $n^{poly(\log n)}$-simulatable argument of knowledge and gives the corresponding proofs.

2 Preliminaries

A function $f(n)$ is called negligible, if for every positive polynomial $q(n)$, there exists N such that for all $n \geq N$, we have $f(n) \leq 1/q(n)$.

Definition 1. *(one-way function secure against sub-exponential circuits [16])* $f : \{0,1\}^\star \rightarrow \{0,1\}^\star$ *is called one-way function secure against* 2^{n^k} *adversary, if the following conditions hold:*

(1) There exists a polynomial time algorithm A, such that $A(x) = f(x)$;

(2) For any probabilistic algorithm B with running time 2^{n^k}, any positive polynomial $p(\cdot)$ and all sufficiently large n, any auxiliary input $z \in \{0,1\}^{poly(n)}$, we have

$$Pr[B\left(f\left(U_n\right), z\right) \in f^{-1}\left(f(U_n)\right] < 2^{-n^k};. \qquad (1)$$

where U_n is a random variable uniformly distributed in $\{0,1\}^n$.

f is called one-way function secure against sub-exponential circuits iff there exists a constant $0 < k < 1$ such that f is secure against 2^{n^k} adversary.

Definition 2. *(Argument of knowledge [9]) Let R be a binary relation, probabilistic polynomial time machine V is called knowledge verifier for language L with relation R, if the following conditions hold:*

(1) (non-triviality) There exists a probabilistic polynomial time machine P, such that for every $(x, y) \in R$ all possible interaction of P with V on common input x and auxiliary input y is accepting;

(2) (knowledge soundness) There exists a probabilistic polynomial time oracle machine K such that for every polynomial time machine $P^\star$ and every $x, y, r \in \{0,1\}^\star$, machine K satisfies the following condition: Let $p(x, y, r)$ be the probability that $P^\star_{x,y,r}$ convinces V. If $p(x, y, r) > u(|x|)$ where $u(|x|)$ is a negligible function, then machine K can output $s \in R(x)$ with probability at least $1 - u(|x|)$ in polynomial time.

If V is a knowledge verifier for language L with relation R and P is a machine satisfying non-triviality condition, (P, V) is called argument of knowledge for language L. K is called knowledge extractor.

Definition 3. *(straight-line $n^{O(\sigma(n))}$-simulatable argument of knowledge [16]) Let (P, V) be an argument of knowledge for language $L \in \mathcal{NP}$ with relation R. (P, V) is called straight-line $n^{O(\sigma(n))}$-simulatable argument of knowledge, if for every probabilistic polynomial time machine $V^\star$, there exists a simulator S with running time $n^{O(\sigma(n))}$ such that the following two ensembles are computational indistinguishable:*

(a) $\{< P(y), V^\star(z) > (x)\}_{z \in \{0,1\}^\star, x \in L}$, where $y \in R(x)$;
(b) $\{< S, V^\star(z) > (x)\}_{z \in \{0,1\}^\star, x \in L}$.

Definition 4. *(straight-line concurrent $n^{O(\sigma(n))}$)-simulatable argument of knowledge [16]) Let (P, V) be an argument of knowledge for language $L \in \mathcal{NP}$ with relation R. (P, V) is called straight-line concurrent $n^{O(\sigma(n))}$-simulatable argument of knowledge, if for every probabilistic polynomial time oracle machine A that is not allowed to rewind the oracle it has access to, for any positive polynomial $g(n)$, there exists a simulator $S(i, x)$ with running time $n^{O(\sigma(n))}$ such that the following two ensembles are computational indistinguishable:*

(a) $\{A^{P(x_1,y_1),\cdots,P(x_{g(n)},y_{g(n)})}(z, x_1, \cdots, x_{g(n)})\}_{z\in\{0,1\}^\star, x_1,\cdots,x_{g(n)}\in L}$, *where* $y_i \in R(x_i)$ *for* $i = 1, \cdots, g(n)$*;*
(b) $\{A^{S(1,x_1),\cdots,S(g(n),x_{g(n)})}(z, x_1, \cdots, x_{g(n)})\}_{z\in\{0,1\}^\star, x_1,\cdots,x_{g(n)}\in L}$.

Σ-Protocol ([5]): Σ-protocol is a special 3-move witness indistinguishable proof (argument) of knowledge. Let the conversation in the interaction be (a, e, z). Σ-protocol for language $L \in \mathcal{NP}$ with relation R is required to satisfy the following conditions:

(a) (special soundness) For common input x, from any pair of accepting conversations (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$;
(b) (special honest verifier computational zero knowledge) There exists a simulator M such that on input x and the challenge e, M can output an conversation which is computational indistinguishable from the real interaction.

Commitment Schemes: A Commitment scheme is a two-party interactive protocol consisting of two phases. S and R are polynomial time machines. In the first phase, sender S computes a commitment c to value m and sends c to receiver R; In the second phase, S reveals the value m along with the random coins used to R and R checks the validity of the commitment c. A commitment scheme is required to satisfy the following property:

(a) Hiding: two commitments to different values are computational indistinguishable; If the two commitments to different values are identically distributed, the commitment scheme is perfectly hiding;
(b) Binding: Once having sent the commitment c to m, except for negligible probability, S cannot reveal a different value $m' \neq m$ such that c is also a valid commitment to m'.
If for $m' \neq m$, the two sets $Com(m)$ and $Com(m')$ is disjoint, it is called perfectly binding, where $Com(m)$ and $Com(m')$ are sets of all possible commitments to m and m' respectively.

3 Concurrent $n^{poly(\log n)}$-Simulatable Argument of Knowledge

In this section, using ordinary Σ-protocol as a tool, an efficient 5-round $n^{poly(\log n)}$-simulatable argument of knowledge which is closed under concurrent composition is presented.

It was pointed out that $n^{poly(\log n)}$-simulatable argument of knowledge is not necessarily closed under concurrent composition [16]. In the construction of [16], if using ordinary Σ-protocol as a sub-protocol, instead of Σ-protocol with honest verifier perfect ZK property, the reduction used to prove concurrency fails. The reason is that when using hybrid argument to reduce concurrency to stand-alone circumstances, some oracle queries to the simulator are required to be answered by polynomial time algorithm, not by oracles. However, the simulator can only compute these answers in $n^{poly(\log n)}$ time. To solve this problem, the new scheme works in the following way: first, V executes a Σ-protocol to prove statement $s \in L_1 \vee t \in L_2$. Then, P executes another Σ-protocol to prove $x \in L \vee (s,t) \in L_3$. Because witness for $t \in L_2$ is also a witness for $(t,c) \in L_3$, by special soundness of Σ-protocol, we can extract a witness for $t \in L_2$ from $V^\star$. With the witness, the queries can be computed efficiently. Thus, the reduction works.

Let $f : \phi : \{0,1\}^\star \rightarrow \{0,1\}^\star$ be one-way permutation secure against 2^{n^k} adversary, where the constant $0 < k < 1$. Suppose $Com(\cdot)$ be a perfectly binding commitment scheme and G be a pseudo-random generator stretching strings of length n to $2n$. Three Languages are defined respectively as follows: $L_1 = \{s : \exists m, a, s.t. s = Com(m,a))\}$, $L_2 = \{t : \exists r, s.t. t = f(r)\}$, $L_3 = \{(t,c) : \exists r, s.t. t = f(r) \wedge c = G(r)\}$.

Construction of Protocol Π
Common Input: $x \in L$, where the length of x is n.
Auxiliary Input of Prover P: $y \in R(x)$.

(P1): P randomly chooses m, then he computes $s = Com(m)$ and sends s to V;
(V1): V randomly picks $r \in \{0,1\}^{(\log n)^m}$, computes $t = f(r)$ and sends t to P, where $m = 1 + \frac{1}{k}$. At the same time, V uses Σ-protocol to prove statement $s \in L_1 \vee t \in L_2$ and sends the first round message a_1 to P;
(P2): P produces a random challenge e_1 for message a_1 and sends it to V. Also, he randomly picks $c \in \{0,1\}^{2(\log n)^m}$ and uses Σ-protocol to prove statement $x \in L \vee (t,c) \in L_3$. He sends c and the first round message a_2 of Σ-protocol to V;
(V2): V produces a random challenge e_2 for message a_2 and computes message z_1 to answer challenge e_1. He sends e_2, z_1 to P;
(P3): If (a_1, e_1, z_1) is an accepting conversation of statement $s \in L_1 \vee t \in L_2$, P produces message z_2 and sends it to V to answer challenge e_2; Otherwise, P aborts;
(V3): If (a_2, e_2, z_2) is an accepting conversation of statement $x \in L \vee (t,c) \in L_3$, V output 1. Otherwise, he outputs 0.

Lemma 1. *If there exists one-way permutation secure against sub-exponential circuits, then protocol Π is straight-line $n^{poly(\log n)}$-simulatable argument of knowledge.*

Proof. First, *Completeness* of protocol Π comes from completeness of Σ-protocol for statement $x \in L \vee (t,c) \in L_3$. For $x \in L$, P can use $y \in R(x)$ to prove the statement $x \in L \vee (t,c) \in L_3$.

Second, we prove *knowledge soundness* of protocol Π. For any probabilistic polynomial time machine $P^\star$ satisfying that $Pr[< P^\star, V > (x)] = p$ is non-negligible in $|x|$, there exists a probabilistic polynomial time oracle machine K such that K can extract witness for statement $x \in L \vee (t,c) \in L_3$ with probability at least $1 - u(|x|)$, where $u(|x|)$ is a negligible function. For these accepting transcripts produced by $P^\star$, assume that there exists non-negligible fraction of (s,t), such that K can extract witnesses of $(t,c) \in L_3$ with non-negligible probability from the transcripts containing (s,t). For such (s,t), we can construct a malicious verifier $V^{\star\star}$ by incorporating algorithm $P^\star$ such that $V^{\star\star}$ can distinguish the witnesses for statement $s \in L_1 \vee t \in L_2$.

Algorithm $V^{\star\star}$ is constructed as follows: On input (s,t) and auxiliary input x, $V^{\star\star}$ runs algorithm $P^\star$ in the following way:

(1) For statement $s \in L_1 \vee t \in L_2$, $V^{\star\star}$ interacts with prover externally to prove the statement. When receiving messages, $V^{\star\star}$ gives the messages to $P^\star$; When is required to send messages, he reads messages from the communication-tape of $P^\star$ and sends them to prover.
(2) For statement $x \in L \vee (t,c) \in L_3$, $V^{\star\star}$ acts as honest verifier to interact with $P^\star$ internally. Then, $V^{\star\star}$ runs knowledge extractor for statement $x \in L \vee (t,c) \in L_3$ to output a witness w. If w is a witness for $x \in L$, $V^{\star\star}$ outputs 0. If w is a witness for $(t,c) \in L_3$, $V^{\star\star}$ outputs 1.

From the construction of $V^{\star\star}$, when prover uses a witness for $t \in L_2$ to interact with $V^{\star\star}$, the probability that $V^{\star\star}$ outputs 1 is non-negligible. When prover uses a witness for $s \in L_1$ to interact with $V^{\star\star}$, the view of $V^{\star\star}$ is independent of witness of $(t,c) \in L_3$. So in this case the probability that $V^{\star\star}$ outputs 1 is negligible. Hence, $V^{\star\star}$ can distinguish the witnesses for statement $s \in L_1 \vee t \in L_2$, which contradicts with witness indistinguishability. Therefore, except for negligible probability, what K extracts from an accepting conversation is a witness for $x \in L$. That is, knowledge soundness of protocol Π holds.

At last, we prove that Π is *straight-line* $n^{poly(\log n)}$*-simulatable.* For any probabilistic polynomial time verifier $V^\star$, simulator S is constructed as follows: First, he randonly chooses a value m and computes $s = Com(m)$. Then, on receiving message (t, a_1) from $V^\star$, S performs exhaustive search to find $r \in \{0,1\}^{(\log n)^m}$ such that $f(r) = t$. Then, S uses r to compute $c = G(r)$ and takes r as a witness for statement $x \in L \vee (t,c) \in L_3$ to interact with $V^\star$. The running time of S is $poly(2^{(\log n)^m}) = poly(n^{(\log n)^{\frac{1}{k}}})$. By witness indistinguishability of Σ-protocol, the distribution produced by simulator is computational indistinguishable from the real interaction of P and $V^\star$.

From above, we can conclude that protocol Π is straight-line $n^{poly(\log n)}$-simulatable argument of knowledge. □

Lemma 2. *In protocol Π, for any probabilistic polynomial time $V^\star$, if he can convince P the statement $s \in L_1 \vee t \in L_2$ with non-negligible probability, then except for negligible probability, what the extractor K_1 extracts from $V^\star$ is a witness for $t \in L_2$.*

Proof. Assume that there exists a probabilistic polynomial time $V^\star$ who can convince P the statement $s \in L_1 \vee t \in L_2$ with non-negligible probability, the probability that K_1 outputs a witness for $s \in L_1$ is non-negligible. we can construct a polynomial time algorithm B to break the hiding property of commitment scheme.

Algorithm B can be constructed as follows: On input s_b, m_0, m_1, where s_b is a commitment to m_0 or m_1, B runs the knowledge extractor K_1 for $s_b \in L_1 \vee t \in$

L_2. When K_1 fails or outputs a witness for $t \in L_2$, B aborts. Otherwise, suppose K_1 output m. If $m = m_0$, B outputs 0, else B outputs 1.

Hence, B can distinguish the commitment of m_0 and m_1 with non-negligible probability, which contradicts with the hiding property of commitment scheme. Therefore, except for negligible probability, what K_1 extracts is a witness for $t \in L_2$. □

Lemma 3. *If there exists one-way permutation secure against sub-exponential circuits, then protocol Π is straight-line concurrent $n^{poly(\log n)}$-simulatable.*

Proof. For any probabilistic polynomial time oracle machine A that is not allowed to rewind the oracle it has access to, for any positive polynomial $g(n)$, let $S(i, x) = S(x)$ where $S(x)$ is the simulator constructed in Lemma 1. It is sufficient to prove the following two ensembles are computational indistinguishable:

(a) $\{A^{P(x_1,y_1),\cdots,P(x_{g(n)},y_{g(n)})}(z, x_1, \cdots, x_{g(n)})\}_{z\in\{0,1\}^\star,x_1,\cdots,x_{g(n)}\in L}$, where $y_i \in R(x_i)$ for $i = 1, \cdots, g(n)$;
(b) $\{A^{S(1,x_1),\cdots,S(g(n),x_{g(n)})}(z, x_1, \cdots, x_{g(n)})\}_{z\in\{0,1\}^\star,x_1,\cdots,x_{g(n)}\in L}$.

We use hybrid argument to prove it. Assume that there exists an polynomial time oracle machine A and $z \in \{0, 1\}^\star$, such that for infinitely many n, the above two ensembles are distinguishable by a probabilistic polynomial time distinguisher D. The distinguishing gap is $\frac{1}{q(n)}$, where $q(n)$ is a positive polynomial. For $i = 0, 1, 2, \cdots, g$, experiment H_i is defined in the following way: For the first i sessions, $P(x_i, y_i)$ acts as the oracle to answer queries of A. For the subsequent sessions, $S(j, x)$ is taken as the oracle to return answers to A's queries, where $j = i + 1, \cdots, g$. It is seen that experiment H_0 corresponds to ensemble (b) and experiment H_g corresponds to ensemble (a). Then, there must exist $i \in \{0, 1, \cdots, g\}$ such that D can distinguish H_i and H_{i+1} with distinguishing gap $\frac{1}{g(n)q(n)}$.

Using algorithm A, we can construct an probabilistic polynomial time oracle machine A_1: On input x_{i+1} and auxiliary input $z' = z \circ x_1 \circ \cdots \circ x_i \circ x_{i+2} \circ \cdots \circ x_{g(n)} \circ y_1 \circ \cdots \circ y_i \circ y_{i+2} \circ \cdots \circ y_{g(n)}$, A_1 runs algorithm $A(x_1, \cdots, x_{g(n)}, z)$ in the following way:

(1) For $j = 1, 2, \cdots, i$, when A queries for the j-th session, A_1 runs algorithm $P(x_j, y_j)$ to provide answers to A;
(2) For $j = i + 2, \cdots, g$, when A queries for the j-th session, A_1 invokes the extractor K_1 for statement $s_j \in L_1 \vee t_j \in L_2$ to get a witness r_j for statement $t_j \in L_2$ (by lemma 2). Then, A_1 uses r_j as a witness to compute messages to answer A;
(3) For the i + 1-th session, A_1 provides answers to A by access to oracle $P(x_{i+1}, y_{i+1})$ or $S(i + 1, x_{i+1})$. At last, A_1 outputs what A outputs.

From the construction of algorithm A_1, it is seen that D can distinguish $A_1^{P(x_{i+1},y_{i+1})}$ and $A_1^{S(i+1,x_{i+1})}$, which contradicts with the fact that Π is straight-line $n^{poly(\log n)}$-simulatable. Therefore, ensembles (a) and (b) are computational indistinguishable. That is, protocol Π is straight-line concurrent $n^{poly(\log n)}$-simulatable. □

From above three lemmas, we conclude that Π is straight-line concurrent $n^{poly(\log n)}$-simulatable arguments of knowledge. Therefore, we have

Theorem 1. *If there exists one-way permutation secure against sub-exponential circuits, then every language $L \in \mathcal{NP}$ has an efficient 5-round straight-line concurrent $n^{poly(\log n)}$-simulatable argument of knowledge.*

4 Conclusion

$n^{O(\sigma(n))}$-simulatable argument of knowledge is a generalization of definition of ZK argument of knowledge and is very useful in constructing multi-party secure computation schemes and universal composable secure protocols. Under the assumption that there exists one-way permutation secure against sub-exponential circuits, using ordinary Σ-protocol as a tool, we construct an efficient 5-round $n^{poly(\log n)}$-simulatable argument of knowledge. The new scheme is closed under concurrent composition.

Acknowledgments. We thank anonymous referees for the helpful suggestions to improve this paper.

References

1. Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. of Computer and System Sciences 37(2), 156–189 (1988)
2. Barak, B., Pass, R.: On the Possibility of One-Message Weak Zero-Knowledge. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 121–132. Springer, Heidelberg (2004)
3. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
4. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-Box Concurrent Zero- Knowledge Requires (almost) Logarithm Many Rounds. SIAM J. on Computing 32(1), 1–47 (2002)
5. On Sigma Protocols, `http://www.daimi.au.dk/~ivan/CPT.html`
6. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: 30th Annual ACM Symposium on Theory of Computing, pp. 409–418. ACM Press, New York (1998)
7. Feige, U., Shamir, A.: Witness Indinstinguishable and Witness Hiding Protocols. In: 22th Annual ACM Symposium on Theory of Computing, pp. 416–426. ACM Press, New York (1990)
8. Secure Multi-Party Computation, `http://www.wisdom.weizmann.ac.il`
9. Goldreich, O.: Foundation of Cryptography-Basic Tools. Cambridge University Press, Cambridge (2001)
10. Goldreich, O., Krawczyk, H.: On the Composition of Zero-Knowledge Proof Systems. SIAM J. on Computing. 25(1), 169–192 (1996)
11. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof System. SIAM J. on Computing. 18(1), 186–208 (1989)

12. Goldreich, O., Micali, S., Widerson, A.: Proofs that Yields Nothing But Their validity or ALL Languages in $\mathcal{NP}$ Have Zero Knowledge Proof Systems. J. of ACM. 38(3), 691–729 (1991)
13. Goldreich, O., Oren, Y.: Definitions and Properties of Zero-Knowledge Proof Systems. J. of Cryptology. 7(1), 1–32 (1994)
14. Lindell, Y.: General Composition and Universal Composability in Secure Multi-Party Computation. In: 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 394–403. IEEE Computer Society, Washington (2003)
15. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
16. Pass, R.: Simulation in Quasi-Polynomial Time and Its Application to Protocol Composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)

New Constructions for Reusable, Non-erasure and Universally Composable Commitments

Huafei Zhu

C&S Department, I^2R, Singapore
huafei@i2r.a-star.edu.sg

Abstract. This paper proposes a novel construction of reusable and non-erasure commitment schemes in the common reference string model. We show that our implementation is secure in the universally composable paradigm assuming that the decisional Diffie-Hellman problem over a squared composite modulus of the form $N = pq$ is hard. Our methodology relies on state-of-the-art double trap-door public-key encryption protocols so that a simulator in charge of a common reference string can extract messages of cipher-text rather than the equivocability of underlying cryptographic systems. As a result, our method differs from those presented in [2] and [7]. The double trap-door mechanism is of great benefit to an ideal-world simulator since no modifications will be charged to unopened commitments in case that the participants who generated these commitments are corrupted, and thus enables us to implement efficient simulators in the ideal-world.

Keywords: Non-erasure, reusability, simulator, universally composable commitment.

1 Introduction

Universally composable (UC) commitments guarantee that commitment protocols behave like an ideal commitment service, even when concurrently composed with an arbitrary set of protocols. A commitment protocol with UC-security implies that it is non-malleable not only respect to other copies of the same protocol but even with respect to other protocols. To prove security of a commitment scheme in the UC model, one must construct an ideal-world adversary such that an adversary's view of a real-life execution of a commitment protocol can be simulated given just the data the adversary is entitled to.

Considering a scenario where a commitment protocol execution between a corrupted committer P_i and an honest receiver P_j is performed, and a dummy adversary is assumed to merely send messages that are generated by an environment $\mathcal{Z}$, and relay to the messages sent to P_i. Assume that the environment $\mathcal{Z}$ secretly picks a random bit $b \in \{0,1\}$ at the beginning of the commitment protocol execution and generates the messages for P_i by running the protocol of the honest committer for b and P_j's answers. To simulate this adversarial behavior, an ideal world simulator $\mathcal{S}$ must first extract input bit b from the messages generated by $\mathcal{Z}$ on the tape of the ideal-world uncorrupted party $\widetilde{P}_i$ and then copies this extracted bit b to the functionality $\mathcal{F}_{\mathtt{com}}$. If the commitment scheme allows the simulator to successfully extract the committed bit, then the commitment is not secure since the successful simulator must come up with the true bit b at the commitment step which contradicts the security of a commitment protocol.

To simulate the adversarial behavior above, Canetti and Fischlin [2] have proposed two methods sketched below for constructing an ideal world simulator: 1) `one-time-usable common reference string model`: a fresh common reference string σ must be generated for each committed bit in this model. The common reference string σ is constructed from Crescenzo, Ishai and Ostrovsky's protocol with a specified trap-door pseudo-random generator which guarantees the required equivocability and thus allows a simulator to adapt committed values of unopened commitment in case that the participants who generated these commitments are corrupted. We remark that the computation complexity of their basic construction is heavy since the commitment scheme uses one-time common reference string, and each generation of a common reference string costs at least $3n$ computations of the underlying one-way permutation, where n is a security parameter for the underlying pseudo-random generator stretching n-bit inputs to $4n$-bit outputs.
2) `reusable common reference string model`: a common reference string in this model is constructed from claw-free trap-door permutation pairs and a public key encryption scheme to guarantee that a simulator can produce a correct cipher-text and a fake random string suggesting that the cipher-text has been obtained by the oblivious sampling procedure. This common reference string can be reused for committing multi-bit. To prove the security of the commitment schemes in this model, Canetti and Fischlin first assume that a committer generates a cipher-text of a committer's identity and then erases the randomness used for the encryption of $(0^n, P_i)$ before the commitment message is sent. Assuming the existence of obliviously sample encryption schemes, Canetti and Fischlin further proposed an improved construction such that the unpleasant assumption of erasure of randomness used for the encryption is eliminated.

The security proof of commitment schemes in the one-time-usable common reference string model relies on the equivocable bit commitment of Crescenzo, Ishai and Ostrovsky [8] which in turn is a modification of Naor's commitment scheme [10]. The security proof of commitment schemes in the reusable common reference string model relies on oblivious sample property of the underlying encryption scheme (constructed from Cramer and Shoup's encryption scheme [5]) such that one cipher-text is open while the remaining cipher-text is never opened.

1.1 Recent Work

At Crypto 2008, Dodis, Shoup and Walfish [7] proposed an efficient constructions of composable commitments based on Ω protocols. An Ω-protocol first introduced and formalized by Garay, MacKenzie and Yang [9] in the context of zero-knowledge proof system, is similar with the notion of a Σ-protocol, with an extra property that one can generate a public parameter σ of a system together with a trap-door information τ, such that the knowledge of τ allows one to extract the witness from any valid conversation between a prover and sender. They use the same technique of Canetti and Fischlin [2] in the sense that the proof of security relies on the equivocability of the underlying cryptographic systems (e.g., Crescenzo, Ishai and Ostrovsky's bit commitment scheme, Canetti and Fischlin's equivocable encryption scheme, and Garay, MacKenzie and Yang's Ω-protocol) such that a simulator can extract inputs of corrupted parties.

1.2 This Work

In this paper, we propose a novel construction of reusable, non-erasure commitment 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.

`The technique`: The novelty of this paper is that an ideal-world simulator does not rely on the equivocability of underlying cryptographic systems but on the existence of double trapdoor public-key encryption schemes, (the existence of semantically secure double trap-door encryption scheme is not a problem, see Section 3 for more details). That is, a simulator in our model is allowed to generate a public-key/secret key pair (pk_S, sk_S) while a simulated participant P_i is allowed to generate its own public-key (pk_i, sk_i) . The double trap-door mechanism ensures that any cipher-text generated by the encryption scheme $E_{pk_i}(m, r_m)$ can be correctly decrypted with the help of trap-door string sk_S, where $m \in \mathcal{M}_{pk_i}$ and $\mathcal{M}_{pk_i}$ is the message space specified by the public-key pk_i. Notice that if Bresson, Catalano and Pointcheval's public-key encryption scheme is applied, then $\mathcal{M}_{pk_i} = \mathcal{M}_{pk_S}$ (throughout the paper, we simply abbreviate the notation $\mathcal{M}_{pk_i}$ as $\mathcal{M}$). Furthermore, we are able to show that the proposed commitment scheme enjoys the following nice features:

1) `Simulatability`: In our model, a public key pk_S is a common reference string that will be used for constructing universally composable commitments. To simulate the scenario where the environment $\mathcal{Z}$ secretly picks a random element $m \in \mathcal{M}$ at the beginning of the protocol execution and generates the messages for P_i by running the protocol of the honest committer for m and P_j's answers, we simply allow a simulator $\mathcal{S}$ to decrypt the delivered cipher-text $E_{pk_i}(m, r_m)$ and obtain the exact value $m \in \mathcal{M}$; The simulator $\mathcal{S}$ then instructs $\widetilde{P}_i$ to forward the extracted string $m \in \mathcal{M}$ the functionality. Consequently, the simulator $\mathcal{S}$ works if it holds the auxiliary string sk_S.

`Non-erasure`: A protocol is called non-erasure if all random strings used for generating cipher-text by a corrupted participant is revealed to the adversary. It will be clear that any participant in our model is forced to provide its randomness to the adversary once it is corrupted (see Section 4 for more details).

`Reusability`: Any public-key and secret-key pair (pk_i, sk_i) (including the common reference string pk_S) of a double trap-door public-key encryption scheme generated by a real-world participant P_i is reusable in our model (see Section 4 for more details).

1.3 Why Our Methodology Is Interesting?

It is important to evident why our methodology is interesting. Two evidences are due to the following observations:

– **efficient constructions of simulators:** consider a scenario where a simulated real-world adversary $\mathcal{A}$ demands to corrupt a real-world party P_i. To simulate this situation, an ideal-world simulator $\mathcal{S}$ must corrupt the corresponding party $\widetilde{P}_i$ (a dummy party of P_i) in the ideal model and learns all internal information of the party. To deal with such an active corruption, the technique presented in [2] and [7] must instruct $\mathcal{S}$ to modify all decommitment information about the unopened commitment of this party P_i to match the received data and hands this modified internal information to $\mathcal{A}$. A simulator $\mathcal{S}$ in our model, however needs not to adapt possible decommitment information about a previous given but yet unopened commitment of this party since the trap-door information sk_p allows to $\mathcal{S}$ to decrypt a valid cipher-text generated by P_i. As a result, the double-trap-door public-key encryption based construction allows us to implement an efficient ideal-world simulator $\mathcal{S}$ corresponding to the real-world adversary $\mathcal{A}$.

– **concise of common reference string:** a common reference string used in our commitment scheme is a public key pk_S of a simulator ($pk_S = N$, if Bresson, Catalano and Pointcheval's public-key encryption scheme is applied). The common reference strings used in [2] consists of two parts: a string describing two claw-free permutations and a string describing a public key encryption scheme; The common reference string used in [7] consists of two parts: a string describing an identity of a participant and a string describing Ω-protocol. As a result, the common reference string used in this paper is much more concise than that used in [2] and [7].

Road map: The rest of this paper is organized as follows: In Section 2, preliminaries including the notions of UC security and notions of functionalities are briefly sketched; In Section 3, a double trap-door encryption scheme due to Bresson, Catalano and Pointcheval is sketched. The proposed commitment protocol is described and analyzed in Section 4. We conclude our work in Section 5.

2 Preliminaries

We work in the standard universally composable framework, where all participants are modeled as probabilistic polynomial time (PPT) Turing machines. Security of protocols is defined by comparing the protocol execution to an ideal process for carrying out the desired task. Namely, the process of executing a protocol in the presence of an adversary and in a given computational environment is first formalized. Next an ideal processing for carrying out the task at hand is formalized. In the ideal processing the parties do not communicate with each other; instead they have access to an ideal functionality which is essentially an incorruptible trust party that is programmed to capture the desired requirements from the task at hand. A protocol is said to securely realize a task if the processing of running the protocol emulates the ideal process of that task. We assume the reader is familiar with the standard notion of UC security. The detailed descriptions of the executions, and definitions of $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ and $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ are omitted and refer to the reader [3] for more details.

2.1 The Common Reference String Model

The functionality of common reference strings (`crs`) model described in Fig.1. assumes that all participants have access to a common string that is drawn from some specified distribution $\mathcal{D}$.

Functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$

$\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$ proceeds as follows, when parameterized by a distribution $\mathcal{D}$.

– 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, sid, P_i, P_j) to the adversary, where sid is a session identity. 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.

Fig. 1. Functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$ (due to [2])

The common reference string model $\mathcal{F}_{\mathsf{crs}}$ produces a string with a distribution that can be sampled by a PPT algorithm $\mathcal{D}$.

2.2 The Commitment Functionalities

The ideal functionality of a commitment scheme is described in Fig.2.

To capture the notion of reusability, we must define the functionality of multi commitment, de-commitment processes (see Fig.3. for more details).

Definition 1. *Let $\mathcal{F}_{\mathsf{com}}$ be a functionality. A protocol π is said to universally composable realize $\mathcal{F}_{\mathsf{com}}$ if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, the ensemble $IDEAL_{\mathcal{F}_{\mathsf{com}},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable with the ensemble $REAL_{\pi,\mathcal{A},\mathcal{Z}}$.*

Definition 2. *If the protocol securely realizes the functionality $\mathcal{F}_{\mathsf{mcom}}$, then it is called a reusable common reference string and universally composable commitment protocol.*

Functionality $\mathcal{F}_{\mathsf{com}}$

$\mathcal{F}_{\mathsf{com}}$ proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$

- Upon receiving a value (commit, sid, P_i, P_j, $m \in \mathcal{M}$), record the value m, send the message ($\mathsf{receipt}$, sid, P_i, P_j) to P_j and $\mathcal{S}$, and ignore any sequent commit messages, where sid is a session identity.
- Upon receiving a value (open, sid, P_i, P_j) from P_j, proceed the following computations: if some value m was previously recorded, then send the message (open, sid, P_i, P_j, m) to P_j and $\mathcal{S}$ and halt; otherwise halt

Fig. 2. Functionality $\mathcal{F}_{\mathsf{com}}$ (due to [2])

Functionality $\mathcal{F}_{\mathsf{mcom}}$

$\mathcal{F}_{\mathsf{mcom}}$ proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$

- Upon receiving a value (commit, sid, cid, P_i, P_j, $m \in \mathcal{M}$, record the tuple (cid, P_i, P_j, m), send the message ($\mathsf{receipt}$, sid, cid, P_i, P_j) to P_j and $\mathcal{S}$, and ignore any subsequent (commit, sid, cid, P_i, P_j, $\ldots$) messages, where sid is a session identity, and cid is a commitment identity.
- Upon receiving a value (open, sid, cid, P_i, P_j) from P_j, proceed the following computations: if the tuple (sid, cid, P_i, P_j, $m \in \mathcal{M}$) was previously recorded, then send the message (open, sid, cid, P_i, P_j, $m \in \mathcal{M}$) to P_j and $\mathcal{S}$ and halt; otherwise halt

Fig. 3. Functionality $\mathcal{F}_{\mathsf{mcom}}$ (due to [2])

3 Building Blocks

Paillier's encryption scheme and its variations will be used as building blocks for implementing reusable, non-erasure and universally composable commitment schemes.

3.1 Paillier's Encryption Scheme

Paillier investigated a novel computational problem called the composite residuosity class problem (CRS), and its applications to public key cryptography in [11].

Decisional composite residuosity class problems: Let $N = pq$, where p and q are two large safe prime numbers. A number z is said to be a N-th residue modulo N^2, if there exists a number $y \in Z^*_{N^2}$ such that $z = y^N \bmod N^2$. The decisional composite residuosity class problem states the following thing: given

$z \in_r Z^*_{N^2}$ deciding whether z is N-th residue or non N-th residue. The decisional composite residuosity class assumption means that there exists no polynomial time distinguisher for N-th residues modulo N^2.

Paillier's encryption scheme: The public key is a $2k$-bit RSA modulus $N{=}pq$, where p, q are two large safe primes with length k and the secret key is (p, q). The plain-text space is Z_N and the cipher-text space is $Z^*_{N^2}$. To encrypt a message $m \in Z_N$, one chooses $r \in Z^*_N$ uniformly at random and computes the cipher-text as $E_{PK}(m, r) = g^m r^N \bmod N^2$, where $g = (1 + N)$ has order N in $Z^*_{N^2}$. The private key is (p, q). It is straightforward to verify that given $c = (1 + N)^m r^N \bmod N^2$, and the trapdoor information (p, q), one can first compute $c_1{=}c \bmod N$, and then compute r from the equation $r{=}c_1^{N^{-1} \bmod \phi(N)} \bmod N$; Finally, one can compute m from the equation $cr^{-N} \bmod N^2 = 1 + mN$. The encryption function is homomorphic, i.e., $E_{PK}(m_1, r_1) \times E_{PK}(m_2, r_2) \bmod N^2 = E_{PK}(m_1 + m_2 \bmod N, r_1 \times r_2 \bmod N)$. Paillier's scheme is semantically secure if the decisional composite residuosity class problem is hard.

3.2 Bresson-Catalano-Pointcheval Cryptosystem

At Asiacrypt'03, Bresson, Catalano and Pointcheval [1] presented a double encryption scheme based on the hardness assumption that inverting $RSA[N, N]$ (i.e., RSA with public exponent set to N, a variant of Paillier's encryption scheme [11]) which is sketched below:

- Key generation algorithm KG: On input a security parameter n, KG produces composite modulus of the form $N = pq$ that is a product of two safe primes p and q, and g that is an element of order $\lambda(N)$ in $Z^*_{N^2}$, where $\lambda(\cdot)$ is Carmichael function. KG randomly chooses $z \in [0, N^2/2]$ and sets $h = g^z \bmod N^2$ (we remark that g can be an element of order $p'q'$ as well, see [4] for more details). The outputs KG is a pair of public/secret keys (pk, sk), where $pk = (N, g, h)$, $sk{=}(z, p, q)$.
- Encryption algorithm E: to encrypt a message $m \in Z_N$, one chooses a random value $r \in [0, N/4]$, and computes the cipher-text (u, v) where $u = g^r \bmod N^2$ and $v = h^r(1 + N)^m \bmod N^2$.
- Decryption algorithm D: to decrypt a cipher-text (u, v), two methods are possible: 1) computing $(1 + mN)$ as v/u^z with the help of the trapdoor string z; 2) computing $v^{\lambda(N)} = 1 + m\lambda(N)N$ with the help of the trap-door string (p, q).

Assuming that the decisional Diffie-Hellman problem is hard over a squared composite modulus of the form $N = pq$, Bresson-Catalano-Pointcheval cryptosystem is semantically secure (see [1] for more details).

4 The Protocols

In this section, we propose an implementation of reusable, non-erasure and universally composable commitment schemes described in Fig.4. The idea behind of our construction is that an ideal-world simulator is allowed to generate a public-key/secret key pair (pk_S, sk_S) while a simulated participant P_i is allowed to generate its own public-key (pk_i, sk_i) such that any valid cipher-text $E_{pk_i}(m, r_m)$ generated by the underlying public-key encryption scheme can be correctly decrypted with the help of trap-door string sk_S. To prevent a corrupted committer from copying a commitment generated by an uncorrupted party, a session identity sid and a commitment identity cid must be included in the encryption scheme.

The description of protocol π

Common reference string generator: On input a security parameter k, a common reference string generator $\mathcal{G}_{\mathsf{crs}}$ produces a composite modulus of the form $N = pq$ that is a product of two safe primes p and q, where $|p| = q = k$, $p = 2p' + 1$, $q = 2q' + 1$. The common reference string σ is N.

Public key/secret key generation algorithm: Let $L \subseteq Z^*_{N^2}$ be a cyclic of order $2NN'$, and G be a subgroup of Nth powers of L, where $N{=}pq$, $N' = p'q'$, $p = 2p' + 1$ and $q = 2q' + 1$. It is clear that $G \subseteq L$ is a cyclic group of order $2N'$. P_i chooses $z_i \in Z^*_{N^2}$ uniformly at random, and computes $g_i = -z_i^{4N} \bmod N^2$. As a result, $< g_i >$ is a subgroup of G such that $| < g_i > | = N'$ with overwhelming probability [4]. P_i then randomly chooses a secret key $x_i \in [0, N^2/2]$ and sets $h_i = g_i^{x_i} \bmod N^2$. The public key is (N, g_i, h_i) and the secret key is x_i.

Commitment: Commitment for a random string $m \in Z_N$ with a session id sid and a commitment cid:

– computing $c = E_{pk_i}$(sid, cid, P_i, P_j, m, r_m), where $E_{pk_i}(\cdot)$ is Bresson, Catalano and Pointcheval's cryptosystem defined over G_i, $G_i = < g_i >$;
– sending (commit, sid, cid, P_i, P_j, c) to the receiver P_j.

De-commitment De-commitment for (sid, cid, P_i, P_j, c)

– The committer P_i sends (sid, cid, P_i, P_j, m, r_m) to P_j;
– P_j checks the valid of the equation $c = E_{pk_i}$(sid, cid, P_i, P_j, m, r_m). If the check is valid, then output accept, otherwise output reject;

Fig. 4. Reusable, Non-erasure and Universally Composable Commitments

4.1 The Proof of Security

Theorem 1. *The protocol described above is reusable, non-erasure and 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. Given a real-world adversary $\mathcal{A}$, we describe an ideal-world adversary $\mathcal{S}$ corresponding to $\mathcal{A}$ such that for all environments $\mathcal{Z}$, the ensemble

$\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{mcom}},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable with the ensemble $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$. Simulator $\mathcal{S}$ produces a composite modulus of the form $N= pq$ that is a product of two safe primes p and q; Let $pk_S =N$, and $sk_S =(p, q)$, and let N be a common reference string; For the given common reference string N, P_i chooses $z_i \in Z^*_{N^2}$ uniformly at random and sets $g_i =-z_i^{4N} \bmod N^2$. P_i randomly chooses a secret key $x_i \in [0, N^2/2]$ and sets $h_i =g_i^{x_i} \bmod N^2$. The public key is (N, g_i, h_i); the secret key is x_i. We then consider five cases below:

- *Case 1*: at some point in the protocol execution $\mathcal{Z}$ secretly picks a random string $m \in Z_N$ and generates a cipher-text c for a corrupted party P_i by running the protocol of the honest committer for m and P_j's answers, then instructs a dummy adversary $\mathcal{A}$ to inform the corrupted party P_i sending c to P_j; To simulate this case, $\mathcal{S}$ first decrypts c with the help of trap-door information sk_S to obtain the exact value $m \in Z_N$, then instructs $\widetilde{P_i}$ to send the message (commit, sid, cid, P_i, P_j, m) to $\mathcal{F}_{\mathsf{mcom}}$. The commitment functionality $\mathcal{F}_{\mathsf{mcom}}$ sends a reply message ($\mathsf{receipt}$, sid, cid, P_i, P_j) to P_j and $\mathcal{S}$, and ignores any subsequent (commit, sid, cid, P_i, P_j) messages.
- *Case 2*: at some point in the protocol execution $\mathcal{Z}$ instructs an uncorrupted party P_i to send a commitment c of a message $m \in Z_N$ to P_j. To simulate this case, $\mathcal{S}$ first decrypts c with the help of trap-door information sk_S to obtain the exact value $m \in Z_N$, then instructs $\widetilde{P_i}$ sending the message (commit, sid, cid, P_i, P_j, m) to $\mathcal{F}_{\mathsf{mcom}}$. The functionality $\mathcal{F}_{\mathsf{mcom}}$ sends the message ($\mathsf{receipt}$, sid, cid, P_i, P_j) to P_j and $\mathcal{S}$, and ignores any subsequent (commit, sid, cid, P_i, P_j) messages.
- *Case 3*: at some point in the execution $\mathcal{A}$ instructs a corrupted party P_i to send a commitment of a message m to P_j. To simulate this case, $\mathcal{S}$ obtains m and instructs $\widetilde{P_i}$ to send the message (commit, sid, cid, P_i, P_j, m) to $\mathcal{F}_{\mathsf{mcom}}$. The functionality $\mathcal{F}_{\mathsf{mcom}}$ sends the message ($\mathsf{receipt}$, sid, cid, P_i, P_j) to P_j and $\mathcal{S}$ if $m \in Z_N$, otherwise, $\mathcal{F}_{\mathsf{mcom}}$ sends $\perp$ to P_i and $\mathcal{S}$. $\mathcal{F}_{\mathsf{mcom}}$ ignores any subsequent (commit, sid, cid, P_i, P_j) messages.
- *Case 4*: at some point in the protocol execution $\mathcal{A}$ tells a corrupted party P_i to open a valid commitment c correctly with the string $m \in Z_N$. To simulate this case, $\mathcal{S}$ compares m with the previously extracted bit and stops if they differ; otherwise $\mathcal{S}$ sends (open, sid, cid, P_i, P_j) in the name of the party of the functionality $\mathcal{F}_{\mathsf{mcom}}$.
- *Case 5*: Whenever the simulated $\mathcal{A}$ demands to corrupt a party, $\mathcal{S}$ corrupts this party in the ideal model and learns all internal information of the party. Notice that $\mathcal{S}$ needs not to adapt possible decommitment information about a previous given but yet unopened commitment of this party since the trap-door information sk_S allows to S to decrypt a valid cipher-text generated by P_i.

According to the simulator described above, we know that pk_S is the common reference string that serves as a global security parameter of the proposed commitment scheme and sk_S is only used for decryption of a submitted cipher-text c. It follows that all commitments/decommitments whether submitted by corrupted parities or honest parties can be simulated by S; Notice that individual commitment c for a string $m \in Z_N$ is computationally hiding and statistically binding assuming that the underlying Bresson, Catalano and Pointcheval public-key encryption scheme is semantically secure which is true assuming that the decisional Diffie-Hellman problem over a squared composite modulus of the form $N = pq$ is hard (i.e., a semantically secure public-key encryption scheme is a computationally hiding and statistically binding commitment, but the UC-security is not claimed at all). Consequently, the environment's output in the real-life model is indistinguishable from its output in the ideal-process.

5 Conclusion

In this paper, an simulation efficient reusable and non-erasure commitment schemes has been proposed and analyzed. Under the assumption that the decisional Diffie-Hellman problem over a squared composite modulus N is hard, we have shown that the proposed commitment scheme is secure in the universally composable model within the common reference model.

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. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
3. Canetti, R.: A new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)
4. 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)
5. 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)
6. 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)
7. Dodis, Y., Shoup, V., Walfish, S.: Efficient constructions of composable commitments and zero-knowledge proofs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 515–535. Springer, Heidelberg (2008)
8. Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-Interactive and Non-Malleable Commitment. In: STOC 1998, pp. 141–150 (1998)
9. Garay, J.A., MacKenzie, P.D., Yang, K.: Strengthening Zero-Knowledge Protocols Using Signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 177–194. Springer, Heidelberg (2003)
10. Naor, M.: Bit Commitment Using Pseudorandomness. J. Cryptology 4(2), 151–158 (1991)
11. 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)

Certificateless Hybrid Signcryption

Fagen Li[1,2,3], Masaaki Shirase[3], and Tsuyoshi Takagi[3]

[1] School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu 610054, China
[2] Key Laboratory of Computer Networks and Information Security,
Xidian University, Xi'an 710071, China
[3] School of Systems Information Science,
Future University-Hakodate, Hakodate 041-8655, Japan
{fagenli,shirase,takagi}@fun.ac.jp

Abstract. Signcryption is a cryptographic primitive that fulfills both the functions of digital signature and public key encryption simultaneously, at a cost significantly lower than that required by the traditional signature-then-encryption approach. In this paper, we address a question whether it is possible to construct a hybrid signcryption scheme in the certificateless setting. This question seems to have never been addressed in the literature. We answer the question positively in this paper. In particular, we extend the concept of signcryption tag-KEM to the certificateless setting. We show how to construct a certificateless signcryption scheme using certificateless signcryption tag-KEM. We also give an example of certificateless signcryption tag-KEM.

Keywords: Certificateless signcryption, hybrid signcryption, signcryption tag-KEM, DEM.

1 Introduction

Confidentiality, integrity, non-repudiation and authentication are the important requirements for many cryptographic applications. A traditional approach to achieve these requirements is to sign-then-encrypt the message. Signcryption, first proposed by Zheng [13], is a cryptographic primitive that fulfills both the functions of digital signature and public key encryption simultaneously, at a cost significantly lower than that required by the traditional signature-then-encryption approach. In Zheng's scheme, the public key of a user is essentially a random bit string picked from a given set. So, the signcryption does not provide the authorization of the user by itself. This problem can be solved via a certificate which provides an unforgeable and trusted link between the public key and the identity of the user by the signature of a certificate authority (CA), and there is a hierarchical framework that is called public key infrastructure (PKI) to issue and manage certificates. However, the certificates management including revocation, storage, distribution and the computational cost of certificates verification is the main difficulty against traditional PKI.

To simplify key management procedures of traditional PKI, Shamir [11] proposed the concept of identity-based cryptography (IBC) in 1984. The idea of IBC is to get rid of certificates by allowing the user's public key to be any binary string that uniquely identifies the user. Examples of such strings include email addresses and IP addresses. The main practical benefit of IBC is in greatly reducing the need for the public key certificates. But IBC uses a trusted third party called private key generator (PKG). The PKG generates the secret keys of all of its users, so a user can decrypt only if the PKG has given a secret key to it (so, certification is implicit), hence reduces the amount of storage and computation. On the other hand, the dependence on the PKG who can generate all users' private keys inevitably causes the key escrow problem to the IBC.

To solve the key escrow problem in the IBC, Al-Riyami and Paterson [2] introduced a new paradigm called certificateless cryptography. The certificateless cryptography does not require the use of certificates and yet does not have the built-in key escrow feature of IBC. It is a model for the use of public key cryptography that is intermediate between traditional PKI and IBC. A certificateless system still makes use of a trusted third party which is called the key generating center (KGC). By way of contrast to the PKG in the IBC, the KGC does not have access to the user's private key. Instead, the KGC supplies a user with a partial private key that the KGC computes from the user's identity and a master key. The user then combines the partial private key with some secret information to generate the actual private key. The system is not identity-based, because the public key is no longer computable from a user's identity. When Alice wants to send a message to Bob in a certificateless system, she must obtain Bob's public key. However, no authentication of Bob's public key is necessary and no certificate is required. In 2008, Barbosa and Farshim [4] introduced the notion of certificateless signcryption (CLSC) and proposed an efficient scheme.

The practical way to perform secrecy communication for large messages is to use hybrid encryption that separates the encryption into two parts: one part uses public key techniques to encrypt a one-time symmetric key; the other part uses the symmetric key to encrypt the actual message. In such a construction, the public key part of the algorithm is known as the key encapsulation mechanism (KEM) while the symmetric key part is known as the data encapsulation mechanism (DEM). A formal treatment of this paradigm originates in the work of Cramer and Shoup [7]. The resulting KEM-DEM hybrid encryption paradigm has received much attention in recent years. It is very attractive as it gives a clear separation between the various parts of the cipher allowing for modular design. In [1], Abe, Gennaro, and Kurosawa introduced tag-KEM which takes as input a tag in KEM. Bentahar et al. [5] extended KEM into identity-based and certificateless settings and gave generic constructions of identity-based KEM (IB-KEM) and certificateless KEM (CL-KEM).

The use of hybrid techniques to build signcryption schemes has been studied by Dent [8,9]. He generalized KEM to signcryption KEM which includes an authentication in KEM. However, he only consider the insider security for authenticity. That is, if the sender's private key is exposed, an attacker is able to recover the key generated by signcryption KEM. The full insider security [3] means that (a) if the sender's private key is exposed, an attacker is still not able to recover the message from the ciphertext and (b) if the receiver's private key is exposed, an attacker is still not able to forge a ciphertext. In 2006, Bjørstad and Dent [6] showed how to built signcryption schemes using tag-KEM. However, they also only consider the insider security for authenticity and not for confidentiality. In 2008, Tan [12] proposed full insider secure signcryption KEM and tag-KEM in the standard model. Tan's schemes are insider secure for both authenticity and confidentiality. Note that the using of tag-KEM yields simpler scheme descriptions and better generic security reductions.

All the above hybrid signcryption schemes [6,8,9,12] are not in the certificateless setting. In this paper, we address a question whether it is possible to construct a hybrid signcryption scheme in the certificateless setting. This question seems to have never been addressed in the literature. We answer the question positively in this paper. In particular, we extend the concept of signcryption tag-KEM to the certificateless setting. We show that a CLSC scheme can be constructed by using a certificateless signcryption tag-KEM (CLSC-TKEM) and a DEM. We also give an example of CLSC-TKEM. Our scheme is insider secure for both authenticity and confidentiality.

2 Preliminaries

2.1 Certificateless Signcryption (CLSC)

A generic CLSC scheme consists of the following six algorithms.

- `Setup`: This algorithm takes as input the security parameter 1^k and returns the KGC's master secret key msk and system parameters $params$ including a master public key mpk and descriptions of message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and randomness space $\mathcal{R}$. This algorithm is executed by the KGC, which publishes $params$.
- `Extract-Partial-Private-Key`: This algorithm takes as input $params$, msk and a user's identity $ID \in \{0,1\}^*$, and returns a partial private key D_{ID}. This algorithm is run by the KGC, after verifying the user's identity.
- `Generate-User-Keys`: This algorithm takes as input $params$ and an identity ID, and outputs a secret value x_{ID} and a public key PK_{ID}. This algorithm is run by a user to obtain a public key and a secret value which can be used to construct a full private key. The public key is published without certification.
- `Set-Private-Key`: This algorithm takes as input a partial private key D_{ID} and a secret value x_{ID}, and returns the full private key S_{ID}. Again, this algorithm is run by a user to construct the full private key.
- `Signcrypt`: This algorithm takes as input $params$, a plaintext message $m \in \mathcal{M}$, the sender's full private key S_{ID_s}, identity ID_s and public key PK_{ID_s}, and the receiver's identity ID_r and public key PK_{ID_r}, and outputs a ciphertext $\sigma \in \mathcal{C}$.

– `Unsigncrypt`: This algorithm takes as input *params*, a ciphertext σ, the sender's identity ID_s and public key PK_{ID_s}, and the receiver's full private key S_{ID_r}, identity ID_r and public key PK_{ID_r}, and outputs a plaintext m or a failure symbol $\perp$ if σ is an invalid ciphertext.

Barbosa and Farshim [4] defines the security notions for CLSC schemes. A CLSC scheme should satisfy confidentiality (indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2)) and unforgeability (existential unforgeability against adaptive chosen messages attacks (UF-CMA)). For the stronger notion of insider security, we use the notion of strong existential unforgeability (sUF-CMA). The strong existential unforgeability means that an adversary wins if it outputs a valid message/signcryption pair (m, σ) for identities ID_s and ID_r and the signcryption σ was not returned by the signcryption oracle when queried on the message m. As in [4], we do not consider attacks targeting signcryptions where the identities of the sender and receiver are the same. That is, we disallow such queries to relevant oracles and do not accept this type of signcryption as a valid forgery. Please see the full version of this paper for details [10].

2.2 Date Encapsulation Mechanism (DEM)

A DEM is a symmetric encryption scheme which consists of the following two algorithms.

– `Enc`: This algorithm takes as input 1^k, a key K and a message $m \in \{0,1\}^*$, and outputs a ciphertext $c \in \{0,1\}^*$, where $K \in \mathcal{K}_{\mathrm{DEM}}$ is a key in the given key space, and m is a bit string of arbitrary length. We denote this as $c \leftarrow \mathtt{Enc}(K, m)$.
– `Dec`: This algorithm takes as input a key K and a ciphertext c, and outputs the message $m \in \{0,1\}^*$ or a symbol $\perp$ to indicate that the ciphertext is invalid.

For the purposes of this paper, it is only required that a DEM is secure with respect to indistinguishability against passive attackers (IND-PA).

3 Certificateless Signcryption Tag-KEM (CLSC-TKEM)

In this section, we extend the concept of signcryption tag-KEM to the certificateless setting. We give the formal definition for certificateless signcryption tag-KEM (CLSC-TKEM).

3.1 Generic Scheme

A generic CLSC-TKEM consists of the following seven algorithms.

– `Setup`: Same to CLSC described in Section 2.
– `Partial-Private-Key-Extract`: Same to CLSC described in Section 2.
– `Generate-User-Keys`: Same to CLSC described in Section 2.

– `Set-Private-Key`: Same to CLSC described in Section 2.
– `Sym`: This is symmetric key generation algorithm which takes as input the *params*, the sender's full private key S_{ID_s}, identity ID_s and public key PK_{ID_s}, the receiver's identity ID_r and public key PK_{ID_r}, and outputs a symmetric key K together with internal state information ω. Here $K \in \mathcal{K}_{\text{CLSC-TKEM}}$ is a key in the space of possible session keys at a given security level. We denote this as $(K, \omega) \leftarrow \texttt{Sym}(params, S_{ID_s}, ID_s, PK_{ID_s}, ID_r, PK_{ID_r})$.
– `Encap`: This is key encapsulation algorithm which takes as input the state information ω and an arbitrary tag τ, and returns an encapsulation $\psi \in \mathcal{E}_{\text{CLSC-TKEM}}$. We denote this as $\psi \leftarrow \texttt{Encap}(\omega, \tau)$.
– `Decap`: This is decapsulation algorithm which takes as input the *params*, an encapsulation ψ, a tag τ, the sender's identity ID_s and public key PK_{ID_s}, the receiver's full private key S_{ID_r}, identity ID_r and public key PK_{ID_r}, and outputs a key K or a special symbol $\perp$ indicating invalid encapsulation. We denote this as $K \leftarrow \texttt{Decap}(params, \psi, \tau, ID_s, PK_{ID_s}, S_{ID_r}, ID_r, PK_{ID_r})$.

We make the consistency constraint that if

$$(K, \omega) \leftarrow \texttt{Sym}(params, S_{ID_s}, ID_s, PK_{ID_s}, ID_r, PK_{ID_r}) \text{ and } \psi \leftarrow \texttt{Encap}(\omega, \tau),$$

then

$$K \leftarrow \texttt{Decap}(params, \psi, \tau, ID_s, PK_{ID_s}, S_{ID_r}, ID_r, PK_{ID_r}).$$

3.2 Security Notions

A CLSC-TKEM should satisfy confidentiality and unforgeability. To define the security notions for CLSC-TKEM, we simply adapt the security notions of CLSC into the TKEM framework. There are two types of adversary against a CLSC-TKEM: Type I and Type II. A Type I adversary models an attacker which is a common user of the system and is not in possession of the KGC's master secret key. But it is able to adaptively replace users'public keys with (valid) public keys of its choice. A Type II adversary models an honest-but-curious KGC who knows the KGC's master secret key. But it cannot replace users' public keys.

For confidentiality, we consider two games "`IND-CCA2-I`" and "`IND-CCA2-II`" where a Type I adversary $\mathcal{A}_I$ and a Type II adversary $\mathcal{A}_{II}$ interact with their "challenger" in these two games, respectively. Note that the challenger keeps a history of "query-answer" while interacting with the attackers. Now we describe the two games.

`IND-CCA2-I`: This is the game in which $\mathcal{A}_I$ interacts with the "challenger":

`Initial`: The challenger runs $(params, msk) \leftarrow \texttt{Setup}(1^k)$ and gives *params* to $\mathcal{A}_I$. The challenger keeps master secret key *msk* to itself.

`Phase 1`: The adversary $\mathcal{A}_I$ can perform a polynomially bounded number of queries in an adaptive manner.

– `Extract partial private key`: $\mathcal{A}_I$ chooses an identity ID. The challenger computes $D_{ID} \leftarrow$`Extract-Partial-Private-Key`$(params, msk, ID)$ and sends D_{ID} to $\mathcal{A}_I$.

– `Extract private key:` $\mathcal{A}_I$ chooses an identity ID. The challenger first computes $D_{ID} \leftarrow$`Extract-Partial-Private-Key`$(params, msk, ID)$ and then $(x_{ID}, PK_{ID}) \leftarrow$`Generate-User-Keys`$(params, ID)$. Finally, it sends the result of $S_{ID} \leftarrow$`Set-Private-Key`(x_{ID}, D_{ID}) to $\mathcal{A}_I$. The adversary is not allowed to query any identity for which the corresponding public key has been replaced. This restriction is imposed due to the fact that it is unreasonable to expect that the challenger is able to provide a full private key for a user for which it does not know the secret value.
– `Request public key:` $\mathcal{A}_I$ chooses an identity ID. The challenger computes $(x_{ID}, PK_{ID}) \leftarrow$`Generate-User-Keys`$(params, ID)$ and sends PK_{ID} to $\mathcal{A}_I$.
– `Replace public key:` $\mathcal{A}_I$ may replace a public key PK_{ID} with a value chosen by it.
– `Symmetric key generation queries:` $\mathcal{A}_I$ chooses a sender's identity ID_s and a receiver's identity ID_r. The challenger finds S_{ID_s} from its "query-answer" list and runs $(K, \omega) \leftarrow$ `Sym`$(params, S_{ID_s}, ID_s, PK_{ID_s}, ID_r, PK_{ID_r})$. The challenger then stores the value ω (hidden from the view of the adversary, and overwriting any previously stored values), and sends the symmetric key K to $\mathcal{A}_I$. Note that, it is possible that the challenger is not aware of the sender's secret value, if the associated public key has been replaced. In this case, we require the adversary to provide it. We disallow queries where $ID_s = ID_r$.
– `Key encapsulation queries:` $\mathcal{A}_I$ produces an arbitrary tag τ. The challenger checks whether there exists a stored value ω. If not, it returns $\perp$ and terminates. Otherwise it erases the value from storage and returns $\psi \leftarrow$ `Encap`(ω, τ) to $\mathcal{A}_I$.
– `Key decapsulation queries:` The adversary $\mathcal{A}_I$ chooses a sender's identity ID_s, a receiver's identity ID_r, an encapsulation ψ, and a tag τ. The challenger finds S_{ID_r} from its "query-answer" list and sends the result of `Decap`$(params, \psi, \tau, ID_s, PK_{ID_s}, S_{ID_r}, ID_r, PK_{ID_r})$ to $\mathcal{A}_I$. Note that, it is possible that the challenger is not aware of the receiver's secret value, if the associated public key has been replaced. In this case, we require the adversary to provide it. We also disallow queries where $ID_s = ID_r$.

`Challenge:` The adversary $\mathcal{A}_I$ decides when Phase 1 ends. $\mathcal{A}_I$ generates a sender's identity ID_s^* and a receiver's identity ID_r^* on which it wishes to be challenged. Note that ID_r^* should not be queried to extract a private key in Phase 1. Note also that ID_r^* cannot be equal to an identity for which both the public key has been replaced and the partial private key has been extracted. The challenger computes $(K_1, \omega^*) \leftarrow$ `Sym`$(params, S_{ID_s^*}, ID_s^*, PK_{ID_s^*}, ID_r^*, PK_{ID_r^*})$. Then the challenger chooses $K_0 \leftarrow \mathcal{K}_{\text{CLSC-TKEM}}$ and a bit $b \in \{0, 1\}$ randomly, and sends K_b to $\mathcal{A}_I$. When $\mathcal{A}_I$ receives K_b, it may ask the same queries as previously. Then $\mathcal{A}_I$ generates a tag τ^*. The challenger computes $\psi^* \leftarrow$ `Encap`(ω^*, τ^*) and sends it to $\mathcal{A}_I$ as a challenge encapsulation.

`Phase 2:` The adversary $\mathcal{A}_I$ can ask a polynomially bounded number of queries adaptively again as in Phase 1. The same rule is applied here: $\mathcal{A}_I$ cannot extract the private key for ID_r^*. $\mathcal{A}_I$ cannot extract the partial private key for ID_r^* if the public key of this identity has been replaced before the challenge phase. In addition, $\mathcal{A}_I$ cannot make a decapsulation query on (K_b, ψ^*) under ID_s^* and ID_r^*, unless the public key $PK_{ID_s^*}$ or $PK_{ID_r^*}$ has been replaced after the challenge phase.

`Guess`: The adversary $\mathcal{A}_I$ produces a bit b' and wins the game if $b' = b$.

The advantage of $\mathcal{A}_I$ is defined to be

$$\mathrm{Adv}_{\mathrm{CLSC-TKEM}}^{\mathrm{IND-CCA2-I}}(\mathcal{A}_I) = |2\Pr[b' = b] - 1|,$$

where $\Pr[b' = b]$ denotes the probability that $b' = b$.

`IND-CCA2-II`: This is the game in which $\mathcal{A}_{II}$ interacts with the "challenger":

`Initial`: The challenger runs $(params, msk) \leftarrow \texttt{Setup}(1^k)$ and gives both $params$ and msk to $\mathcal{A}_{II}$.

`Phase 1`: The adversary $\mathcal{A}_{II}$ can perform a polynomially bounded number of queries in an adaptive manner. Note that we do not need `Extract partial private key` since $\mathcal{A}_{II}$ can computes partial private keys by itself.

- `Extract private key`: Same to CLSC-TKEM's `IND-CCA2-I` game.
- `Request public key`: Same to CLSC-TKEM's `IND-CCA2-I` game.
- `Symmetric key generation queries`: Same to CLSC-TKEM's `IND-CCA2-I` game.
- `Key encapsulation queries`: Same to CLSC-TKEM's `IND-CCA2-I` game.
- `Key decapsulation queries`: Same to CLSC-TKEM's `IND-CCA2-I` game.

`Challenge`: The adversary $\mathcal{A}_{II}$ decides when Phase 1 ends. $\mathcal{A}_{II}$ generates a sender's identity ID_s^* and a receiver's identity ID_r^* on which it wishes to be challenged. Note that ID_r^* should not be queried to extract a private key in Phase 1. The challenger runs $(K_1, \omega^*) \leftarrow \texttt{Sym}(params, S_{ID_s^*}, ID_s^*, PK_{ID_s^*}, ID_r^*, PK_{ID_r^*})$. Then the challenger chooses $K_0 \leftarrow \mathcal{K}_{\mathrm{CLSC-TKEM}}$ and a bit $b \in \{0, 1\}$ randomly, and sends K_b to $\mathcal{A}_I$. When $\mathcal{A}_{II}$ receives K_b, it may ask the same queries as previously. Then $\mathcal{A}_{II}$ generates a tag τ^*. The challenger computes $\psi^* \leftarrow \texttt{Encap}(\omega^*, \tau^*)$ and sends it to $\mathcal{A}_{II}$ as a challenge encapsulation.

`Phase 2`: The adversary $\mathcal{A}_{II}$ can ask a polynomially bounded number of queries adaptively again as in Phase 1. $\mathcal{A}_{II}$ cannot extract the private key for ID_r^*. In addition, $\mathcal{A}_{II}$ cannot make a decapsulation query on (K_b, ψ^*) under ID_s^* and ID_r^*, unless the public key $PK_{ID_s^*}$ or $PK_{ID_r^*}$ has been replaced after the challenge phase.

`Guess`: The adversary $\mathcal{A}_{II}$ produces a bit b' and wins the game if $b' = b$.

The advantage of $\mathcal{A}_{II}$ is defined to be

$$\mathrm{Adv}_{\mathrm{CLSC-TKEM}}^{\mathrm{IND-CCA2-II}}(\mathcal{A}_{II}) = |2\Pr[b' = b] - 1|,$$

where $\Pr[b' = b]$ denotes the probability that $b' = b$.

Definition 1. *A CLSC-TKEM scheme is said to be IND-CCA2-I secure (resp. IND-CCA2-II secure) if there is no PPT adversary $\mathcal{A}_I$ (resp. $\mathcal{A}_{II}$) which wins* `IND-CCA2-I` *(resp.* `IND-CCA2-II`*) with non-negligible advantage. A CLSC-TKEM scheme is said to be IND-CCA2 secure if it is both IND-CCA2-I secure and IND-CCA2-II secure.*

Notice that the adversary is allowed to extract the private key of ID_s^* in the `IND-CCA2-I` and `IND-CCA2-II` games. This condition corresponds to the stringent requirement of insider security for confidentiality of signcryption [3].

For the strong existential unforgeability, we consider two games "`sUF-CMA-I`" and "`sUF-CMA-II`" where a Type I adversary $\mathcal{F}_I$ and a Type II adversary $\mathcal{F}_{II}$ interact with their "challenger" in these two games, respectively. Note that the challenger keeps a history of "query-answer" while interacting with the attackers. Now we describe the two games.

`sUF-CMA-I`: This is the game in which $\mathcal{F}_I$ interacts with the "challenger":

`Initial`: The challenger runs $(params, msk) \leftarrow \texttt{Setup}(1^k)$ and gives $params$ to $\mathcal{F}_I$. The challenger keeps master secret key msk to itself.

`Attack`: The adversary $\mathcal{F}_I$ performs a polynomially bounded number of queries just like in the CLSC-TKEM's `IND-CCA2-I` game.

`Forgery`: $\mathcal{F}_I$ produces a quadruple $(\tau^*, \psi^*, ID_s^*, ID_r^*)$. Note that ID_s^* should not be queried to extract a private key. Note also that ID_s^* cannot be equal to an identity for which both the public key has been replaced and the partial private key has been extracted. In addition, ψ^* was not returned by the key encapsulation oracle on the input (τ^*, ID_s^*, ID_r^*) during Attack stage. $\mathcal{F}_I$ wins the game if the result of $\texttt{Decap}(params, \psi^*, \tau^*, ID_s^*, PK_{ID_s^*}, S_{ID_r^*}, ID_r^*, PK_{ID_r^*})$ is not the $\perp$ symbol.

The advantage of $\mathcal{F}_I$ is defined as the probability that it wins.

`sUF-CMA-II`: This is the game in which $\mathcal{F}_{II}$ interacts with the "challenger":

`Initial`: The challenger runs $(params, msk) \leftarrow \texttt{Setup}(1^k)$ and gives both $params$ and msk to $\mathcal{F}_{II}$.

`Attack`: The adversary $\mathcal{F}_{II}$ performs a polynomially bounded number of queries just like in the CLSC-TKEM's `IND-CCA2-II` game.

`Forgery`: $\mathcal{F}_{II}$ produces a quadruple $(\tau^*, \psi^*, ID_s^*, ID_r^*)$. ID_s^* should not be queried to extract a private key. In addition, ψ^* was not returned by the key encapsulation oracle on the input (τ^*, ID_s^*, ID_r^*) during Attack stage. $\mathcal{F}_{II}$ wins the game if the result of $\texttt{Decap}(params, \psi^*, \tau^*, ID_s^*, PK_{ID_s^*}, S_{ID_r^*}, ID_r^*, PK_{ID_r^*})$ is not the $\perp$ symbol.

The advantage of $\mathcal{F}_{II}$ is defined as the probability that it wins.

Definition 2. *A CLSC-TKEM scheme is said to be sUF-CMA-I secure (resp. sUF-CMA-II secure) if there is no PPT adversary $\mathcal{F}_I$ (resp. $\mathcal{F}_{II}$) which wins* `sUF-CMA-I` *(resp.* `sUF-CMA-II`*) with non-negligible advantage. A CLSC-TKEM scheme is said to be sUF-CMA secure if it is both sUF-CMA-I secure and sUF-CMA-II secure.*

Note that the adversary is allowed to extract the private key of ID_r^* in the above definition. Again, this condition corresponds to the stringent requirement of insider security for signcryption [3].

4 Certificateless Hybrid Signcryption

We can combine a CLSC-TKEM with a DEM to form a CLSC scheme. We describe it in Figure 1. Note that the tag is the ciphertext output by the DEM.

Such construction yields simpler scheme descriptions and better generic security reductions.

We give the security results for such construction in Theorems 1 and 2.

Theorem 1. *Let CLSC be a certificateless hybrid signcryption scheme constructed from a CLSC-TKEM and a DEM. If the CLSC-TKEM is IND-CCA2 secure and the DEM is IND-PA secure, then CLSC is IND-CCA2 secure. In particular, we have*

$$\mathrm{Adv}_{\mathrm{CLSC}}^{\mathrm{IND-CCA2-i}}(\mathcal{A}) \leq 2\mathrm{Adv}_{\mathrm{CLSC-TKEM}}^{\mathrm{IND-CCA2-i}}(\mathcal{B}_1) + \mathrm{Adv}_{\mathrm{DEM}}^{\mathrm{IND-PA}}(\mathcal{B}_2),$$

where $i \in \{I, II\}$, $\mathrm{Adv}_{\mathrm{CLSC}}^{\mathrm{IND-CCA2-i}}(\mathcal{A})$ is the advantage of the IND-CCA2 adversary against CLSC, $\mathrm{Adv}_{\mathrm{CLSC-TKEM}}^{\mathrm{IND-CCA2-i}}(\mathcal{B}_1)$ is the advantage of the IND-CCA2 adversary against CLSC-TKEM, and $\mathrm{Adv}_{\mathrm{DEM}}^{\mathrm{IND-PA}}(\mathcal{B}_2)$ is the advantage of the IND-PA adversary against DEM.

Proof. See the full version of this paper [10]. □

Theorem 2. *Let CLSC be a certificateless hybrid signcryption scheme constructed from a CLSC-TKEM and a DEM. If the CLSC-TKEM is sUF-CMA secure, then CLSC is sUF-CMA secure. In particular, we have*

$$\mathrm{Adv}_{\mathrm{CLSC}}^{\mathrm{sUF-CMA-i}}(\mathcal{F}) \leq \mathrm{Adv}_{\mathrm{CLSC-TKEM}}^{\mathrm{sUF-CMA-i}}(\mathcal{B}),$$

where $i \in \{I, II\}$, $\mathrm{Adv}_{\mathrm{CLSC}}^{\mathrm{sUF-CMA-i}}(\mathcal{F})$ is the advantage of the sUF-CMA adversary against CLSC, and $\mathrm{Adv}_{\mathrm{CLSC-TKEM}}^{\mathrm{sUF-CMA-i}}(\mathcal{B})$ is the advantage of the resulting sUF-CMA adversary against CLSC-TKEM.

Proof. See the full version of this paper [10]. □

5 An Example of CLSC-TKEM

The Barbosa-Farshim CLSC scheme [4] fits the new generic framework. Here we give an example of CLSC-TKEM based on the Barbosa-Farshim scheme. If we combine the CLSC-TKEM with a DEM as Figure 1, we can get a scheme that is very similar to the Barbosa-Farshim scheme.

The CLSC-TKEM consists of the following seven algorithms.

- `Setup`: Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. $\hat{e} : G_1 \times G_1 \to G_2$ is a pairing. Let H_1, H_2, H_3, and H_4 be four cryptographic hash functions where $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to \{0,1\}^n$, $H_3 : \{0,1\}^* \to G_1$, and $H_4 : \{0,1\}^* \to G_1$. Here n is the key length of a DEM. The PKG chooses a master secret key $s \in Z_q^*$ randomly and computes $P_{pub} \leftarrow sP$. The PKG publishes system parameters $\{G_1, G_2, n, \hat{e}, P, P_{pub}, H_1, H_2, H_3, H_4\}$ and keeps the master key s secret.

```
CLSC.Setup: On input $1^k$:
1. $(params, msk) \leftarrow$ CLSC-TKEM.Setup($1^k$)
2. Output the system parameters $params$ and the master secret key $msk$
CLSC.Partial-Private-Key-Extract: On input the $params$, $msk$, and an identity $ID \in \{0,1\}^*$:
1. $D_{ID} \leftarrow$ CLSC-TKEM.Partial-Private-Key-Extract($params, msk, ID$)
2. Output the partial private key $D_{ID}$ of the identity $ID$
CLSC.Generate-User-Keys: On input the $params$ and an identity $ID \in \{0,1\}^*$:
1. $(x_{ID}, PK_{ID}) \leftarrow$ CLSC-TKEM.Generate-User-Keys($params, ID$)
2. Output the secret value $x_{ID}$ and the public key $PK_{ID}$ of the identity $ID$
CLSC.Set-Private-Key: On input the partial private key $D_{ID}$ and the secret value $x_{ID}$:
1. $S_{ID} \leftarrow$ CLSC-TKEM.Set-Private-Key($D_{ID}, x_{ID}$)
2. Output the full private key $S_{ID}$
CLSC.Signcrypt: On input the $params$, a message $m \in \{0,1\}^*$, the sender's full private key $S_{ID_s}$, identity $ID_s$ and public key $PK_{ID_s}$, the receiver's identity $ID_r$ and public key $PK_{ID_r}$:
1. $(K, \omega) \leftarrow$ CLSC-TKEM.Sym($params, S_{ID_s}, ID_s, PK_{ID_s}, ID_r, PK_{ID_r}$)
2. $c \leftarrow$ DEM.Enc($K, m$)
3. $\psi \leftarrow$ CLSC-TKEM.Encap($\omega, c$)
4. Output the ciphertext $\sigma \leftarrow (\psi, c)$
CLSC.Unsigncrypt: On input the $params$, a ciphertext $\sigma$, the sender's identity $ID_s$ and public key $PK_{ID_s}$, the receiver's full private key $S_{ID_r}$, identity $ID_r$ and public key $PK_{ID_r}$:
1. $K \leftarrow$CLSC-TKEM.Decap($params, \psi, c, ID_s, PK_{ID_s}, S_{ID_r}, ID_r, PK_{ID_r}$)
2. If $K = \perp$, then output $\perp$ and stop
3. $m \leftarrow$DEM.Dec($K, c$)
4. Output the message $m$
```

Fig. 1. Certificateless hybrid signcryption

– `Partial-Private-Key-Extract`: Given an identity $ID \in \{0,1\}^*$, the PKG computes $Q_{ID} \leftarrow H_1(ID)$ and returns the partial private key $D_{ID} \leftarrow sQ_{ID}$.
– `Generate-User-Keys`: A user with identity ID chooses a random element x_{ID} from Z_q as the secret value, and sets $PK_{ID} \leftarrow x_{ID}P$ as the public key.
– `Set-Private-Key`: Given a partial private key D_{ID} and a secret value x_{ID}, this algorithm returns the full private key $S_{ID} \leftarrow (x_{ID}, D_{ID})$.
– `Sym`: Given the sender's full private key S_{ID_s}, identity ID_s and public key PK_{ID_s}, the receiver's identity ID_r and public key PK_{ID_r}, this algorithm works as follows.
  1. Choose $r \in Z_q^*$ randomly.
  2. Compute $U = rP$ and $T \leftarrow \hat{e}(P_{pub}, Q_{ID_r})^r$.
  3. Compute $K \leftarrow H_2(U, T, rPK_{ID_r}, ID_r, PK_{ID_r})$.
  4. Output K and set $\omega \leftarrow (r, U, S_{ID_s}, ID_s, PK_{ID_s}, ID_r, PK_{ID_r})$.
– `Encap`: Given the state information ω and an arbitrary tag τ, this algorithm works as follows.
  1. Compute $H \leftarrow H_3(U, \tau, ID_s, PK_{ID_s})$.
  2. Compute $H' \leftarrow H_4(U, \tau, ID_s, PK_{ID_s})$.
  3. Compute $W \leftarrow D_{ID_s} + rH + x_{ID_s}H'$.
  4. Output $\psi \leftarrow (U, W)$.
– `Decap`: Given the the sender's identity ID_s and public key PK_{ID_s}, the receiver's full private key S_{ID_r}, identity ID_r and public key PK_{ID_r}, an encapsulation ψ and a tag τ, this algorithm works as follows.
  1. Compute $H \leftarrow H_3(U, \tau, ID_s, PK_{ID_s})$.
  2. Compute $H' \leftarrow H_4(U, \tau, ID_s, PK_{ID_s})$.
  3. If $\hat{e}(P_{pub}, Q_{ID_s})\hat{e}(U, H)\hat{e}(PK_{ID_s}, H') = \hat{e}(P, W)$, compute $T = \hat{e}(D_{ID_r}, U)$ and output the $K \leftarrow H_2(U, T, x_{ID_r}U, ID_r, PK_{ID_r})$. Otherwise, output symbol $\perp$.

We give the security results for the CLSC-TKEM in Theorems 3 and 4.

Theorem 3. *In the random oracle model, the above CLSC-TKEM is IND-CCA2 secure under the assumption that the gap bilinear Diffie-Hellman problem is intractable.*

Proof. See the full version of this paper [10]. □

Theorem 4. *In the random oracle model, the above CLSC-TKEM is sUF-CMA secure under the assumption that the GDH′ problem is intractable.*

Proof. See the full version of this paper [10]. □

6 Conclusions

In this paper, we extended the concept of signcryption tag-KEM to the certificateless setting. We showed that a certificateless signcryption scheme can be constructed by combining a certificateless signcryption tag-KEM with a DEM. To show that our framework is reasonable, we also gave an example of certificateless signcryption tag-KEM based on the Barbosa-Farshim certificateless signcryption scheme.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant Nos. 60673075, 60803133 and 60873233), the Key Laboratory of Computer Networks and Information Security of Xidian University (2008CNIS-02), and the Youth Science and Technology Foundation of UESTC. Fagen Li is supported by the JSPS postdoctoral fellowship for research in Japan.

References

1. Abe, M., Gennaro, R., Kurosawa, K.: Tag-KEM/DEM: a new framework for hybrid encryption. Journal of Cryptology 21(1), 97–130 (2008)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
3. 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)
4. Barbosa, M., Farshim, P.: Certificateless signcryption. In: ACM Symposium on Information, Computer and Communications Security-ASIACCS 2008, Tokyo, Japan, pp. 369–372 (2008)
5. Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic constructions of identity-based and certificateless KEMs. Journal of Cryptology 21(2), 178–199 (2008)
6. Bjørstad, T.E., Dent, A.W.: Building better signcryption schemes with tag-kEMs. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 491–507. Springer, Heidelberg (2006)
7. 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)
8. Dent, A.W.: Hybrid signcryption schemes with outsider security. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 203–217. Springer, Heidelberg (2005)
9. Dent, A.W.: Hybrid signcryption schemes with insider security. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 253–266. Springer, Heidelberg (2005)
10. Li, F., Shirase, M., Takagi, T.: Certificateless hybrid signcryption. Full version will be available in Cryptology ePrint Archive
11. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
12. Tan, C.H.: Insider-secure signcryption KEM/tag-KEM schemes without random oracles. In: The Third International Conference on Availability, Reliability and Security-ARES 2008, Barcelona, Spain, pp. 1275–1281 (2008)
13. 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)

On Non-representable Secret Sharing Matroids

Qi Cheng[1,*], Yong Yin[1], Kun Xiao[1], and Ching-Fang Hsu[2]

[1] Engineering Department, Wuhan Digital Engineering Institute, Wuhan, China
[2] College of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan, China
cherryjingfang@gmail.com

Abstract. The characterization of the access structures of ideal secret sharing schemes is one of the main open problems in secret sharing and has important connections with matroid theory. Because of its difficulty, it has been studied for several particular families of access structures. Multipartite access structures, in which the set of participants is divided into several parts and all participants in the same part play an equivalent role, have been studied in seminal works on secret sharing by Shamir, Simmons, and Brickell, and also recently by several authors.. In the EUROCRYPT'07, Farras made a important contribution to this work: By using discrete polymatroids, they obtained a necessary condition and a sufficient condition for a multipartite access structure to be ideal respectively. In particular, they further gave a very difficult open problem, that is, characterizing the representable discrete polymatroids, i.e., which discrete polymatroids are representable and which ones are non-representable. In this paper, by dealing with a family of matroids derived from the Vamos matroid, which was the first matroid that was proved to be non-representable, we obtain a family of non-representable matroids. As a consequence, we extend it to the general case and obtain a sufficient condition for a discrete polymatroid to be non-representable, which is a new contribution to the open problem given by Farras.

Keywords: Ideal secret sharing schemes, Ideal access structures, Multipartite access structures, Discrete polymatroids, Vamos matroid.

1 Introduction

Secret sharing schemes were introduced independently by Shamir [2] and Blakley [1] in 1979. In a secret sharing scheme, every participant receives a share of a secret value. Only the qualified sets of participants, which form the access structure of the scheme, can recover the secret value from their shares. This paper deals exclusively with unconditionally secure perfect secret sharing schemes, that is, the shares of the participants in a non-qualified set do not provide any information about the secret value.

The length of the shares is the main measure of the complexity of secret sharing schemes. In general, the shares must be much larger than the secret. An access structure is said to be ideal if it admits an ideal secret sharing scheme. The characterization

[*] Corresponding author.

of ideal access structures is one of the main open problems in secret sharing and has important connections with matroid theory.

A necessary condition for an access structure to be ideal was given by Brickell and Davenport [4], who proved that every ideal access structure is matroid-related. Matroids that are obtained from ideal secret sharing schemes are said to be secret sharing representable (or ss-representable for short). Vamos matroid was the first matroid that was proved to be non-ss-representable. Nevertheless, as a consequence of the results in [3], all representable matroids (that is, matroid that can be represented by a matrix over some finite field) are ss-representable. This implies a sufficient condition for an access structure to be ideal. Namely, an access structure is ideal if it is related to a representable matroid.

Due to the difficulty of finding general results, the characterization of ideal access structures has been studied for several particular classes of access structures. Multipartite access structure, informally, is that the set of participants can be divided into several parts in such a way that all participants in the same part play an equivalent role in the structure. Since we can always consider as many parts as participants, every access structure is multipartite. More accurately, we can consider in any access structure the partition that is derived from a suitable equivalence relation on the set of participants. Because of its practical interest, secret sharing for multipartite access structures has been studied by several authors[2,3,5,6,7,8].

Recently, in the EUROCRYPT'07, Farras[9] made a important contribution to this work by using discrete polymatroids. In particular, for solving the main open problems in secret sharing, they further gave a very difficult open problem, that is, characterizing the representable discrete polymatroids, i.e., which discrete polymatroids are representable and which ones are non-representable. In this paper, by dealing with a family of matroids derived from the Vamos matroid, which was the first matroid that was proved to be non-representable, we obtain a family of non-representable matroids. As a consequence, we extend it to the general case and obtain a sufficient condition for a discrete polymatroid to be non-representable, which is a new contribution to the open problem given by Farras.

2 Definitions and Preliminaries

In this section we review some basic definitions and notations that will be used through the paper.

2.1 Matroids and Ideal Secret Sharing

The reader is referred to [12] for an introduction to secret sharing and to [10, 11] for general references on Matroid Theory.

A matroid $\mathcal{M} = (\mathcal{Q}, \mathcal{I})$ is formed by a finite set $\mathcal{Q}$ together with a family $\mathcal{I} \subseteq \mathcal{P}(\mathcal{Q})$ ($\mathcal{P}(\mathcal{Q})$ is the power set of the set $\mathcal{Q}$.) such that

1. $\phi \in \mathcal{I}$, and

2. if $I_1 \in \mathcal{I}$ and $I_2 \subseteq I_1$, then $I_2 \in \mathcal{I}$, and

3. if $I_1, I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then there exists $x \in I_2 - I_1$ such that $I_1 \cup \{x\} \in \mathcal{I}$.

The set $\mathcal{Q}$ is the ground set of the matroid $\mathcal{M}$ and the elements of $\mathcal{I}$ are called the independent sets of $\mathcal{M}$. The bases of the matroid are the maximally independent sets. The family $\mathcal{B}$ of the bases determines the matroid. Moreover, by [10, Theorem 1.2.5], $\mathcal{B} \subseteq \mathcal{P}(\mathcal{Q})$ is the family of bases of a matroid on $\mathcal{Q}$ if and only if

1. $\mathcal{B}$ is nonempty, and
2. for every $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 - B_2$, there exists $y \in B_2 - B_1$ such that $(B_1 - \{x\}) \cup \{y\}$ is in $\mathcal{B}$.

All bases have the same number of elements, which is the rank of $\mathcal{M}$ and is denoted $r(\mathcal{M})$. The dependent sets are those that are not independent. A circuit is a minimally dependent subset. A matroid is said to be connected if, for every two points $x, y \in \mathcal{Q}$, there exists a circuit C with $x, y \in C$. The rank of $X \subseteq \mathcal{Q}$, which is denoted $r(X)$, is the maximum cardinality of the subsets of X that are independent. Observe that the rank of $\mathcal{Q}$ is the rank of the matroid $\mathcal{M}$ that was defined before. The rank function $r : \mathcal{P}(\mathcal{Q}) \to \mathbb{Z}$ of a matroid satisfies

1. $0 \le r(X) \le |X|$ for every $X \subseteq \mathcal{Q}$, and
2. r is monotone increasing: if $X \subseteq Y \subseteq \mathcal{Q}$, then $r(X) \le r(Y)$, and
3. r is submodular: $r(X \cap Y) + r(X \cup Y) \le r(X) + r(Y)$ for every $X, Y \subseteq \mathcal{Q}$.

Let $\mathbb{K}$ be a field. A matroid $\mathcal{M} = (\mathcal{Q}, \mathcal{I})$ is $\mathbb{K}$-representable (or representable for short) if there exists a matrix M over $\mathbb{K}$ whose columns are indexed by the elements of $\mathcal{Q}$ such that a subset $I = \{i_1, \ldots, i_k\} \subseteq \mathcal{Q}$ is independent if and only if the corresponding columns of M are independent. In this situation, we say that the matrix M is a $\mathbb{K}$-representation of the matroid $\mathcal{M}$.

Let $\mathbb{K}$ be a finite field and let $\mathcal{M} = (\mathcal{Q}, \mathcal{I})$ be a $\mathbb{K}$-representable matroid. Let $p_0 \in \mathcal{Q}$ be special participant called dealer.and $\mathcal{Q} = P \cup \{p_0\}$. For every $k \times (n+1)$ matrix M representing $\mathcal{M}$ over $\mathbb{K}$, let E be a vector space of finite dimention $\dim E = k$ over $\mathbb{K}$. For every $i \in \mathcal{Q}$, we define a surjective linear mapping: $\pi_i : E \to \mathbb{K}$, and the i-th column of M corresponds to the linear form π_i. In that situation, for every random choice of an element $x \in E$, we can obtain $s_i = \pi_i(x) \in \mathbb{K}$ is the share of the participant $i \in P$ and $s = \pi_{p_0}(x) \in \mathbb{K}$ is the shared secret value. Hence, by the columns of M, we define an ideal secret sharing scheme with access structure $\Gamma_{p_0}(\mathcal{M})$. Therefore, the access structures that are related to representable matroids are ideal.

2.2 Multipartite Access Structures and Multipartite Matroids

We write $\mathcal{P}(P)$ for the power set of the set P. An m-partition $\Pi = \{P_1, \ldots, P_m\}$ of a set P is a disjoint family of m nonempty subsets of P with $P = P_1 \cup \ldots \cup P_m$. Let $\Lambda \subseteq \mathcal{P}(P)$ be a family of subsets of P. For a permutation σ on P, we define $\sigma(\Lambda) = \{\sigma(A) : A \in \Lambda\} \subseteq \mathcal{P}(P)$. A family of subsets $\Lambda \subseteq \mathcal{P}(P)$ is said to be Π-partite if $\sigma(\Lambda) = \Lambda$ for every permutation σ such that $\sigma(P_i) = P_i$ for every $P_i \in \Pi$. We say that Λ is m-partite if it is Π-partite for some m-partition Π. These concepts can be applied to access structures, which are actually families of subsets, and they can be applied as well to the family of independent sets of a matroid. A matroid $\mathcal{M} = (\mathcal{Q}, \mathcal{I})$ is Π-partite if $\mathcal{I} \subseteq \mathcal{P}(\mathcal{Q})$ is Π-partite.

Let $\mathcal{M} = (\mathcal{Q}, \mathcal{I})$ be a connected matroid and, for a point $p_0 \in \mathcal{Q}$, let $\Pi = \{P_1, \ldots, P_m\}$ and $\Pi_0 = \{\{p_0\}, P_1, \ldots, P_m\}$ be partitions of the sets $P = \mathcal{Q} - \{p_0\}$ and $\mathcal{Q}$ respectively. Then the access structure $\Gamma = \Gamma_{p_0}(\mathcal{M})$ is Π-partite if and only if the matroid $\mathcal{M}$ is Π_0-partite.

The partition Π' is a refinement of the partition Π if every set in Π' is a subset of some set in Π. Clearly, if $\Lambda \subseteq \mathcal{P}(P)$ is Π-partite and Π' is a refinement of Π, then Λ is Π'-partite. Among all partitions Π for which a family of subsets $\Lambda \subseteq \mathcal{P}(P)$ is Π-partite, there exists a partition Π_Λ that is not a refinement of any other such partition. Following [13], we consider the following equivalence relation: two elements $p, q \in P$ are said to be equivalent according to Λ if the transposition τ_{pq} satisfies $\tau_{pq}(\Lambda) = \Lambda$. The partition Π_Λ is the one defined by this equivalence relation. It is not difficult to check that Λ is Π-partite if and only if Π is a refinement of Π_Λ.

For every integer $m \geq 1$, we consider the set $J_m = \{1, \ldots, m\}$. Let $\mathbb{Z}_+^m$ denote the set of vectors $u = (u_1, \ldots, u_m) \in \mathbb{Z}^m$ with $u_i \geq 0$ for every $i \in J_m$. For a partition $\Pi = \{P_1, \ldots, P_m\}$ of a set P and for every $A \subseteq P$ and $i \in J_m$, we define $\Pi_i(A) = |A \cap P_i|$. Then the partition Π defines a mapping $\Pi : \mathcal{P}(P) \to \mathbb{Z}_+^m$ by considering $\Pi(A) = (\Pi_1(A), \ldots, \Pi_m(A))$. If $\Lambda \subseteq \mathcal{P}(P)$ is Π-partite, then

$A \in \Lambda$ if and only if $\Pi(A) \in \Pi(\Lambda)$. That is, Λ is completely determined by the partition Π and the set of vectors $\Pi(\Lambda) \subset \mathbb{Z}_+^m$.

Discrete polymatroids, a combinatorial object introduced by Herzog and Hibi [13], are closely related to multipartite matroids and, because of that, they play an important role in the characterization of ideal multipartite access structures. Before giving the definition of discrete polymatroid, we need to introduce some notation. If $u, v \in \mathbb{Z}_+^m$, we write $u \leq v$ if $u_i \leq v_i$ for every $i \in J_m$, and we write $u < v$ if $u \leq v$ and $u \neq v$. The vector $w = u \vee v$ is defined by $w_i = \max(u_i, v_i)$. The modulus of a vector $u \in \mathbb{Z}_+^m$ is $|u| = u_1 + \cdots + u_m$. For every subset $X \subseteq J_m$, we write $u(X) = (u_i)_{i \in X} \in \mathbb{Z}_+^{|X|}$ and $|u(X)| = \sum_{i \in X} u_i$

A discrete polymatroid on the ground set J_m is a nonempty finite set of vectors $D \subset \mathbb{Z}_+^m$ satisfying:

1. if $u \in D$ and $v \in \mathbb{Z}_+^m$ is such that $v \leq u$, then $v \in D$, and
2. for every pair of vectors $u, v \in D$ with $|u| < |v|$, there exists $w \in D$ with $u < w \leq u \vee v$.

The next proposition, which is easily proved from the axioms of the independent sets of a matroid, shows the relation between multipartite matroids and discrete polymatroids.

Proposition 2.1. Let Π be a partition of a set $\mathcal{Q}$ and let $\mathcal{I} \subseteq \mathcal{P}(\mathcal{Q})$ be a Π-partite family of subsets. Then $\mathcal{I}$ is the family of the independent sets of a Π-partite matroid $\mathcal{M} = (\mathcal{Q}, \mathcal{I})$ if and only if $\Pi(\mathcal{I}) \subset \mathbb{Z}_+^m$ is a discrete polymatroid.

A basis of a discrete polymatroid D is a maximal element in D, that is, a vector $u \in D$ such that there does not exist any $v \in D$ with $u < v$. Similarly to matroids, a discrete polymatroid is determined by its bases. Specifically, the following result is proved in [13, Theorem 2.3].

Proposition 2.2. A nonempty subset $\mathcal{B} \subset \mathbb{Z}_+^m$ is the family of bases of a discrete polymatroid if and only if it satisfies:

1. all elements in $\mathcal{B}$ have the same modulus, and
2. for every $u \in \mathcal{B}$ and $v \in \mathcal{B}$ with $u_i > v_i$, there exists $j \in J_m$ such that $u_j < v_j$ and $u - e_i + e_j \in \mathcal{B}$, where e_i denotes the i-th vector of the canonical basis of $\mathbb{Z}^m$.

The rank function of a discrete polymatroid D with ground set J_m is the function $h:\mathcal{P}(J_m)\to\mathbb{Z}$ defined by $h(X)=\max\{|u(X)|:u\in D\}$. The next proposition is a consequence of [13, Theorem 3.4].

Proposition 2.3. A function $h:\mathcal{P}(J_m)\to\mathbb{Z}$ is the rank function of a discrete polymatroid with ground set J_m if and only if it satisfies

1. $h(\phi)=0$, and
2. h is monotone increasing: if $X\subseteq Y\subseteq J_m$, then $h(X)\le h(Y)$, and
3. h is submodular: if $X,Y\subseteq J_m$, then $h(X\bigcup Y)+h(X\bigcap Y)\le h(X)+h(Y)$.

Moreover, a polymatroid D is completely determined by its rank function. Specifically, $D=\left\{u\in\mathbb{Z}_+^m:\ |u(X)|\le h(X)\text{ for all }X\subseteq J_m\right\}$.

For a discrete polymatroid D with ground set J_m and for every $X\subseteq J_m$, we define the discrete polymatroid $D(X)$ with ground set X by $D(X)=\{u(X):u\in D\}\subset\mathbb{Z}_+^{|X|}$. This concept will be very useful in this paper.

Let $\mathbb{K}$ be a field, E a $\mathbb{K}$-vector space, and $V_1,\ldots,V_m$ subspaces of E. It is not difficult to check that the mapping $h:\mathcal{P}(J_m)\to\mathbb{Z}$ defined by $h(X)=\dim(\sum_{i\in X}V_i)$ is the rank function of a discrete polymatroid $D\subset\mathbb{Z}_+^m$. In this situation, we say that D is $\mathbb{K}$-representable and the subspaces $V_1,\ldots,V_m$ are a $\mathbb{K}$-representation of D. The next proposition is proved in [9, Theorem 7.1].

Proposition 2.4. Let $\mathcal{M}=(\mathcal{Q},\mathcal{I})$ be a Π-partite matroid and let $D=\Pi(\mathcal{I})$ be its associated discrete polymatroid. If $\mathcal{M}$ is $\mathbb{K}$-representable, then so is D. In addition, if D is $\mathbb{K}$-representable, then $\mathcal{M}$ is representable over some finite extension of $\mathbb{K}$.

3 A Family of Non-representable Secret Sharing Matroids

In this section, we give a family of non-representable matroids derived from the Vamos matroid. Firstly, we introduce the Vamos matroid and give the proof of Vamos matroid being a non-representable multipartite matroid. Afterwards, through combining the partition of the ground set of Vamos matroid, we construct three "matroids", which are proved to be non-representable. However, according to the definition of matroid, we obtain these three "matroids" are pseudo matroids. Finally, from the concept of $D(X)$ defined above, a family of non-representable matroids derived from the Vamos matroid is obtained, which we call Vamos Family.

3.1 Vamos Matroid

The definition of Vamos matroid is as follows:

Definition 3.1. The Vamos matroid is defined on $\mathcal{Q}=\{1,2,3,4,5,6,7,8\}$ with bases all 4-sets except the five 4-sets which are: $\{1,2,3,4\}$, $\{1,2,5,6\}$, $\{1,2,7,8\}$, $\{3,4,5,6\}$, $\{3,4,7,8\}$.

The following proposition gives a new proof of the Vamos matroid being a non-representable multipartite matroid.

Proposition 3.1. The Vamos matroid is non-representable.

Proof. For a partition $\Pi_0=\{P_1,P_2,P_3,P_4\}$ ($P_1=\{1,2\}, P_2=\{3,4\}, P_3=\{5,6,\}, P_4=\{7,8\}$) of the ground set $\mathcal{Q}$, the partition Π_0 defines a mapping $\Pi_0:\mathcal{P}(\mathcal{Q})\to\mathbb{Z}_+^4$. For every non-basis 4-set A, we compute $\Pi_0(A)$ and obtain $(2,2,0,0),(2,0,2,0),(2,0,0,2),(0,2,2,0),(0,2,0,2)$. Similarly, for every basis B, we compute $\Pi_0(B)$ and obtain $(1,1,1,1),(1,1,2,0),(1,1,0,2),(1,2,1,0),(1,2,0,1),(1,0,1,2)$, $(1,0,2,1)$, $(0,1,1,2)$, $(0,1,2,1)$, $(0,2,1,1)$, $(2,1,0,1),(2,1,1,0),(2,0,1,1),(0,0,2,2)$. We can verify that for every 3-set C, there must exist a basis B such that $\Pi_0(C)<\Pi_0(B)$ and $C\subset B$. Therefore, all 3-sets are independent.

Suppose that over some finite field $\mathbb{K}$ there exists a matrix M which is a representation of the Vamos matroid, and every element $i\in\mathcal{Q}$ correspond to the column vector v_i of M. Apparently, all vectors of M are non-zero vectors. Arbitrary four column vectors of M are linearly independent except (v_1,v_2,v_3,v_4), (v_1,v_2,v_5,v_6), (v_1,v_2,v_7,v_8), (v_3,v_4,v_5,v_6), (v_3,v_4,v_7,v_8). Because all 3-sets are independent, for every one of these five vector groups, its rank is 3 and every vector in it can be uniquely represented by the other three vectors over $\mathbb{K}$. The following operations are over the finite field $\mathbb{K}$:

For the vector group (v_1,v_2,v_7,v_8), let $v_8=a_1v_1+a_2v_2+a_7v_7$ (1)

For the vector group (v_3,v_4,v_7,v_8), let $v_8=a_3v_3+a_4v_4+a_7{}'v_7$. (2)

where $a_1,a_2,a_7,a_3,a_4,a_7{}'\in\mathbb{K}$ and $a_1,a_2,a_7,a_3,a_4,a_7{}'\neq 0$.
Simultaneous equations (1)(2), then

$$(a_7{}'-a_7)v_7=a_1v_1+a_2v_2-a_3v_3-a_4v_4 \tag{3}$$

For the vector group (v_1,v_2,v_3,v_4), let $v_4 = b_1v_1 + b_2v_2 + b_3v_3$ (4)

where $b_1,b_2,b_3 \in \mathbb{K}$ and $b_1,b_2,b_3 \neq 0$.

If $(a_7{'} - a_7) \neq 0$, then simultaneous equations (3)(4) and we obtain (v_1,v_2,v_3,v_7) are linearly dependent. Since $\{1,2,3,7\}$ is a basis of the Vamos matroid, a contradiction. Hence, there must be $(a_7{'} - a_7) = 0$, that is $a_7{'} = a_7$, then from equation (3) we obtain:

$$a_1v_1 + a_2v_2 = a_3v_3 + a_4v_4 \tag{5}$$

For the vector group (v_1,v_2,v_5,v_6), let $v_6 = c_1v_1 + c_2v_2 + c_5v_5$ (6)

For the vector group (v_3,v_4,v_5,v_6), let $v_6 = c_3v_3 + c_4v_4 + c_5{'}v_5$ (7)

Similarly, we can obtain $c_5 = c_5{'}$ and $c_1v_1 + c_2v_2 = c_3v_3 + c_4v_4$ (8)

Computing equation $c_1(5) - a_1(8)$, then:

$$(a_2c_1 - a_1c_2)v_2 = (a_3c_1 - a_1c_3)v_3 + (a_4c_1 - a_1c_4)v_4 \tag{9}$$

Because (v_2,v_3,v_4) are linearly independent, then $a_2c_1 - a_1c_2 = 0$ (10)

Computing equation $c_1(1) - a_1(6)$, then:

$$c_1v_8 - a_1v_6 = (a_2c_1 - a_1c_2)v_2 + a_7c_1v_7 - a_1c_5v_5 \tag{11}$$

Simultaneous equations (10)(11), then $c_1v_8 - a_1v_6 = a_7c_1v_7 - a_1c_5v_5$ (12)

Due to $a_1,c_1,a_7,c_5 \neq 0$, from equation (12) we can obtain (v_5,v_6,v_7,v_8) are linearly dependent. Since $\{5,6,7,8\}$ is a basis of the Vamos matroid, a contradiction. Therefore, it is impossible that there exists a matrix M over some finite field $\mathbb{K}$ which is a representation of the Vamos matroid, that is, the Vamos matroid is non-representable.

3.2 Three Non-representable Pseudo Matroids

Through combining the partition of the ground set of Vamos matroid, we construct three "matroids" as follow:

Definition 3.2. The Pseudo-1 matroid is defined on $\mathcal{Q} = \{1,2,3,4,5,6,7,8\}$ with bases all 4-sets except the thirteen 4-sets which are: $\{1,2,3,4\}$, $\{1,2,5,6\}$, $\{1,2,5,7\}$, $\{1,2,5,8\}$, $\{1,2,6,7\}$, $\{1,2,6,8\}$, $\{1,2,7,8\}$, $\{3,4,5,6\}$, $\{3,4,5,7\}$, $\{3,4,5,8\}$, $\{3,4,6,7\}$, $\{3,4,6,8\}$, $\{3,4,7,8\}$.

Definition 3.3. The Pseudo-2 matroid is defined on $\mathcal{Q}=\{1,2,3,4,5,6,7,8\}$ with bases all 4-sets except the thirteen 4-sets which are: $\{1,2,3,4\}$, $\{1,2,5,6\}$, $\{1,3,5,6\}$, $\{1,4,5,6\}$, $\{2,3,5,6\}$, $\{2,4,5,6\}$, $\{3,4,5,6\}$, $\{1,2,7,8\}$, $\{1,3,7,8\}$, $\{1,4,7,8\}$, $\{2,3,7,8\}$, $\{2,4,7,8\}$, $\{3,4,7,8\}$.

Definition 3.4. The Pseudo-3 matroid is defined on $\mathcal{Q}=\{1,2,3,4,5,6,7,8\}$ with bases all 4-sets except the thirty-seven 4-sets which are: $\{1,2,3,4\}$, $\{1,2,5,6\}$, $\{1,3,5,6\}$, $\{1,4,5,6\}$, $\{2,3,5,6\}$, $\{2,4,5,6\}$, $\{3,4,5,6\}$, $\{1,2,5,7\}$, $\{1,3,5,7\}$, $\{1,4,5,7\}$, $\{2,3,5,7\}$, $\{2,4,5,7\}$, $\{3,4,5,7\}$, $\{1,2,5,8\}$, $\{1,3,5,8\}$, $\{1,4,5,8\}$, $\{2,3,5,8\}$, $\{2,4,5,8\}$, $\{3,4,5,8\}$, $\{1,2,6,7\}$, $\{1,3,6,7\}$, $\{1,4,6,7\}$, $\{2,3,6,7\}$, $\{2,4,6,7\}$, $\{3,4,6,7\}$, $\{1,2,6,8\}$, $\{1,3,6,8\}$, $\{1,4,6,8\}$, $\{2,3,6,8\}$, $\{2,4,6,8\}$, $\{3,4,6,8\}$, $\{1,2,7,8\}$, $\{1,3,7,8\}$, $\{1,4,7,8\}$, $\{2,3,7,8\}$, $\{2,4,7,8\}$, $\{3,4,7,8\}$.

In the following propositions, we prove that these three "matroids" stated above are all non-representable.

Proposition 3.2. The Pseudo-1 matroid is non-representable.

Proof. For a partition $\Pi_1=\{P_1,P_2,P_3\}$ ($P_1=\{1,2\}, P_2=\{3,4\}, P_3=\{5,6,7,8\}$) of the ground set $\mathcal{Q}$, the partition Π_1 defines a mapping $\Pi_1:\mathcal{P}(\mathcal{Q})\to\mathbb{Z}_+^3$. For every non-basis 4-set A, we compute $\Pi_1(A)$ and obtain $(2,2,0),(2,0,2),(0,2,2)$. Similarly, for every basis B, we compute $\Pi_1(B)$ and obtain $(1,1,2),(1,2,1),(1,0,3),(0,1,3),(2,1,1),(0,0,4)$. We can verify that for every 3-set C, there must exist a basis B such that $\Pi_1(C)<\Pi_1(B)$ and $C\subset B$. Therefore, all 3-sets are independent.

Suppose that over some finite field $\mathbb{K}$ there exists a matrix M which is a representation of the Pseudo-1 matroid, and every element $i\in\mathcal{Q}$ correspond to the column vector v_i of M. Apparently, all vectors of M are non-zero vectors. Arbitrary four column vectors of M are linearly independent except (v_1,v_2,v_3,v_4), (v_1,v_2,v_5,v_6), (v_1,v_2,v_5,v_7), (v_1,v_2,v_5,v_8), (v_1,v_2,v_6,v_7), (v_1,v_2,v_6,v_8), (v_1,v_2,v_7,v_8), (v_3,v_4,v_5,v_6), (v_3,v_4,v_5,v_7), (v_3,v_4,v_5,v_8), (v_3,v_4,v_6,v_7), (v_3,v_4,v_6,v_8), (v_3,v_4,v_7,v_8). Because all

3-sets are independent, for every one of these thirteen vector groups, its rank is 3 and every vector in it can be uniquely represented by the other three vectors over $\mathbb{K}$. The following proof is the same to the proof of Proposion 3.1.

Proposition 3.3. The Pseudo-2 matroid is non-representable.

Proof. For a partition $\Pi_2 = \{P_1, P_2, P_3\}$ ($P_1 = \{1,2,3,4\}, P_2 = \{5,6\}, P_3 = \{7,8\}$) of the ground set $\mathcal{Q}$, the partition Π_2 defines a mapping $\Pi_2 : \mathcal{P}(\mathcal{Q}) \to \mathbb{Z}_+^3$. For every non-basis 4-set A, we compute $\Pi_2(A)$ and obtain $(4,0,0),(2,0,2),(2,2,0)$. Similarly, for every basis B, we compute $\Pi_2(B)$ and obtain $(2,1,1),(3,1,0),(3,0,1),(1,1,2),(1,2,1),(0,2,2)$. We can verify that for every 3-set C, there must exist a basis B such that $\Pi_2(C) < \Pi_2(B)$ and $C \subset B$. Therefore, all 3-sets are independent. The following proof is the same to the proof of Proposion 3.1.

Proposition 3.4. The Pseudo-3 matroid is non-representable.

Proof. For a partition $\Pi_3 = \{P_1, P_2\}$ ($P_1 = \{1,2,3,4\}, P_2 = \{5,6,7,8\}$) of the ground set $\mathcal{Q}$, the partition Π_3 defines a mapping $\Pi_3 : \mathcal{P}(\mathcal{Q}) \to \mathbb{Z}_+^2$. For every non-basis 4-set A, we compute $\Pi_3(A)$ and obtain $(4,0),(2,2)$. Similarly, for every basis B, we compute $\Pi_3(B)$ and obtain $(1,3),(3,1),(0,4)$. We can verify that for every 3-set C, there must exist a basis B such that $\Pi_3(C) < \Pi_3(B)$ and $C \subset B$. Therefore, all 3-sets are independent. The following proof is the same to the proof of Proposion 3.1.

If these three non-representable "matroids" accord with the definition of matroid, it means there exist non-representable bipartite and tripartite matroids. However, we will show these three non-representable "matroids" are pseudo matroids.

From Proposition 2.2, for every $u \in \mathcal{B}$ and $v \in \mathcal{B}$ with $u_i > v_i$, there exists $j \in J_m$ such that $u_j < v_j$ and $u - e_i + e_j \in \mathcal{B}$, where e_i denotes the i-th vector of the canonical basis of $\mathbb{Z}^m$. In Pseudo-1 matroid, for $u = (2,1,1)$ and $v = (0,0,4)$ with $u_2 > v_2$, there only exists $u_3 < v_3$ but $(2,0,2)$ is not a basis. Therefore, Pseudo-1 matroid is not a matroid, namely, a pseudo matroid. Similarly, in Pseudo-2 matroid, for $u = (1,2,1)$ and $v = (3,1,0)$ with $u_3 > v_3$, there only exists $u_1 < v_1$ but $(2,2,0)$ is not a basis. Therefore, Pseudo-2 matroid is a pseudo matroid. In Pseudo-3 matroid, for $u = (3,1)$ and $v = (0,4)$ with $u_1 > v_1$, there only exists $u_2 < v_2$ but $(2,2)$ is not a basis. Therefore, Pseudo-3 matroid is a pseudo matroid. As a consequence, these three non-representable "matroids" are all pseudo matroids.

3.3 Vamos Family

For the Vamos matroid $\mathcal{M} = (\mathcal{Q}, \mathcal{I})$, there exists a partition $\Pi_0 = \{P_1, P_2, P_3, P_4\}$ ($P_1 = \{1,2\}, P_2 = \{3,4\}, P_3 = \{5,6,\}, P_4 = \{7,8\}$) of the ground set $\mathcal{Q}$, and the partition Π_0 defines a mapping $\Pi_0 : \mathcal{P}(\mathcal{Q}) \to \mathbb{Z}_+^4$ and, hence, we obtain a discrete polymatroid $D_V = \Pi_0(\mathcal{I})$ corresponding to the Vamos matroid.

Proposition 3.5. For a discrete polymatroid D with ground set J_m, if there exists $X \subseteq J_m$, where $|X| = 4$, such that $D(X) = D_V$, then D must be a non-representable discrete polymatroid, and hence, the multipartite matroid corresponding to D must be non-representable. All of these discrete polymatroids construct a family of non-representable matroids, that is, $F_{D_V} = \left\{ D \subset \mathbb{Z}_+^m : D(X) = D_V, X \subset J_m \text{且} |X| = 4 \right\}$, which we call Vamos Family.

The proof of Proposition 3.5 is very simple, which is a special case of the proof of Theorem 4.1. Suppose a discrete polymatroid D in Vamos Family is representable. We will obtain the Vamos matroid is representable, contradiction. Therefore, the discrete polymatroids in Vamos Family is non-representable.

4 A Sufficient Condition for a Discrete Polymatroid to Be Non-representable

In this section, we extend the Vamos Family to the general case and obtain a sufficient condition for a discrete polymatroid to be non-representable.

Theorem 4.1. Let $D \subset \mathbb{Z}_+^m$ be a discrete polymatroid with ground set J_m, if there exists $X \subseteq J_m$ such that $D(X) = \{u(X) : u \in D\} \subset \mathbb{Z}_+^{|X|}$ is a non-representable discrete polymatroid, then D must be a non-representable discrete polymatroid and, hence, the multipartite matroid corresponding to D must be non-representable.

Proof. Let $D \subset \mathbb{Z}_+^m$ be a discrete polymatroid with ground set J_m. There exists $X \subseteq J_m$ such that $D(X) = \{u(X) : u \in D\} \subset \mathbb{Z}_+^{|X|}$ is a non-representable discrete polymatroid. Suppose D is representable over some finite field $\mathbb{K}$, i.e., there exists a vector space $E = \mathbb{K}^s$ over $\mathbb{K}$, where $s = h(J_m)$, such that m subspaces $V_1, \dots, V_m$ of E are a $\mathbb{K}$-representation of D. Let $X = \{x_1, \dots, x_r\}$, where $|X| = r$ and, hence, the subspaces corresponding to the elements of $X \subseteq J_m$ are

$V_{x_1},\ldots,V_{x_r}$. Since $D(X)=\{u(X):u\in D\}\subset\mathbb{Z}_+^{|X|}$, it means r subspaces $V_{x_1},\ldots,V_{x_r}$ of $E=\mathbb{K}^s$ are a $\mathbb{K}$-representation of $D(X)$, namely, $D(X)$ is a $\mathbb{K}$-representable discrete polymatroid, contradiction. Therefore, D is a non-representable discrete polymatroid and, hence, the multipartite matroid corresponding to D must be non-representable.

As a consequence, Theorem 4.1 gives a sufficient condition for a discrete polymatroid to be non-representable.

5 Conclusion

In this paper, by dealing with a family of matroids derived from the Vamos matroid, which was the first matroid that was proved to be non-representable, we obtain a family of non-representable matroids. As a consequence, we extend it to the general case and obtain a sufficient condition for a discrete polymatroid to be non-representable, which is a new contribution to the open problem given by Farras.

References

1. Shamir, A.: How to share a secret. Commun. of the ACM 22, 612–613 (1979)
2. Blakley, G.R.: Safeguarding cryptographic keys. In: AFIPS Conference Proceedings, vol. 48, pp. 313–317 (1979)
3. Matus, F.: Matroid representations by partitions. Discrete Math. 203, 169–194 (1999)
4. Seymour, P.D.: On secret-sharing matroids. J. Combin. Theory Ser. B 56, 69–73 (1992)
5. Marti-Farre, J., Padro, C.: On Secret Sharing Schemes, Matroids and Polymatroids. Cryptology ePrint Archive, Report 2006/077, `http://eprint.iacr.org/2006/077`
6. Tassa, T.: Hierarchical threshold secret sharing. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 473–490. Springer, Heidelberg (2004)
7. Ng, S.-L., Walker, M.: On the composition of matroids and ideal secret sharing schemes. Des. Codes Cryptogr. 24, 49–67 (2001)
8. Collins, M.J.: A Note on Ideal Tripartite Access Structures. Cryptology ePrint Archive, Report 2002/193, `http://eprint.iacr.org/2002/193`
9. Farràs, O., Martí-Farré, J., Padró, C.: Ideal multipartite secret sharing schemes. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 448–465. Springer, Heidelberg (2007)
10. Oxley, J.G.: Matroid theory. Oxford Science Publications/ The Clarendon Press/ Oxford University Press, New York (1992)
11. Welsh, D.J.A.: Matroid Theory. Academic Press, London (1976)
12. Ng, S.-L.: Ideal secret sharing schemes with multipartite access structures. IEE Proc.-Commun. 153, 165–168 (2006)
13. Herzog, J., Hibi, T.: Discrete polymatroids. J. Algebraic Combin. 16, 239–268 (2002)

A Novel Adaptive Watermarking Scheme Based on Human Visual System and Particle Swarm Optimization

Shaomin Zhu[1] and Jianming Liu[2]

[1] China Electric Power Research Institute,
100192 Beijing, China
[2] State Grid Information & Telecommunication Co.Ltd,
100761 Beijing, China
{smzhu,jianming-liu}@sgcc.com.cn

Abstract. In this paper, we proposed a novel watermarking scheme based on adaptive quantization index modulation and singular value decomposition in the hybrid discrete wavelet transform (DWT) and discrete cosine transform (DCT). The secret watermark bits are embedded on the singular values vector of blocks within low frequency subband in host image hybrid DWT-DCT domain. To embed watermark imperceptibly, robustly and securely, we model the adaptive quantization steps by utilizing human visual system (HVS) characteristics and particle swarm optimization (PSO) algorithm. Experimental results demonstrate that the proposed scheme is robust to variety of image processing attacks. In the proposed algorithm the quantized embedding strategy is adopted, so no host image is needed for blind extraction of watermarking image.

Keywords: quantization index modulation, singular value decomposition, human visual system, particle swarm optimization, blind extraction.

1 Introduction

With the development and popularization of multimedia technology and computer network technology, various multimedia products such as image, audio, video and three-dimensional model are increasingly vulnerable to illegal possession, reproduction and dissemination. How to effectively protect the copyright and content integrity of multimedia information has been attached importance to by more and more researchers. Digital watermarking has emerged as a leading technique that could solve the fundamental problem of legal ownership and content authentication for digital multimedia data. In general, the watermarking scheme shall satisfy two properties. First, the watermark should not affect the quality of the host media and be imperceptible to human eyes. Second, if watermark is used for Internet applications such as transmitting data through a noisy channel or compressing data, the watermark must survive under those situations [1].

In accordance with embedded position, watermarking algorithm can be divided into two categories: space domain and transform domain. The transform domain algorithm robustness is better than the space domain algorithm generally. The common transform includes Discrete Cosine Transform (DCT), Discrete Fourier Transform

(DFT) and Discrete Wavelet Transform (DWT) and so on. Image matrix singular value decomposition (SVD) reflects the internal image characteristics and has good stability if image processing is performed. Therefore many watermarking schemes that are combined different transforms with SVD have been proposed lately. For example, Tsai and Yang [2] proposed an approach emphasized that watermark message bit is embedded on the blocks of the DCT coefficient's singular value within an original color image. Bao and Ma [3] described a hybrid DWT-SVD watermarking algorithm that watermark bits are embedded on the singular value of the blocks within wavelet subband of the original image. However, most of these hybrid transform methods are non-adaptive or may not exploit the HVS features for more effectively embedding robust and secure watermark.

Recently, many researchers focus on adaptive determination of the quantization parameters for SVD-based watermarking. They consider to solving this problem using adopting artificial intelligence techniques or analyzing statistical model of each block in the image. Aslantas [4][5] respectively utilized the genetic algorithm (GA) and particle swarm optimization algorithm (PSO) for optimizing the quantization steps. Lai et al. [6] choose the micro-genetic algorithm (micro-GA) for optimization quantization steps. However, these methods are all non-blinding algorithm so that the security and practicality of these algorithms are not robust. Qi et al. [7] applied a grid search algorithm to find the effective pairs for a set of training images covering a variety of textures ranging from high, medium, low, to extremely low textures. The quantization steps are determined to be adaptive to the statistical model of the block, but this approach is not incorporate with the human visual system for improving the adaptive capability of the watermarking system.

In this paper, we design a novel adaptive watermarking scheme by combing the HVS and PSO. The watermark bits are embedded on singular value vector of each embedding block within low frequency subband in the hybrid DWT-DCT domain. One quantization parameter is determined by exploiting the characteristics of HVS, the other quantization parameter is optimized through PSO algorithm. These two quantization parameters are combined for ensuring the final adaptive quantization steps are optimal for all embedding blocks and reaching better trade off between the imperceptibility and robustness of the digital watermarking system. Experimental results show that the proposed scheme not only has good visual quality irrespective of the nature of the images chosen but also is robust to image processing attacks.

The rest of the paper is structured as follows. In section 2, we present the basic idea and key techniques of PSO algorithm. In section 3, the proposed watermarking scheme combining with human visual masking and particle swarm optimization algorithm in the hybrid DWT-DCT is described. The performance analysis for the proposed watermarking scheme and comparisons with other designs are presented in section 4. Finally, the paper ends with a brief conclusion.

2 Particle Swarm Optimization

PSO is a population-based stochastic optimization method introduced first by Kennedy and Eberhart [8], derived by the social behavior of bird flocking and fish schooling. The population is called a swarm consisted of different particles, which have a position and a velocity and change their positions in multi-dimensional search space over time. Each particle represents a possible solution to the optimization problems in the multi-dimensional problem space. These particles start at a random initial position and search for the minimum or maximum of a given objective function by moving through the search space. After each iteration process in the search space, each particle records its own personal best solution as well as the discovered global best position. The movement of each particle depends only on its velocity and the location where good solutions have already been found by the particle itself or in neighboring particles. When a particle's neighborhood is defined as the whole swarm, the PSO is called the global version, otherwise it is called the local version [8].In the following parts global version will be discussed.

Let a swarm include different particles $X_i (i = 1, 2, ..., m)$ and the ith particle in a d-dimensional space be represented as $X_i = (x_{i1}, x_{i2}, ..., x_{id})$ .The best previous position of the ith particle $pbest_i$ is denoted by $P_i = (p_{i1}, p_{i2}, ..., p_{id})$ .The best global location is represented $pbest_g$ among all the particles. The velocity for the ith particle is represented as $V_i = (v_{i1}, v_{i2}, ..., v_{id})$ .During each iteration process, each particle in the swarm is updated the velocity and location towards its $pbest_i$ and $pbest_g$ locations each iteration according to following two equations respectively:

$$V_i = w_i v_i + c_1 \xi \left(pbest_i - X_i \right) + c_2 \eta \left(pbest_g - X_i \right) \tag{1}$$

$$X_i = X_i + V_i \tag{2}$$

where ξ and η are random variables drawn from a uniform distribution in the range [0,1] so as to provide a stochastic weighting of the different components participating in the particle velocity definition. c_1 and c_2 are two acceleration constants and called as cognitive acceleration and social acceleration respectively. They are factors regulating the particle relative velocities with respect to the best global and local positions respectively. The inertia weight w_i is used to decide a tradeoff between the global and local exploration capabilities of the swarm. Typical implementations of the PSO adapt the value of linearly decreasing it from 1.0 to near 0 over the execution. In general, the inertia weight w_i is set according to the following equation [9]:

$$w_i = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \tag{3}$$

where $iter_{\max}$ is the maximum number of iterations, and $iter$ is the current number of iterations.

3 Proposed Watermarking Scheme

In our image watermarking scheme, the watermark can be embedded into the host image by three steps. First, DWT is performed on the host image. Second, the low frequency component is segmented into non-overlapping blocks and these blocks are performed on DCT. Then a set of final quantization steps are modeled both the characteristics of the DCT domain human visual masking and particle swarm optimization of each block to ensure a high perceptual quality of watermarked image and a low bit error rate of the detected watermark. Finally, watermark is embedded into the singular values vector of each block by adaptive and optimized quantization steps. A diagram of our image watermark embedding scheme is shown in Fig.1.

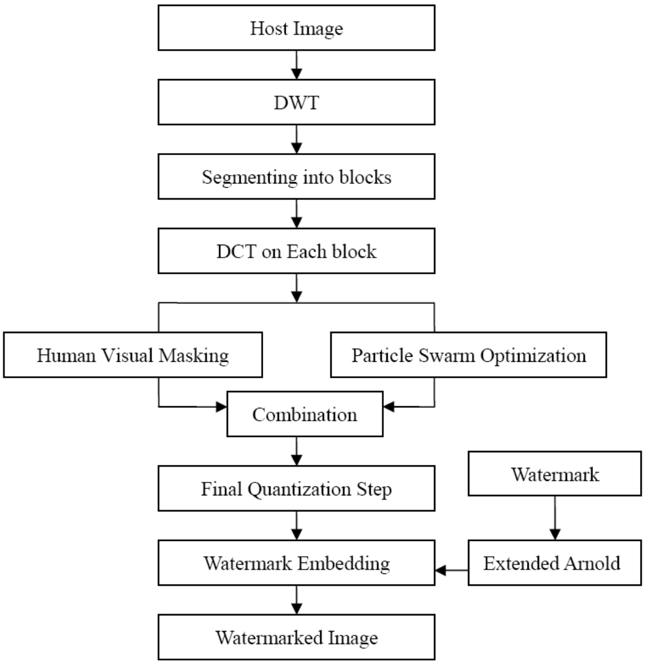


Fig. 1. Watermark embedding scheme

3.1 Watermarking Embedding Scheme

Suppose the host image is I_0 with m×n that is decomposed in j levels DWT, we obtain the low frequency subband LL_j and three high frequency subbands HL_j, LH_j, HH_j .To take the advantage of low frequency coefficients which have a higher energy value and robustness against various signal processing, the DCT is only performed on low frequency coefficient LL_j .The embedding procedure can be described as follows steps:

Step 1: Segment the LL_j into non-overlapping blocks A_i of size $w \times w$, $i = 1, 2, ..., M$,where M is the number of the blocks.

Step 2: Each block A_i is individually transformed to a frequency coefficient using DCT transform, and then compute the singular values vector of each frequency coefficient block A_i by SVD according to the following equation.

$$DCT(A_i) = U_i \sum{}_i V_i^T \tag{4}$$

Step 3: Compute $N_i^s = \|s_i\| + 1$ and quantize it by adaptive quantization step δ_i that represents the quantization level for N_i^s corresponds to the frequency coefficient block A_i according to the following equation.

$$N_i = \left\lfloor N_i^s \big/ \delta_i \right\rfloor (i = 1,2,...M) \tag{5}$$

where $s_i = (\sigma_{i1}, \sigma_{i2}, ..., \sigma_{iw})$, s_i denotes a vector formed by the singular values of the frequency coefficient block A_i.

Step 4:Embed each watermark bit by modifying integer number N_i, according to the following equation.

$$N_{iw} = \begin{cases} N_i + 1 & if \quad \begin{matrix} \mathrm{mod}(N_i, 2) = 1 \,\&\, w_i = 1 \\ \mathrm{mod}(N_i, 2) = 0 \,\&\, w_i = 0 \end{matrix} \\ N_i & else \qquad otherwise \end{cases} \tag{6}$$

Step 5: Compute the value $N_{iw}^s = \delta_i \times (N_{iw} + 1/2)$ and the modified singular values.

$$(\sigma'_{i1}, \sigma'_{i2}, ..., \sigma'_{iw}) = (\sigma_{i1}, \sigma_{i2}, ..., \sigma_{iw}) \times (N_{iw}^s \big/ N_i^s) \tag{7}$$

Step 6: Compute the watermarked block A'_i with modified singular values. The watermarked low frequency subband LL'_j is reshaped through A'_i performed on Inverse Discrete Cosine Transform (IDCT), then the watermarked image I'_0 is obtained utilizing Inverse Discrete Wavelet Transform (IDWT).

3.2 Adaptive Quantization Steps

The embedding strength is more or less proportional to the perceptual sensitivity to distortions for using adaptive quantization step size. In order to resist the normal signal processing and other different attacks, we wish the quantization step to be as high as possible. However, because the watermark directly affects the host image, it is obvious that the higher the quantization step, the lower the quality of the watermarked image will be. In other words, the robustness and the imperceptibility of the watermark are contradictory to each other. In this section, we propose a novel model for determining the adaptive quantization steps combining with the characteristics of the HVS and PSO in order to guarantee robustness and transparency of watermark.

3.2.1 Human Visual Masking

HVS is one of the most complex biological systems which include three stages: encoding, representation and interpretation [10]. Many factors cause human vision to have a limited sensitivity. For example, the surface of the cornea causes the refraction; the circular entrance of the pupil causes the diffraction; the optical lens has the chromatic aberration effects, and the mosaic of photo-receptors only does the spatial sampling process [10]. Human vision perceptive redundancy gives us a good opportunity to choose proper quantization step for embedding watermark in the image.

According to the Watson model [11] that is designed in DCT domain, the brighter the background, the higher the visibility threshold (luminance masking), the background texture more complex, the higher the visibility threshold (texture masking).We determine the luminance masking and texture masking based on the image features in DCT domain, and then combine these two masking together to get a comprehensive final masking. The main steps of the proposed HVS model can be described as follows:

Step 1: Segment the LL_j into non-overlapping blocks A_i of size $w \times w$, $i = 1, 2, ..., M$, where M is the number of the blocks.

Step 2: Calculate luminance masking M_i^L

The reason to use luminance is that the darkness and brightness in the image is reflected by luminance. The visibility of luminance threshold in the DCT domain depends on background luminance that is expressed by the Direct Current (DC) components. We can compute a luminance masking value M_i^L for each block A_i.

$$M_i^L = D_i(0,0) \tag{8}$$

where $D_i(0,0)$ is the DC coefficient of A_i that performed on DCT domain.

Step3:Calculate texture masking M_i^T

As for texture masking, the simplest method is to use the block variance.

$$M_i^T = \frac{1}{w \times w} \sum_{i=1}^{w} \sum_{j=1}^{w} (D_i(i,j) - \bar{D}_i)^2 \tag{9}$$

$$\bar{D}_i = \frac{1}{w \times w} \sum_{i=1}^{w} \sum_{j=1}^{w} D_i(i,j) \tag{10}$$

where $\bar{D}_i$ is average value of each block A_i DCT coefficients .

Step 4: Calculate quantization step δ_i for each block A_i. δ_{i0} is the basic step of each block that will be obtained by utilizing PSO for achieving optimal watermarking performance depending on both the transparency and the robustness factors.

$$\delta_i = \left\lfloor \log 2^{M_i^L \times M_i^T} \times 1000 \right\rfloor / 1000 + \delta_{i0} \tag{11}$$

3.2.2 Optimal Basic Step

This section illustrates how to use PSO help search proper basic step of each block in order to optimize watermark embedding process. It is a difficult for determining the proper values of multiple basic quantization steps. In some cases, the choice of them may be based on some general assumption [12]. Therefore, an efficient and optimal algorithm is required for achieving both invisibility and robustness. Here we use PSO to automatically determine these values without making any assumption. In the application of PSO, we should consider three essential components:

(1)Solution representation and Initialization

The representation scheme determines how the problem is structured, as well as the iteration operators that can be used [6].Each particle in the swarm represents a possible solution to the problem and hence consists of a set of basic step of each block. Meanwhile we randomly generate each particle value the in the initial swarm.

(2) Fitness function

The watermarked image transparency and robustness should be measured in order to formulate a proper fitness function. We adopt the same fitness function as used in [4], that is:

$$f_i = \left[\frac{m}{\sum_{i=1}^{m} NC_i(W_i', W)} - NC(I_0', I_0) \right]^{-1} \tag{12}$$

where NC_i denotes the two-dimensional normalized correlation, and m represents the number of attacking methods.

(3) PSO training operation

The similar PSO based watermarking algorithm proposed by Aslantas et al [5] is adopted. A diagram of PSO optimization training is shown in Fig.2.

1).Define the swarm size, particle size, cognitive acceleration and social acceleration and set the inertia weight of the swarm equation.

2).Generate the initial swarm, initial velocity, and initial fitness function value randomly.

3).Produce watermarked images utilizing the solutions in the swarm by means of embedding scheme combing with HVS.

4).Calculate the NC values between the host image and watermarked images.

5).Apply the attacks upon the watermarked images one by one.

6).Extract out watermarks from the attacked images using the extraction scheme.

7).Calculate the NC values between the watermark and the extracted ones.

8).Calculate the value of each particle, feedback optimal values to the PSO and obtain new particle value.

9).Repeat Steps 3-8 until the predefined termination criterion is satisfied.

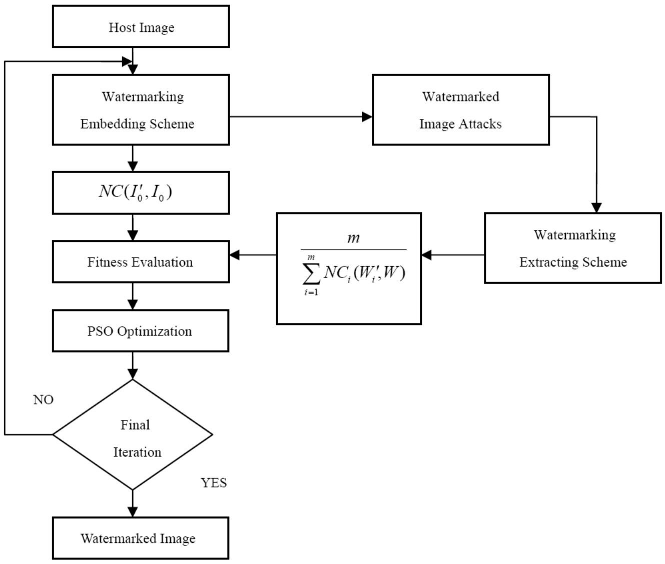

Host Image
Watermarking Embedding Scheme
Watermarked Image Attacks
$NC(I_0', I_0)$
Watermarking Extracting Scheme
Fitness Evaluation
$\frac{m}{\sum_{i=1}^{m} NC_i(W_i', W)}$
PSO Optimization
NO
Final Iteration
YES
Watermarked Image

Fig. 2. Diagram of PSO training

3.3 Watermarking Extracting Scheme

The watermark extracting scheme is the inverse of embedding procedure that neither needs the host image signal nor any other side information. Suppose the watermarked image is I_0' that is decomposed in j levels DWT, we obtain low frequency sub-band LL_j'. The extracting procedure is given as follows:

Step 1: Segment the LL_j' into non-overlapping blocks A_i' of size $w \times w$ $i = 1, 2, ..., M$, where M is the number of the blocks.
Step 2: Transform each block A_i' to the frequency coefficient by DCT, then compute $N_i'^s = \|s_i'\| + 1$ and quantize it by optimal final quantization step δ_i , where $s_i' = (\sigma_{i1}', \sigma_{i2}', ..., \sigma_{iw}')$, s_i' denotes a vector formed by the singular values of the frequency coefficient block A_i' .

$$N_i' = \left\lfloor N_i'^s / \delta_i \right\rfloor (i = 1, 2, ..., M) \tag{13}$$

Step 3: Extract watermark bits according to the following equation.

$$w_i' = \begin{cases} 1 & if \quad \mathrm{mod}(N_i', 2) = 0 \\ 0 & else \quad \mathrm{mod}(N_i', 2) = 1 \end{cases} \tag{14}$$

Step 4: Reshape the original binary watermark image by performing inverse extended Arnolded scrambling on watermark image according to the extracted watermark bits.

4 Experimental Results

The performance of the proposed watermarking scheme is tested on a large number of experiments. Here the results are presented for grayscale 8-bit different texture images of size 512×512.The logo used for watermark image is the binary image of size 32×32. For 1-level wavelet decomposition Haar filter coefficients are used. In the PSO training process, c_1 and c_2 are set as 1.2 and 1.8 respectively, the number of particles are chosen as 20 and the number of total iterations is set as 20. At the same time, the experiments compare the performance of proposed scheme with Qi's scheme that is a novel watermarking based on adaptive quantization steps determined by the statistical model of the block in the image.

Fig.3 (a) and (b) are the host image and the watermarked image. In subjective, it seems difficult to distinguish the difference between the host and the watermarked images by the human eye.

Fig. 3. Host image and watermarked image

In objective, The Peak Signal to Noise Ratio (PSNR) is used efficiently measure of visual fidelity between the host image and the watermarked image. It is found that the image quality measured by PSNR among the watermarked image is greater than 43dB. This indicates that the proposed watermarking scheme has good visual fidelity. Meanwhile Tab.1 simultaneity compares in terms of PSNR for evaluating the visual fidelity performance of our scheme with Qi's scheme.

Table 1. The values of PSNR

Image	Proposed Scheme	Qi' Scheme
	PSNR(dB)	PSNR(dB)
Lena	43.94	43.67
Peppers	44.63	44.26
Baboon	44.83	44.54

The watermarking scheme should be robust to signal processing which could be intentional or unintentional. Normalized Correlation (NC) and Bit Error Ratio (BER) that are defined as following equation are adopted for evaluating the robustness of the watermarking scheme. Without any image attacks, the NC value is 1 and the BER value is 0. In other words, the watermark can be completely extracted.

$$NC(W',W) = \frac{\sum_{i=1}^{p}\sum_{j=1}^{q} w(i,j)w'(i,j)}{\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q} w^2(i,j)}\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q} w'^2(i,j)}} \tag{15}$$

$$BER(W',W) = \frac{\sum_{i=1}^{p}\sum_{j=1}^{q} \sim\overline{w'(i,j)\oplus w(i,j)}}{p\times q}\times 100\% \tag{16}$$

Tab. 2. illustrates results after various attacks on the watermarked image including noise addition, low pass filtering, scaling and cropping.

Table 2. Experiment results after different attacks

Attack	NC	BER
Gauss noise(var 0.1%)	0.9689	2.54
Pepper&Salt nosie(density 0.1%)	0.9963	0.29
Gaussian filtering	0.9804	1.56
Median filtering	0.9038	7.71
Scaling 50%	0.9766	0.98
Cropping 25%	0.8569	15.38

Then, we test the robustness by JPEG compression that is one of the most important attacks with different quality factors (QF). Related results are shown in Tab.3.for the BER versus JPEG compression. Compared with Qi's scheme, it is observed that there is higher robustness to JPEG compression with proposed scheme.

Table 3. Experiment results comparison under JPEG compression

QF	Proposed scheme's BER(%)	Qi's scheme BER(%)
40	0.18	0.2
30	0.78	1.25
20	8.49	8.98
10	43.45	45.13

5 Conclusion

In this paper, we propose a novel adaptive watermarking scheme based on HVS and PSO in hybrid DWT-DCT transform. After decomposing the host image by DWT and DCT, watermark bits are embedded into the singular values of the embedded blocks within low frequency coefficients subband of the host image. The quantization steps of watermarking are modeled based on HVS characteristics and PSO. For adopting the quantization steps, the embedded information can be extracted without host image. The experimental results show that the proposed scheme preserves not only the high perceptual quality, but also is robust against many different types of attacks.

References

1. Langelaar, G., Setyawan, I., Lagendijk, R.L.: Watermarking Digital Image and Video Data. IEEE Signal Processing Magazine 12, 20–46 (2000)
2. Tsai, C.-F., Yang, W.-Y.: Real-time color image watermarking based on D-SVD scheme. In: Mery, D., Rueda, L. (eds.) PSIVT 2007. LNCS, vol. 4872, pp. 289–297. Springer, Heidelberg (2007)
3. Paul, B., Xiaohu, M.: Image Adaptive Watermarking Using Wavelet Domain Singular Value Decomposition. IEEE Transactions on Circuits and Systems for Video Technology 15, 96–102 (2005)
4. Aslantas, V.: A Singular-Value Decomposition-Based Image Watermarking Using Genetic Algorithm. International Journal of Electronics and Communications 62, 386–393 (2007)
5. Aslantas, V., Latif Dogan, A., Ozturk, S.: DWT-SVD Based Image Watermarking Using Particle Swarm Optimizer. In: Proceedings of 2008 IEEE International Conference on Multimedia & Expo, Hannover, Germany, pp. 241–244. IEEE Computer Society Press, Los Alamitos (2008)
6. Lai, C.C., Huang, H.-C., Tsai, C.-C.: Image Watermarking Scheme Using Singular Value Decomposition and Micro-genetic Algorithm. In: Proceedings of 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Harbin, China, pp. 469–472. IEEE Computer Society Press, Los Alamitos (2008)
7. Qi, X., Bialkowski, S., Brimley, G.: An Adaptive QIM-and SVD-Based Digital Image Watermarking Scheme in the Wavelet Domain. In: Proceedings of 2008 IEEE International Conference on Image Processing, California, U.S.A, pp. 421–424. IEEE Computer Society Press, Los Alamitos (2008)
8. Eberhart, R., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
9. Kennedy, J.: The Particle Swarm:Social Adaptation of Knowledge. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation, Indianapolis, USA, pp. 303–308. IEEE Computer Society Press, Los Alamitos (1997)
10. Delaigle, J.F., Devleeschouwer, C., Macq, B., Langendijk, I.: Human Visual System Features Enabling Watermarking. In: Proceedings of 2002 IEEE International Conference on Multimedia & Expo, Lusanne, Switzerland, pp. 489–492. IEEE Computer Society Press, Los Alamitos (2002)
11. Watson, B.: DCT Quantization Matrices Visually Optimized for Individual Images. SPIE: Human Vision, Visual Processing and Digital DisplayIV 1913, 202–216 (1993)
12. Xiangyang, W., Hong, Z., Yongrui, C.: New Adaptive Digital Audio Watermarking Based on DWT and DCT. Mini-Micro Systems 27, 316–319 (in Chinese) (2006)

Defending against the Pirate Evolution Attack

Hongxia Jin and Jeffrey Lotspiech

IBM Almaden Research Center
San Jose, CA 95120
jin@us.ibm.com, lots@almaden.ibm.com

Abstract. A trace and revoke scheme is an encryption scheme for secure content distribution so that only authorized users can access the copyrighted content. When a clone device is recovered, the "trace" component detects the pirate users that have compromised the secret keys in their devices and participated in the construction of the clone device. The "revoke" component excludes the pirate users from accessing the future content. The state-of-art trace-revoke scheme is the very efficient subset difference based NNL scheme [11] which is also deployed in AACS [1], the industry new content protection standard for high definition DVDs. While its revocation and tracing are both very efficient, as pointed out by Kiayias and Pehlivanoglu from Crypto 2007, in its deployment NNL scheme may suffer from a new attack called *pirate evolution attack*. In this attack attackers reveal the compromised secret keys to the clone decoder very slowly through a number of generations of pirate decoders that will take long time to disable them all. They showed in a system with N users, the attacker can produce up to $t * logN$ generations of pirate decoders given t sets of keys. In AACS context, that means a pirate can produce more than 300 generations of decoders by compromising only 10 devices. If this happens, it will indeed be a nightmare.

In this paper we are interested in practical solutions that can defend well against the pirate evolution attack in practice. In particular we devise an easy and efficient approach for the subset difference based NNL scheme [11] to defend well against the potential pirate evolution attack. Indeed it takes as small as 2 generations to detect and disable a traitor in a coalition. This can be achieved by only negligibly increasing the cipher text header size in an application like AACS. The simplicity, efficiency and practicality of our approach has made AACS to adopt it to defend against the pirate evolution attack.

Keywords: Traitor Tracing, Broadcast Encryption, Pirate Evolution Attack.

1 Introduction

In this paper we are concerned with content protection for copyrighted materials when distributed to large number of users (receivers or devices), for example, a system for distributing physical media, like DVDs. Broadcast encryption schemes have been used to enable a set of privileged users to access content but exclude another set of revoked users from accessing it. In a broadcast encryption system, each user is assigned a set of secret keys, sometimes termed as *device keys*. The broadcaster broadcasts *header* information and *body* to a group of users. The *header* carries information of a session key, with which the *body* is encrypted. To be more concrete, a session key (sometimes termed a *media key* K) is indirectly used to encrypt the content in the body. The header contains a structure sometimes called a MKB (media key block), which is basically the media key encrypted with secret device keys from privileged users and only privileged users. It may contain some random string (garbage) encrypted by device keys from revoked users. In this way, each privileged user can use his secret device keys to process the MKB differently but get the same valid media key. In contrast, all the revoked users will process the MKB and get garbage.

Traitor tracing is another important technology used in content protection to detect the non-compliant users (denoted as *traitors*) who are involved in pirate attacks. A particular pirate attack in a broadcast encryption scheme is when some of the legitimate users circumvent the devices and extract secret device keys out of the boxes. They collude to build a pirate decoder using the extracted device keys and sell it to illegitimate users for commercial interest. For example, pirate decoders enable illegitimate users to make permanent copies of DVD movies that they rented. When a pirate decoder is found, a traitor tracing scheme detects the traitorous devices' keys inside the clone decoder.

When future content is distributed, the license agency will construct and distribute new MKBs together with the new content. In the new MKBs, those newly compromised device keys detected by the "tracing" component will be revoked. As a result, the clone decoder who has the newly revoked device keys in it will not be able to decrypt the new MKB successfully and play back new content. The clone device and all the guilty device keys in it are therefore "revoked" by the new MKB. Figure 1 shows the high level architecture of a content protection system using a trace-revoke scheme. While step 0 is only done once, step 1, 2 and 3 (trace and revoke) are repeated throughout the lifetime of a content protection system.

Note that the MKB is distributed together with the content, for example, on a DVD. This enables revocation to happen in off line mode which is essential for protecting content on consumer electronics devices.

In the literature broadcast encryption was first introduced by [3]. Both symmetric-key-based [11] and asymmetric-key-based broadcast encryption schemes [4] were studied further. Traitor tracing for the clone attack has also been studied extensively [6,8,9] since its introduction in [5]. Furthermore, trace and revoke schemes that combine these two functionalities have been studied in [7,11,12] for the pirate decoder attack. A trace-revoke scheme on another type of attack, namely the anonymous attack, has also appeared in [10].

The current state-of-art symmetric-key-based broadcast encryption scheme, the subset difference based "NNL scheme" [11], is adopted in the new industry content protection standard "Advanced Access Content System", AACS in short [1], for next generation high definition DVDs.

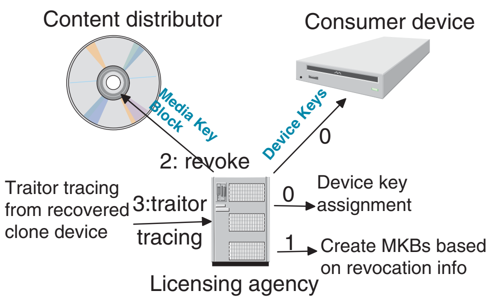


Fig. 1. A content protection system using a trace-revoke scheme

1.1 Trace-Revoke in NNL Scheme [11] for Clone Attack

Naor, et. al [11] present a broadcast encryption framework using subset covers. In this section we will briefly overview its revocation and tracing mechanism. Let $\mathcal{D}$ be the set of devices and $\mathcal{K}$ be the set of device keys. Every device $d \in \mathcal{D}$ owns a subset of keys, denoted by $\mathcal{K}_d$. Similarly, associated with every key $k \in \mathcal{K}$ is a set of users $\mathcal{D}_k = \{d \in \mathcal{D} : k \in \mathcal{K}_d\}$.

Suppose we want to broadcast some media M, which, for all intent and purpose, is a binary string. We would like to encrypt M in such a way that a set of legitimate devices $L \subseteq \mathcal{D}$ is able to decrypt and view the media. The first step is to encrypt M with some media key K. We will use the term *key* without a qualifier to refer to device keys. We then find a subset of device keys C such that all legitimate devices are *covered*. That is, C is chosen such that $\bigcap_{k \in C} \mathcal{D}_k = L$. Now, for every $k \in C$ we separately encrypt the media key, giving us $E_k(K)$. These items together $\langle E_{k_1}(K), E_{k_2}(K), \ldots, E_{k_{|C|}}(K)\rangle$ are referred to as a *media key block (MKB)*. Every device $d \in L$ will own a key used in the MKB and every device $r \in \mathcal{D}/L$ will own none. Hence, they cannot recover the content.

When a clone decoder built from compromised device keys is discovered and brought to the forensic lab, the goal of the NNL tracing algorithm is either identify a traitor by detecting its compromised device keys, or create a MKB that the clone device cannot decrypt (i.e., the clone decoder is disabled). In a black box tracing algorithm, the only means to diagnosis traitors is to submit tests, termed as *forensic MKBs*, to the clone and observe its response. In this context, a subset cover is also called a *Frontier*. When constructing a forensic MKB at frontier $\mathcal{F}$, we intentionally *enable* certain keys by using them to encrypt a valid media key and *disable* certain keys (not necessary traitorous) by encrypting a random bit string instead of the media key. The tracing agency feeds encrypted content together with a forensic MKB to the clone. Based on the keys inside the clone, the clone may or may not decrypt/play the content. If all of the keys in a clone are enabled then it will play. If none of the keys are enabled then it cannot play. If some keys are enabled and some are not, the clone will play with some probability. Suppose we construct forensic MKBs that enable a subset T keys and disable keys in $\mathcal{F}/T$. We feed this type of MKBs to the clone multiple times. Let p_T be the probability that the clone plays. Since the clone is assumed stateless, NNL treats the response of the clone as a Bernoulli random variable. If the probability of playing two different MKBs, (enabling T and T' respectively), are not equal then it must be case that the clone owns a key in the exclusive-or of T and T'. This is the essence of the *subset tracing procedure* that can identify a key in $\mathcal{F}$ owned by the clone. The overall clone tracing relies on two things:

1. *Bifurcation property*: for every key $k \in \mathcal{K}$ such that $|\mathcal{D}_k| > 1$, there exists keys k_1 and k_2 such that $\mathcal{D}_{k_1} \cup \mathcal{D}_{k_2} = \mathcal{D}_k$ and $D_{k_1} \cap D_{k_2} = \emptyset$. With this, we can replace k with k_1 and k_2 and still cover the same set of devices.
2. *subset tracing procedure*: finds at least one key in current frontier $\mathcal{F}$ owned by the clone device. This is used as a subroutine in the clone tracing algorithm.

The tracing algorithm starts from an initial frontier $\mathcal{F}$ and proceeds by repeatedly using the subset tracing procedure to identify a compromised key $k \in \mathcal{F}$, removing it, and adding to $\mathcal{F}$ k_1 and k_2 satisfying the bifurcation property. If $|\mathcal{D}_k| = 1$ then the single device in $\mathcal{D}_k$ is a traitor. This process is reiterated until the detected compromised key is at the lowest level of the frontier and cannot split further, or the clone box is unable to play the MKB associated with $\mathcal{F}$. Figure 2 illustrates this process. The efficiency of this type of tracing is mainly measured by how many forensic MKBs are totally needed to complete tracing. That number for NNL tracing is $O(T^3 logT)$ where T is the number of colluding traitors involved in clone attack.

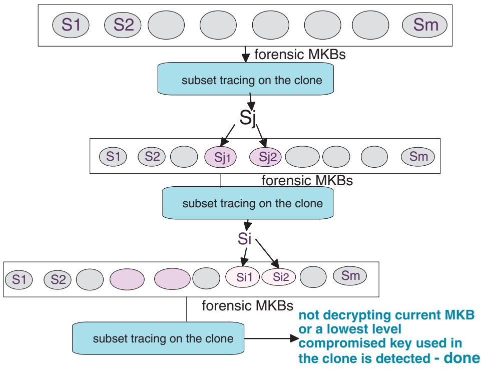

S1
S2
Sm
forensic MKBs
subset tracing on the clone
Sj
S1
S2
Sj1
Sj2
Sm
forensic MKBs
subset tracing on the clone
Si
S1
S2
Si1
Si2
Sm
forensic MKBs
subset tracing on the clone
not decrypting current MKB or a lowest level compromised key used in the clone is detected - done

Fig. 2. Dynamic Clone Tracing algorithm using forensic MKBs (in the lab)

1.2 Pirate Evolution Attack

The pirate evolution attack is a newly described attack in [2] that particularly targets on trace-and-revoke schemes. In this attack, after the attacker succeeds in circumventing devices and extracts their device keys, they reveal those keys slowly to the decoders. They will build and release a first pirate decoder. After this decoder and the device keys in it are detected by the tracing agency and then disabled by the new updated MKB, they roll out a second version of the pirate decoder. This step is repeated. Using this evolving strategy, the pirates will only evolve a new version of the decoder after the current generation of the decoder is disabled by an updated MKB.

In particular they studied the pirate evolution attack on the NNL scheme [11]. The t attackers compromised t set of device keys. But each set of keys are released to clones one at a time. In other words, the keys are drizzled. Only when the revealed key gets revoked will the attacker release the next key in the next clone. In fact, it takes the entire height of the tree number of iterations (i.e., $logN$) to identify the hacked device (traitor) at a leaf node. So in general it could take up to $t * logN$ generations of clone decoders to detect and disable t traitors. In a system like AACS that supports billion devices, there are 30 levels in the tree. A pirate can produce more than 300 pirate decoder generations by compromising only 10 devices. It could take 30 generations (iterations)' battle with attackers before the license agency can revoke all the keys one guilty device revealed. In reality this converts to years of time to stop one hacked device. This is certainly unacceptable. It is highly desirable to be able to quickly respond to this attack.

In this paper we will present our efficient approach that can make the NNL scheme [11] defend well against the pirate evolution attack without sacrificing much of the efficiency of the scheme. We know forensic MKBs intentionally choose which keys to enable/disable; they are not operational and only serve for forensic purpose. In our approach to defend against pirate evolution attack, we introduce a special type of forensic MKBs that are also operational. We will illustrate how to construct such type of MKBs and show why this enables the trace-revoke scheme [11] to defend against pirate evolution attacks. In AACS context it will take as small as 2 generations to detect and disable traitors.

The rest of the paper is organized as follows. In Section 2 we will present our approach to defend well against pirate evolution attack using a special type of MKBs. In section 2.1, we will illustrate how to construct the special type of MKBs for the efficient subset-difference method in NNL scheme. In Section 3, we will analyze the efficiency of our approach and the overhead incurred in our approach. We will conclude in Section 4.

2 Defending against Pirate Evolution Attack

In an evolution attack, the attackers exploit the fact that each device owns a set of device keys, one at each level, to release only one key from each compromised device at a time. A new version of the clone is not generated until its previous version (keys) are disabled. That evolving strategy is important to slow down the license agency to disable/revoke a set of traitors.

Our idea is simple. In order to shut down this attack quickly, we will disable the attackers' possibility of releasing their keys one at a time. We will force them to use their device keys at a deeper level when they create each of their evolving clone decoders. In this way we can reduce the number of clone decoder generations that the attackers can produce from the sets of device keys they compromised. In our approach, the tracing relies on two things:

1. *dynamic clone tracing*: given a clone that was generated when the lowest level frontier is F, finds at least one key k in F owned by the clone device. This is the original NNL dynamic tracing algorithm as illustrated in Figure 2.
2. *split i levels down*: for every key $k \in \mathcal{K}$ such that $|\mathcal{D}_k| > 1$, there exists keys $k_1, k_2, \ldots, k_{2^i}$ such that $\mathcal{D}_{k_1} \cup \mathcal{D}_{k_2} \cdots \cup \mathcal{D}_{k_{2i}} = \mathcal{D}_k$ and $D_{k_1} \cap D_{k_2} \cdots \cap D_{k_{2i}} = \emptyset$. With this, we can replace k with $k_1, k_2, \ldots, k_{2^i}$ at the *ith* level down and still cover the same set of devices.

Similar to the original subset-based trace-revoke scheme, our tracing algorithm also starts from an initial frontier $\mathcal{F}$. Given a recovered clone decoder that builds upon current frontier $\mathcal{F}$, the algorithm identifies a compromised key $k \in \mathcal{F}$ in current clone. This is done at a forensic lab using forensic MKBs exactly as shown in Figure 2. Once a compromised key $k \in \mathcal{F}$ is detected, our algorithm removes it, and adds to $\mathcal{F}$ $k_1, k_2, \ldots, k_{2^i}$. If $|\mathcal{D}_k| = 1$ then the single device in $\mathcal{D}_k$ is a traitor. This process is reiterated with the new frontier $\mathcal{F}$ until it reaches the leaf level and no more clone decoder can be built, or the attackers are not able to play the MKB associated with $\mathcal{F}$. Figure 3 illustrates this process.

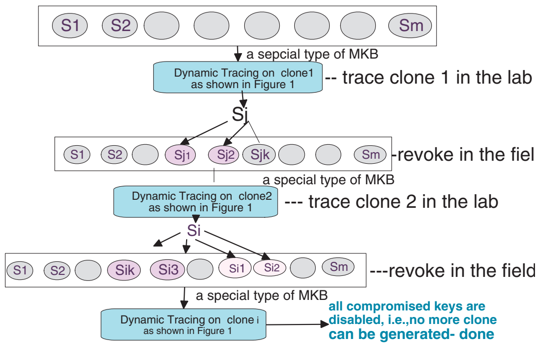

S1
S2
Sm
a sepcial type of MKB
Dynamic Tracing on clone1 as shown in Figure 1
-- trace clone 1 in the lab
Sj
S1
S2
Sj1
Sj2
Sjk
Sm
-revoke in the fiel
a special type of MKB
Dynamic Tracing on clone2 as shown in Figure 1
--- trace clone 2 in the lab
Si
S1
S2
Sik
Si3
Si1
Si2
Sm
---revoke in the field
a special type of MKB
Dynamic Tracing on clone i as shown in Figure 1
all compromised keys are disabled, i.e.,no more clone can be generated- done

Fig. 3. Our Trace-revoke Approach for Evolution Attacks

There are some subtleties here. First of all, when a traitorous key in the current generation of the clone is detected, it must always be possible to find keys $k_1, k_2, \ldots, k_{2^i}$ such that $\mathcal{D}_{k_1} \cup \mathcal{D}_{k_2} \cdots \cup \mathcal{D}_{k_{2i}} = \mathcal{D}_k$ and $D_{k_1} \cap D_{k_2} \cdots \cap D_{k_{2i}} = \emptyset$. The choice of i depends on how quickly one wants to shut down the attack (refer to Theorem 1 in Section 3). Second, the original clone tracing algorithm shown in Figure 2 is now only a procedure of our trace-revoke approach. Keep in mind the original clone tracing occurs only when a clone decoder is found and brought to a forensic lab where forensic MKBs can be created to probe the clone. More concretely, for a frontier $\mathcal{F}$, a forensic MKB is constructed in a way that intentionally chooses to enable some keys in $\mathcal{F}$ and disable other keys in $\mathcal{F}$. It does not necessary enable all compliant device keys and disable all non-compliant device keys. For that reason a forensic MKB is almost always non-operational. However, a pirate evolution attack is an on-going evolving attack in the field.

Our goal is to reduce the number of generations of clone decoders the attackers can build given the sets of compromised device keys. In order to achieve this goal, we need to build a type of MKB that is both operational and forensic.

Creating this type of special MKB is very different from creating a normal operational MKB. There the creation of the MKB is driven by revocations, and the algorithm generates the fewest possible subsets that cover the nodes that are not revoked. In our special MKBs that also serve the forensic purpose we intentionally want to cover some of the enabled nodes with a larger number of subsets at a much lower level, so that each subset covers far fewer devices. Note that each device needs to use one of its keys enabled in the MKB in order to process the MKB. In a normal MKB, with fewer subsets, an attacker uses a key shared by many devices. In our special MKB, the attacker is forced to use a key shared by far fewer devices, so he has to identify himself more quickly.

In order to create a special type of MKB for our tracing, we will start from an operational MKB that uses the fewest subsets to enable/disable the right sets of devices, in other words, enable all compliant devices and disable all non-compliant devices. We will then recursively perform the splitting operation to the next level based on the bifurcation property until we reach the desired ith level down. It is easy to see the recursive splitting operations based on bifurcation property guarantee that for a key k it is always possible to find keys $k_1, k_2, \ldots, k_{2^i}$ such that $\mathcal{D}_{k_1} \cup \mathcal{D}_{k_2} \cdots \cup \mathcal{D}_{k_{2^i}} = \mathcal{D}_k$ and $D_{k_1} \cap D_{k_2} \cdots \cap D_{k_{2^i}} = \emptyset$.

The recursive approach also guarantees that it is possible to construct MKBs that are at a much deeper level but are still operational. This works for two reasons. First of all, the split subsets after splitting operation enable the same set of devices as the original subset. Second, the splitting operations always make progress to the next level, so the recursion will eventually terminate. Therefore it is always possible to split a subset any levels down. Next we will show how to construct this special type of MKB for the subset-difference NNL scheme.

2.1 Creating Special Type of MKBs for Subset-Difference Method[11]

Each device in NNL scheme is associated with a leaf of the tree. Each node (representing a subset) in the tree is associated with a key. In a subset-difference NNL tree, a device is given every key in the tree except the keys between its leaf and the root. In a subset-difference method, a subset is not a simple subset, it is actually the difference between two simple subsets, thus called a subset-difference. As shown in Figure 4, a subset difference S_{ij} is defined as all the leaf nodes in the subtree rooted at node V_i but not in the subtree rooted at V_j. The subset-difference method contains a mechanism to find a subset cover given a revoked node list R. A sample subset cover is illustrated in Figure 5. The MKB for the subset-difference method represents the revocation information by the subset differences in the cover set. Each subset difference represents the set V_i minus V_j.

Furthermore, NNL does not just have a single system of keys; it has literally billions of systems of keys. In fact, every subtree in the master tree defines a completely independent system of keys. For example, one level down from the

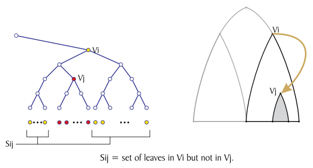


Fig. 4. Subset Difference Definition

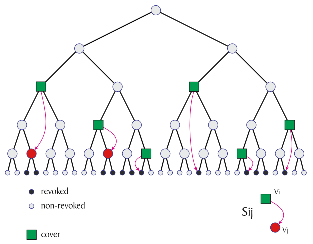


Fig. 5. Subset Cover of non-revoked devices

root of the tree, there are two subtrees, each with one billion devices in them. Each of these subtrees has its own system of keys, in addition to the system of keys in the master tree. Likewise, the next level down there are four subtrees of size one-half billion devices, and each have independent systems of keys, and so on. Eventually, at the bottom, every pair of sibling leaf devices also defines its own system of keys. A device has a whole set of device keys in every subtree that it belongs to, 31 independent subtree systems per device if there are 31 levels in the tree. Of course NNL tree needs a mechanism (refer to NNL [11])so that the device does not have to store billions of keys. Now, with all these different key systems based on subtrees, NNL tree provides a mechanism to revoke all possible combinations of revoked devices. On average, there are 1.28 encryptions per revocation in NNL tree. It has the most concise expression of revocations.

In our approach, to create the special type of MKB, we must subdivide subtrees deep down but without adding any revocations. We will first show how to split a subset and go down just one level. There are three primitive splitting operations.

Splitting a subset without any revocation, see Figure 6
In this case, the subset difference excludes the left most leaf, but it is not a real revocation. It can be split into two mutually excluding subsets. The split subsets do not exclude the left most leaf. The original subset is replaced by two new subsets. The new subsets enable/disable the exact same set of devices.

Splitting a subset with an excluded child, see Figure 7
The enabled child is split into two subsets, each with an excluded child. Again these two subsets are mutually excluding. The original excluded child disappears from the list of subsets, unless some previously existing subset enables it.

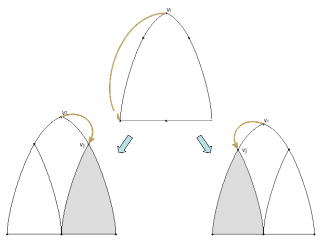


Fig. 6. Splitting a subset without any revocation

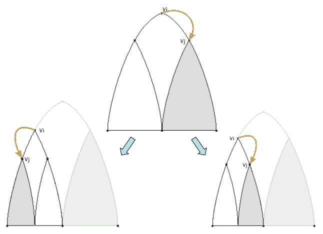


Fig. 7. Splitting a subset with an excluded child

Splitting a subset with an excluded descendant of a child
As illustrated in Figure 8, one subset enables the included child. The other subset enables the child with an excluded descendant. The descendant is still excluded; it may be a child in the new subset, or a descendant of a child.

Now that we have shown how to split a subset to one level down, to descend further, it is necessary to apply the same operations recursively. In general, because the special MKB has to be fully operational, we have to start with a cover set that incorporates the existing revocations, and then subdivide the existing subset differences using the above described splitting operations recursively so that the (suspect) subtree is covered by a large number of subset differences.

We will use an example to illustrate how we can expand and subdivide the existing subset differences into more subset differences. In a tree of height 22 with no revocations, Figure 9 (part 1) shows how we can expand the fourth eighth of the tree with more subset differences. Using the above described primitive splitting operations, we first split the height 22 tree in two; then the first half of it in two again, then the second quarter in two. The fourth eighth of the tree will be of height 19. Again, by recursively performing splitting operations we finely subdivide the fourth eighth into $2^{10} = 1024$ small subtrees of uniform height at $9th$ level. The 1024 subtrees are enlarged and shown in part 2 of the Figure 9.

During splitting operation, to enable devices, the subset differences covering the small subtrees are encoded in mutually excluding sibling pairs, as shown in Figure 6 and re-shown as part 3 of the Figure 9. The original subset difference, which covered all the devices in the height 22 subtree, is replaced by 1027 new subset differences. Each of the 1024 subset differences will cover subtrees of

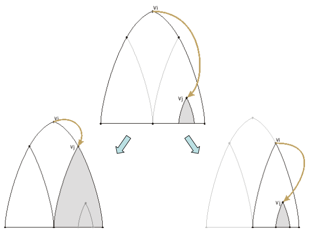


Fig. 8. Splitting a subset with an excluded descendant of a child

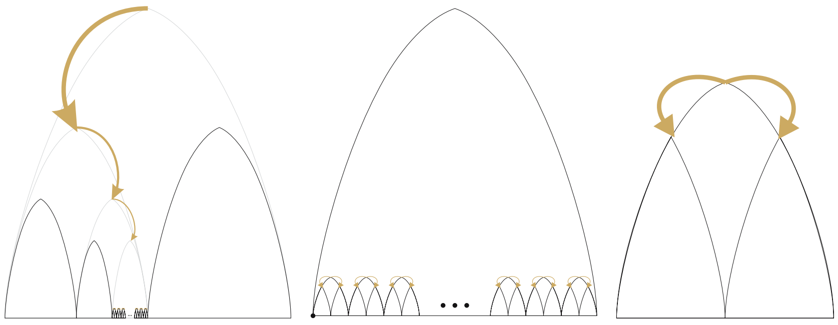


Fig. 9. Splitting a subset into multiple subsets without adding revocations

height $19 - 10 = 9$, each containing only $2^9 = 512$ devices. As a result, if the attacker reveals a key that can be used to process this MKB, it will be a key shared with only 512 devices, much more identifying than a key shared by 4 million devices with the original MKB.

3 Discussion

As one can imagine, in our approach expanding into more subsets will increase the MKB size which is used to measure the efficiency of the trace-revoke scheme. There is a tradeoff between the number of iterations the license agency is willing to spend and the MKB size. This tradeoff is shown in the following theorem.

Theorem 1. *If the tracing algorithm wants to detect and disable a traitor in only b generations of pirate decoder, then the MKB size is $O(N^{1/b})$.*

Proof. Suppose every time when a primitive splitting function is performed, a subset is split into j subsets. We know the devices/users are arranged as the leaf nodes. The height of the subset structure is therefore $log_j N$. If the tracing wants to identify a traitor at the leaf node in b generations, it means for each generation of the special MKB (clone decoder), the splitting dives $i = \frac{\log_j N}{b}$

levels down on average. In other words, the primitive splitting function needs to perform $\frac{\log_j N}{b}$ times. So a subset is expanded into $j^{\frac{\log_j N}{b}} = N^{1/b}$ subsets. Therefore the MKB size is $O(N^{1/b})$.

The above theorem shows a tradeoff between immunity to evolution attack and revocation efficiency. Different applications may need different tradeoffs between these two aspects. Based on the above theorem, in order to reduce the number of generations of clone decoder to 2, our approach will dive 11 levels down each time to build and distribute a special type of MKB that is of size $O(\sqrt{N})$. However in practice, that MKB size does not convert to big numbers. For example, in a tree of height 22 an optimally-sized MKBs (i.e. using the fewest number of subsets based on NNL scheme) are less than 16KB. Using our approach we first create multiple subsets at 11th level then again multiple subsets at the leaf (22th) level. That dive would increase the size of an MKB by only 46KB. In AACS, disc replicators allocate 1MB space for the MKB, and even that is negligible overhead on a blue-ray disc that can store at least 25GB. There is plenty of room to store our special type of MKBs. Therefore our approach is very feasible for AACS to quickly shut down the pirate evolution attack. For this reason, AACS adopts our approach to defend against the evolution attack.

Our approach is applicable to any subset-based broadcast encryption scheme. At a high level, a subset-based scheme organizes the devices into many overlapping subsets, with each subset associated with a key. Each device belongs to many different subsets and knows the key for every subset it is a member of. In face of pirate evolution attack, from the minimal sets of subsets that cover all and only innocent devices, one needs to intentionally split a subset into multiple subsets with each subset covering fewer devices and construct a special type of MKB that is both operational and forensic. This type of MKB will force the attackers to reveal keys that are much more identifying.

4 Conclusion

In this paper we are concerned with the pirate evolution attack (introduced in [2]) on trace-revoke schemes where, during deployment, the pirate users reveal the compromised keys slowly into pirate decoders trying to stay ahead of the revocation. In this paper we are interested in a practical approach that can defend well against this attack in practice.

In particular, we are interested in the subset-difference-based NNL scheme that is very efficient in revocation and is deployed by AACS in practice, but was pointed out in [2] that it can suffer very badly from the evolution attack. We devise an easy and efficient approach for the NNL scheme to defend well against the evolution attack while maintaining very reasonable revocation efficiency. For example in the AACS context our approach only takes as small as 2 iterations to detect and disable a traitor. And achieving this only increases the ciphertext size by a negligible 46KB in a 25GB disc space. The simplicity, efficiency and practicality of our approach caused its adoption by AACS to use to defend against the evolution attack.

As future work we would like to formalize our approach more so as to guide the future design of trace-revoke schemes that can achieve balances between immunity to evolution attack and revocation efficiency.

References

1. http://www.aacsla.com/specifications
2. Kiayias, A., Pehlivanoglu, S.: Pirate evolution: How to make the most of your traitor keys. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 448–465. Springer, Heidelberg (2007)
3. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
4. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
5. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
6. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing traitors. IEEE Transactions on Information Theory 46, 893–910 (2000)
7. Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 1–20. Springer, Heidelberg (2001)
8. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
9. Staddon, J.N., Stinson, D.R., Wei, R.: Combinatorial properties of frameproof and traceability codes. IEEE Transactions on Information Theory 47, 1042–1049 (2001)
10. Jin, H., Lotspiech, J.: Renewable traitor tracing: A trace-revoke-trace system for anonymous attack. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 563–577. Springer, Heidelberg (2007)
11. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
12. Boneh, D., Waters, B.: A collusion resistant broadcast, trace and revoke system. ACM Communication and Computer Security, 211–220 (2006)

Security Specification for Conversion Technologies of Heterogeneous DRM Systems

Heasuk Jo, Woongryul Jeon, Yunho Lee, Seungjoo Kim, and Dongho Won*

Information Security Group,
Sungkyunkwan University,
300 Chunchun-dong, Suwon, Gyeonggi-do, 440-746, Korea
{hsjo,wrjeon,leeyh,skim,dhwon}@security.re.kr
http://www.security.re.kr

Abstract. Digital Right Management (DRM) can be used to prohibit illegal reproduction, and redistribution of digital content, to protect copyrights. However, current DRM systems are incompatible and lack of interoperability which exchange of data, different platform, designed and protected by different content providers. To overcome these drawbacks, three ways of interoperability are full-formation interoperability, connected interoperability, configuration-driven interoperability, allowing consumers to use the purchased content in their equipments of choice. In this paper, we study on the security specification of configuration-driven interoperability for heterogeneous DRM systems, using the Common Criteria. Then, we study security boundary, security environment, security objectives, and rationale of an CTHDS_PP(Conversion Technologies of Heterogeneous DRM Systems Protection Profile) to find important security features. The CTHDS_PP gives a discussion covered the current security problems to conversion technologies and lists threats to solve those problems. Moreover, this CTHDS_PP can be used for potential developers and system integrators, and reviewed and assessed by evaluators.

Keywords: Digital rights management(DRM), Common Criteria(CC), Protection Profile(PP), Interoperability.

1 Introduction

Nowadays, wireless mobile device not only be everywhere but also owned by the majority of the people. Business and technology trends indicate an exponential growth in the potential sales of content through mobile devices. The protection of content used to prohibit illegal reproduction or redistribution of digital content, is achieved by technology that protects the copyrights of digital content. This is called Digital Rights Management (DRM) system. Various DRM systems exist, e.g., Windows Media DRM[1], Apple iTunes' Fairplay[2], the Open Mobile Alliance's(OMA) DRM scheme[12], PachyDRM[3], Secure Digital Container(SDC)[4], etc.

* Corresponding author.

However, the incompatibility and the lack of interoperability between current DRM technologies heavily restricts the rights of end-users, such as private reproduction. For example, once any contents are made under one specific DRM system, those can not be used under the other DRM systems because system and data of different type designed and produced by a different providers.

Therefore, many researchers study interoperability technology for heterogeneous DRM systems such as Open DRM platforms[19], EXIM(EXport and IMport)[9], Networked Environment for Media Orchestration(NEMO)[18], etc. Full interoperability can be categorized into three ways[17][16]:

- $Full-formation$ *interoperability*: All protected content conforms to some globally standardized format.
- *Connected interoperability*: On-line third parties are used to translate operations from one DRM regime to another.
- $Configuration-driven$ *interoperability*: End-user device can acquire the ability to process content protected by any DRM regime by downloading appropriate "tools".

In this paper, we focus on configuration-driven interoperability such as Motion Picture Exports Group(MPEG)[20] and EXIM[9][10]. And we will describe using EXIM approached configuration-driven interoperability. Recently, EXIM is planing the commercial service in South Korea. This paper proposes the security specification for conversion technologies of heterogeneous DRM systems, using standardized CC methodology, syntax, and notation. A security specification for configuration-driven interoperability do not exist. As a result, EXIM needs acceptable international terms and standards for security protection, such as by employing a Protection Profile. This protection profile is hereafter referred to as the CTHDS_PP(Conversion Technologies of Heterogeneous DRM Protection Profile). The CTHDS_PP is a stated security environment that includes assumptions, potential threats, organizational security policies, and security objectives to uphold counter potential threats, enforce organizational security policies, and assumptions[5].

The remainder of this paper is organized as follows. Section 2 reviews related works, Section 3 presents the security specification for CTHDS_PP. Section 4 provides an rationale of the CTHDS_PP. Finally, section 5 concludes the paper.

2 Related Works

2.1 Overview of the Common Criteria

The Common Criteria (CC)[5] is an internationally approved set of security standards, providing a clear and reliable evaluation of the security capabilities of Information Technology products. The CC is useful as a guide for the development of products or systems with IT security functions and for the procurement

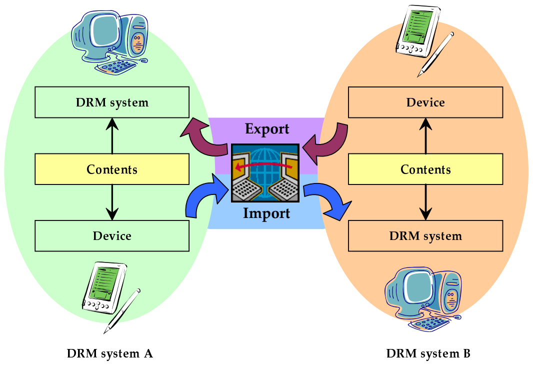


Fig. 1. Basic EXIM Architecture Model

of commercial products and systems with such functions. This is the belief behind the ISO/IEC standard 15408, CC for IT Security Evaluations[5][6][7]. It represents the outcome of efforts to develop criteria for evaluation of IT security, and is widely applicable internationally. The CC defines a set of IT requirements of known validity, which can be used in establishing security requirements for prospective products and systems [5]. The CC also defines the Protection Profile (PP) construct, which allows prospective consumers or developers to create standardized sets of security requirements. Moreover, consumers state their security functional and assurance requirements in a PP[8].

These profiles form the basis for Common Criteria evaluation. By listing required security features for product families, the Common Criteria allows products to state conformity to a relevant PP. Follows are TOE steps to derive IT security requirements. During Common Criteria evaluation, the product is tested against a specific protection profile, providing reliable verification of the security capabilities of the product [13]. A PP is intended to be reusable and to define requirements which are known to be useful and effective in meeting the identified objectives [5][14][15].

2.2 EXIM Based Configuration-Driven Interoperability

DRM provides a method to prevent digital content from being copied and distributed, and manages the circulation and use of digital content[12]. However, there is no standard DRM system, though many content providers create DRM systems. These digital content providers operate in a competitive market. As a result, each content provider creates its own DRM system, and their DRM systems are not interoperable with each other, heterogeneous DRM systems. This is a common problem between provider and user. The problem is that the providers cannot sell their digital content to users who do not have a suitable device to use their content, and users can only use certain digital content because each format is different. The user needs to purchase same content two or more times if he wants to play digital content on his/her various devices. EXIM (EXport and IMport)[10] is a technique developed by the ETRI(Electronics and Telecommunications Research Institute, in South Korea), to solve the DRM systems' tack of interoperability based configuration-driven interoperability. EXIM is not a kind of DRM system, but a technique that guarantees secure translation of digital content among the different DRM systems. The architecture of EXIM is illustrated in Fig. 1.

For example, in Fig. 1, contents in a DRM system can be freely played a device having a DRM system A. To play a content of the DRM system A to a device of DRM system B, DRM system or device in DRM system A send a converted a EXIM format content through EXIM to the device of DRM system B. After the device of the DRM system B convert the EXIM format content into a content of the DRM system B through EXIM, then can freely play the content.

3 The Security Specification for CTHDS_PP

This section consists of TOE (Target Of Evaluation) description, identification of the TOE, TOE security Environment, and security object of the CTHDS_PP. It is designed to provide potential threats, organizational security policies, assumptions, and security objectives of the CTHDS_PP to compatible services. We regards that as now, conversion technologies for heterogeneous DRM system is only one, EXIM, and so we study CTHDS_PP based on the EXIM.

3.1 TOE Description

The property for security of TOE is content, TOE and important data (security and TOE security function data) in TOE, when each other DRM device transfers content. In addition, TOE of CTHDS_PP protects the content user rights object and user information from various threats. The Fig. 2 illustrates CTHDS scenarios for content distribution.

TOE Environment

- *Content Mall*: This is similar to a content provider. A user connects to a content mall through source DRM agent(it is explained in a next subsection) and then selects her preferred content. After payment of chosen content by the user, the content mall sends user's Rights Object about the content to a content storage server.
- *DRM Clearing House*: The DRM clearing house is an entity that assigns permissions and control to DRM content, and generated Rights Objects. After the DRM clearing house receives a Right Object Token from the source DRM agent, the DRM clearing house issues a Rights Object information to

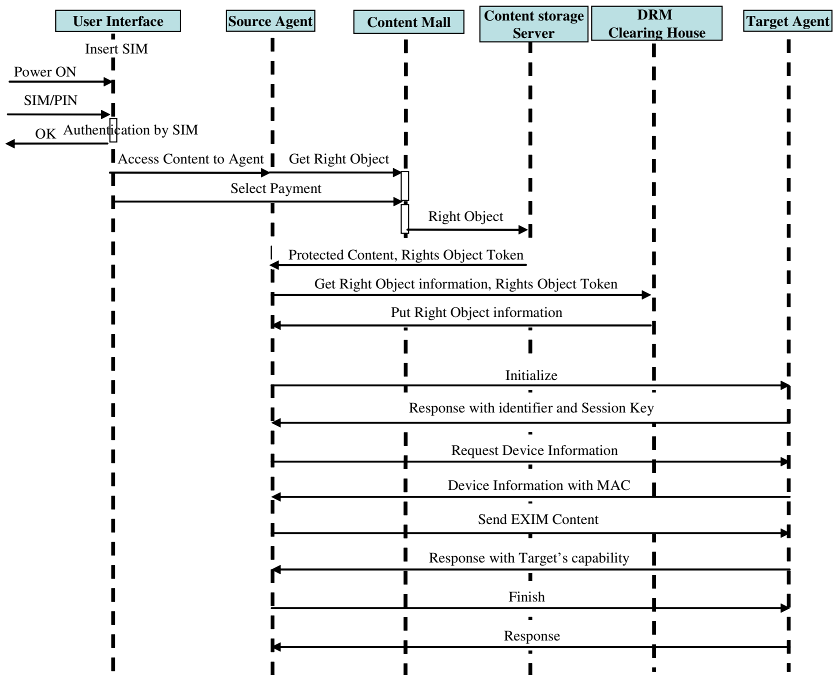


Fig. 2. CTHDS Scenarios Based on EXIM[10]

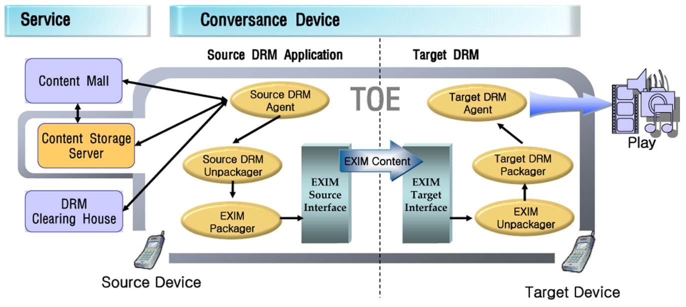


Fig. 3. TOE

the source DRM agent. The Rights Object information is the "Right status" of using the content, with an expiration date, number of times accessed, etc.

TOE Components

– *DRM Agent*: In source device or target device, a DRM agent can negotiate the delivery of content, services, and Rights Object with suppliers.

The DRM agent may need to access to the content mall, content storage server, DRM clearing house, DRM unpackager or packager, and EXIM interface.

- *DRM Unpackager or Packager*: A DRM unpackager converts a packaged(encrypted) content into unpackaged(decrypted) content. The DRM packager is its opposite. The DRM unpackager sends the unpackaged DRM content to the EXIM packager or the DRM packager sends the packaged DRM to the target DRM agent.
- *EXIM Packager or Unpackager*: The EXIM packager converts the unpackaged DRM content into the EXIM content format, that consists of the encrypted content is metadata(identifier, title, creator, contributor,etc.), rights expression(valid date, valid times, rights permission, etc.), resource, etc. The EXIM unpackager unpacks the content in EXIM content format.
- *EXIM Interface*: The EXIM source interface exchanges the content of the EXIM content format with the EXIM target interface.
- *Content Storage Server*: The content storage server stores content, and may be located in the content mall or user's device, for example in network storage, a PC, on removable media, etc.

3.2 TOE Security Environment

The TOE security environment describes the security aspects of the environment in which the TOE is intended to be used and the manner in which it is expected to be employed. The TOE security environment consists of threats, organizational security politics, and assumptions of the security of the TOE environment.

Threats. This subsection characterizes potential threats, including accidental and malicious attempts, and any known or presumed threats to the assets against which protection will be required, either by the TOE or by its environment[11]. The potential threats of TOE are as follows, Table 1. Label of Threats started with letter 'T'.

Organizational Security Policies. This subsection relates to the TOE. It includes Information flow control rules, Access control rules, policies regarding the use of encryption, security audit policies and procedures, and Policies regarding use of a standardized IT base. Label of Polices started with letter 'P'. Table 2 shows the organizational security policies.

Assumptions. The assumptions regarding secure usage of the TOE are made when defining the security function requirements and security assumption requirements. Label of Assumption started with letter 'A'. Table 3 shows the assumptions.

Table 1. The Threats

Threat Lable	**Description**
T.P_Replace	Physical Replace: Physical replacement of any component of the TOE.
T.P_Damage	Physical Damage: Physical damage to any part of the TOE.
T.Permit	Access: An attacker may access information or services without having permission from the rightful owner, service mall, or rights issuer.
T.Trans	Transmission to others: After the Source User delivers content to the Target User, the Target User may pass the content to an unauthenticated user.
T.Residual	Residual content in reallocated resources: An authorized user may access content or service that he is not entitled to access, or does not have permission to access, through reallocation of TOE resources. This occurs when the reallocated resources contain residual content or access permissions from another user or process, gained through legitimate access.
T.Illegal_Access	Resource consumption: An authorized user may extremely consume resource, through legitimate access, in Content Mall or DRM Clearing House, which ensure other authorized users do not access information or services.
T.Damage	Damage to other resources: An authorized user may access information or services in a way that result in damage or discloses other information or services.
T.Install	Installation of new functionality: An authorized user may install a malicious program or new functionality into the TOE that has the capability to damage or disclose TOE resources.
T.Intro	Introduction of new functionality: An authorized user may create or introduce new functionality into the TOE that has the capability to damage or disclose TOE resources.
T.Permit	Permission of access: An authorized user may permit access permissions to others that do not have a legitimate need to access the resources.
T.Deny	User deny of service: An authorized user may send the content to a target user, and then the target user may deny having participated in the transaction.
T.Delet	Deletion of audit information: Particular information of the authorized user may be damaged or destroyed. Audit information should only be destroyed in accordance with security policy.
T.Deny_Service	Denial of illegal access: An attacker may deny authorized users legitimate access to services or information e.g. through resource exhaustion, or by causing legitimate users to be blocked out from access.
T.TSF_Block	Access blacked by TSF modification: An attacker may modify TOE security functionality so that subsequently authorized users can no longer access services or information that they are authorized to access.
T.Disguise	Disguise of authorized user using stolen authentication data: An attacker may masquerade as a legitimate user by presenting authentication data of an authorized user.
T.TOE_Error	Insecure TOE after system error: While a source user sends information or content to a target user, if the TOE security state becomes incorrect or inconsistent, following a system error, TOE security functionality may not operate correctly.

Table 1. (*Continued*)

T.TOE_Change	Insecure TOE after TOE changes: Changes to the TOE may disable or corrupt security functionality. Very few systems remain in their original, delivered configuration. Even small maintenance actions may disrupt or disable security.
T.Sys_Error	System Error in TOE: Content, information, or service protected by the TOE may occur due to system error such as user error, hardware errors, transmission error, and failure to allocate adequate external resources.
T.Env_Failure	Environmental failure: Environmental failure, such as human error or failure of software, hardware or power supply, may result in unexpected interruption of TOE operations, causing damage or loss to information or service.
T.Env_Stress	Stress by environment factor: Environmental factors may obstruct TOE services (e.g. electrical power, temperature, humidity).
T.Eavesdrop	Information eavesdropping from TOE: Content or information may be disclosed from the TOE, and an attacker may be able to connect to legal users of the network in a trusted zone.

Table 2. The Organizational Security Policies

Policy Lable	**Description**
P.Audit	Audit : The TOE must audit recognizing, recoding, and storing information related to security relevant activities.
P.S_Update	Update TOE service software : When a device accesses a Content Mall, if EXIM software is updated for DRM service, the device must automatically download the software. This allows the Agent to convert content to another DRM device.
P.Guide	Guide for User : A Content Mall must provide guide lines demonstrating how a user can use service.
P.Sec_Mang	Secure Management: An authenticated manager must manage the TOE with the safe method.
P.Auth_Crypt	Authenticated Cryptographic Algorithm : Using cryptographic algorithms, such as RSA, AES, SHA-1, and DES, must be authenticated by ISO.
P.Ident_Chk	Identity Check : Devices must check the identity each other through trusted Certification Authority(CA).

Table 3. The Assumptions

Assumption Lable	**Description**
A.See_Ch	Secure Channels : Each component is connected through secure channels.
A.User_SIM	User security owner of SIM Card: SIM Card is securely issued to user.
A.Data_Protec	Data Protection: User's right and content are securely encrypted and stored in a content mall, content storage server, and DRM clearing house.
A.SIM_Lose	Lose SIM Card: If a user loses the SIM Card, which is subsequently gained by an attacker, the attacker cannot retrieve the user's private information.

Table 4. The Security Objectives

Name	**Description**
O.Sec_Ch	Secure Channels: TSF must provide secure channels.
O.Issu_SIM	Issue of SIM Card: SIM Card must be securely issued to users.
O.Data_Prot	Data Protection: The TOE must support cryptographic functions in a secure manner.
O.S_Udate	Software Update: The TOE must regularly update DRM service software.
O.Guide	Guide: The TOE must provide guidelines for users consuming the content.
O. Auth_Fail	Authentication fail of Password: TSF must detect repeated authentication failures in the user authentication system.
O.Auth	User Authentication: The TOE must be accessed by authenticated Users.
O.Data_Prot	Data Protection: The TOE must protect exposure, modification, and deletion of the recoded data.
O.Flow_Cont	Flow Control: The TOE must control the flow of unauthenticated data through security policy
O.Install	Installation: The TOE must be installed by unauthenticated software.
O.TSF_Sec	Protect of TSF data: The TOE must protect the recoded TSF data in the TOE from unauthenticated exposure, modification, and delete.
O.Audit	Audit: The TOE must have audit information and digital signature function.
O.Test	Test: TSF must prove the integrity of data(Metadata, Rights, Resource, etc.) in the TOE.
O.Update	Update: The TOE must be reinstalled or updated when becoming aware of the system error.
OE.Phy_Prot	Physical Protection: The TOE must be located in a safe environment
OE.Secu_Manag	Security Function Management: TSF must be restricted within the authenticated manager who has the capability to modify and delete securely.
OE.Sec_Maintain	Security Maintain: When the TOE or network service suffers an unpredicted fault, the TOE must provide identical service to the previous state.

3.3 Security Object

This section provides the security object is traced back to the Security Environment (threats, organizational security policies, assumptions). Each security object is traced back to at least one threat or organizational security policy. In other words, a threat or organizational security policy is addressed entirely by one or more security objectives. Table 4 shows the security objectives.

4 Rationale

This section contains the security objectives rationale. It demonstrates that the CTHDS_PP is complete, correct, consistent, and coherent, with Assurance, Threat, and Policy, and suitable to counter the identified threats to security. In other words, this subsection proves that the security objectives uphold counter all threats, enforce all organizational security policies, and all assumptions. The Table 5 shows mapping between threat, and policy and objectives, assurance.

Table 5. Mapping Security Objectives to Threats, Policies, and Assumptions

Objectives / Threats Policies Assumptions	O.Sec_Ch	O.Issu.SIM	O.Data_Prot	O.S_Udate	O.Guide	O.Auth_Fail	O.Auth	O.Data_Prot	O.Flow_Cont_	O.Install	O.TSF_Sec	O.Audit	O.Test	O.Update	OE.Phy_Prot	OE.Sec_Maintain	OE.Secu_Manag
A.Sec_Ch	*																
A.User_SIM		*															
A.Data_Protec			*														
A.SIM_Lose			*			*	*										
P.Audit												*	*				
P.S_Update				*													
P.Guide					*												
P.sec_Mang																	*
P.Auth_Crypt			*														
P.Ident_Chk								*					*				
T.P_Replace								*									*
T.P_Damage																*	
T.Permit							*										
T.Trans							*										
T.Residual								*									
T.Illegal_Acc									*								
T.Damage									*								
T.Install										*							
T.Permit							*										
T.Intro											*						

Table 5. (*Continued)*

T.Deny								*				*					
T.Delet											*						
T.Deny_Service						*											
T.TSF_Block											*						
T.Disguise							*										
T.Pretend_User							*										
T.TOE_Error													*				
T.TOE_Change														*			
T.Sys_Error								*									
T.Env_Failure													*				
T.Env_Stress															*		
T.Eavesdrop	*																

5 Conclusion

For protection of content, DRM can be used to prohibit illegal reproduction or redistribution of digital content, enabling the protection of copyrights. However, the incompatibility and lack of interoperability of the current DRM technologies heavily restricts the right of end-users, such as private reproduction. To address these problems, EXIM was developed by ETRI. In this paper, we studied the security specification for conversion technologies of heterogeneous DRM system, using the CC that meets specific consumer needs. The CTHDS_PP is a stated security environment that includes potential threats, and organizational security policies assumptions and security objectives, in order to uphold counter potential threats, and enforce organizational security policies, assumptions. The CTHDS_PP can be used to communicate these security requirements to potential developers, and provides a foundation from which a Security Target (ST) can be developed and formal evaluation can be conducted. Furthermore, a PP of similar security requirements can reuse all or part of existing PP or the CTHDS_PP.

References

1. Microsoft Windows Media Rights Manager, http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx
2. iTunes FairPlay, http://www.apple.com/lu/support/itunes/authetication.html
3. PachyDRM, http://pachydrm.com
4. Secure Digital Container, http://www.digicont.com
5. International Standard ISO/IEC 15408, Common Criteria for Information Technology Security Evaluation, Part 1 (2005)
6. International Standard ISO/IEC 15408, Common Criteria for Information Technology Security Evaluation, Part 2 (2005)

7. International Standard ISO/IEC 15408, Common Criteria for Information Technology Security Evaluation, Part 3 (2005)
8. International Standard ISO/IEC 15408, Common Methodology for Information Technology Security Evaluation, Evaluation methodology (2005)
9. http://digital-lifestyles.info/drm-interchange-alive-and-living-in-korea/
10. http://technomart.etri.re.kr
11. Herrmann, D.S.: Using the Common Criteria for IT Security Evaluation. Auerbach publications (2003)
12. Open Mobile Alliance, DRM Architecture Approved Version 2.0 (2006)
13. Apple Inc. Common Criteria Certification: Apple's Ongoing Commitment to Security, Whitepaper
14. Jaafari, A.B.: Common Criteria for Information Technology Security Evaluation Mobile Phone Digital Rights Management Protection Profile, Polytechnic University (2004)
15. Jaafari, A.B.: Protection Profile Reuse: Case Study of the reusability of the Smart Card Protection Profile for producing the Mobile Phone Digital Rights Management Protction Profile, Polytechnic University (2004)
16. Naini, R.S., Sheppard, N.P., Uehara, T.: Import/Export in Digital Rights Management. In: ACM Workshop on Digital Rights Management (2004)
17. Koenen, R.H., Lacy, J., Mackay, M., Mitchell, S.: The Long March to Interoperable Digital Rights Management. Proceedings of the IEEE 92, 883–897 (2004)
18. Bradley, W., Maher, D.: The NEMO P2P service orchestration framework. In: Proc. 37th Annu. Hawaii Int. Conf. System Sciences (2004)
19. Torres, V., Serrao, C., Dias, M.S., Delgado, J.: Open DRM and the Future of Media. IEEE Computer Society, Los Alamitos (2008)
20. Rump, N.: Can digital rights management be standardized? IEEE Signal Processing Magazine (2004)

Analysing Protocol Implementations[⋆]

Anders Moen Hagalisletto, Lars Strand,
Wolfgang Leister, and Arne-Kristian Groven

Norwegian Computing Center
{Anders.Moen,Lars.Strand,Wolfgang.Leister,Arne-Kristian.Groven}@nr.no

Abstract. Many protocols running over the Internet are neither formalised, nor formally analysed. The amount of documentation for telecommunication protocols used in real-life applications is huge, while the available analysis methods and tools require precise and clear-cut protocol clauses. A manual formalisation of the Session Initiation Protocol (SIP) used in Voice over IP (VoIP) applications is not feasible. Therefore, by combining the information retrieved from the specification documents published by the IETF, and traces of real world SIP traffic we craft a formal specification of the protocol in addition to an implementation of the protocol. In the course of our work we detected several weaknesses, both of SIP call setup and in the Asterisk implementation of the protocol. These weaknesses could be exploited and pose as a threat for authentication and non-repudiation of VoIP calls.

1 Introduction

Voice over IP (VoIP) is widely used, and is about to replace the traditional, public switched telephone networks (PSTN) for two main reasons: (1) Providers and customers experience cost savings, especially for long distance calls. Since VoIP uses the Internet as carrier, the cost of setting up a phone call needs no more effort than sending an email. (2) Added functionality and flexibility. The VoIP protocols are capable of providing a number of additional services like instant messaging, presence, conferencing, events notification, video calls and other multimedia transmissions and location independence (mobility).

VoIP services cannot rely their security on the telecommunication infrastructure, dedicated lines, physically protected switches, and certified telephony equipment. VoIP services have to be secured by cryptographic techniques. Therefore, the employed protoscols and their implementations must undergo a thorough formal crypto-analysis.

A common combination in VoIP is to use the Session Initiation Protocol (SIP) [14] for signalling, e.g., setting up and tearing down calls, and specific protocols for the actual media transfer. The designers of SIP focused on functionality for

[⋆] This research is funded by the EUX2010SEC project in the VERDIKT framework of the Norwegian Research Council. The authors would like to thank Jørn Inge Vestgaarden, Lothar Fritsch, Einar W. Høst, Svetlana Boudko, and Truls Fretland for comments on earlier drafts of this paper.

providing specific services rather than security features [15]. However, security issues have been recognised to be an area of further investigation and improvement [1,6,4,10]. Discussions about potential weaknesses and attacks on SIP have, in most cases, been kept on an informal level [16,4,13,18,17,12].

Our goal has been to use formal modelling of SIP in order to (1) verify whether the Asterisk implementation of SIP follow the specifications; and (2) perform attacks exploiting weaknesses in the protocol definition, and its implementations. This work is based on previous work [9] where we analysed digest access authentication in the SIP registration process. The same authentication mechanism is used in the call-setup explained in this paper.

There have been other works that analyse SIP and its security configurations formally [7,2], and the work of the AVISPA project[1]. However, they consider the authentication and registration sub-protocol in combination with the Diameter protocol, rather than the call-setup protocol as presented here.

The rest of the paper is organised as follows: In Section 2, we give a high level overview of SIP, and the tool PROSA that we have used to analyse the call setup. In Section 3, the formal specification of digest access authentication and call setup is described. After discussing whether Asterisk implements the SIP protocol correctly (Section 3.4), we show that a vicious attack on the call setup specification can be performed using the specification obtained previously is presented (Section 4). Finally, in Section 5, we evaluate the approach and point out future extensions of the PROSA tool.

2 Background

SIP is an application layer signalling protocol developed by the IETF. The core functionality of SIP is specified in RFC3261 [14], additional functionality is specified in over 40 RFCs, and nearly 30 pending SIP-related drafts[2]. SIP is used to establish, maintain and tear down multimedia sessions between two or more participants. A session can be an ordinary call between two participants, or an advanced multimedia conference session with several participants. More specific, SIP set up the session context, but does not carry multimedia content.

We illustrate the operation of SIP with a scenario where Alice calls Bob. A session including call setup and tear down is shown in Fig. 1. While a session generally may traverse several SIP proxies, we restrict our scenario to only one SIP proxy. The SIP proxy has three roles: (*i*) it acts as registration server, (*ii*) handles call setup, and (*iii*) routes the SIP messages and in some cases the media stream. In the call setup Alice calls Bob sending an 'INVITE' message (1, 2). A receipt for each such hop is returned by the 'Trying' message (3, 4). If Bob's phone is connected then it starts to ring and propagates the 'Ringing' message back to Alice (5, 6). When Bob answers the call, he sends an 'OK' message

[1] Automated Validation of Internet Security Protocols and Applications (AVISPA) project: `http://avispa-project.org/library/sip.html`

[2] IETF Session Initiation Protocol Charter: `http://www.ietf.org/html.charters/sip-charter.html`

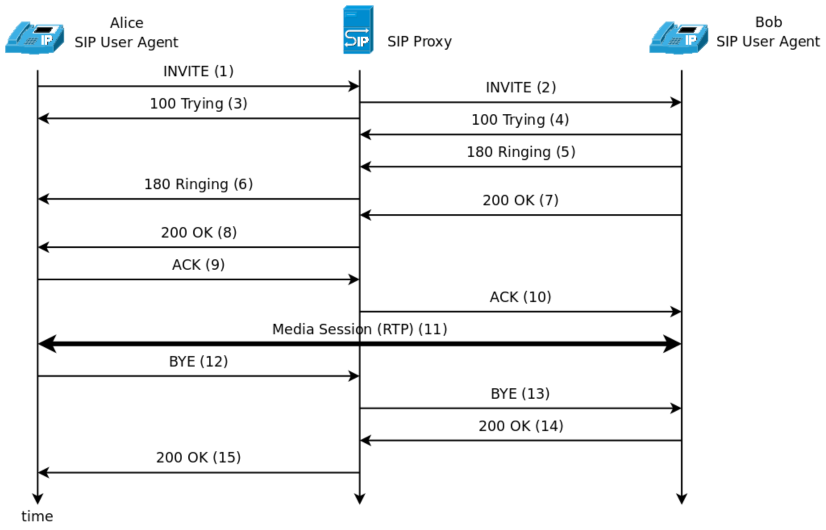


Fig. 1. Flow graph showing successful session establishment and termination using SIP

routed back to Alice through proxy S. Bob answers the phone in message (7), and the call content (voice) is transferred using the Real-time Transport Protocol (11). Alice terminates the call (12) and a 'BYE' message is sent. Before the 'INVITE' message (1) is sent, a SIP proxy may challenge Alice to authenticate, as formalised in Section 3.1.

For the purpose of this paper we restrict our work to how the widely used open source telephony platform Asterisk[3] [11] implements SIP. Asterisk is a private branch exchanges (PBX), whose functionality is to connect phone calls. Asterisk also supports a range of other common telephony services like voice mail, conference calls and telephone menus.

2.1 The Method

In order to gain initial knowledge of the behaviour of the SIP implementation in Asterisk, we record traces from real phone traffic going through the Asterisk server on a real-world Asterisk configuration. This is done by using the network monitoring tool Wireshark[4]. Traces retrieved from Wireshark can be presented both textually and as interaction diagrams. The process of obtaining a formal specification of SIP was roughly as follows: From the core IETF standards we derived an as accurate descriptions of SIP as possible. These resulting specifications were typically incomplete, interaction diagrams showing the transmissions and message content were lacking. Traces of the call setup with real softphones were then used to supplement the incomplete specifications, with details of message

[3] Asterisk homepage: `http://www.asterisk.org/`

[4] Wireshark homepage: `http://www.wireshark.org`

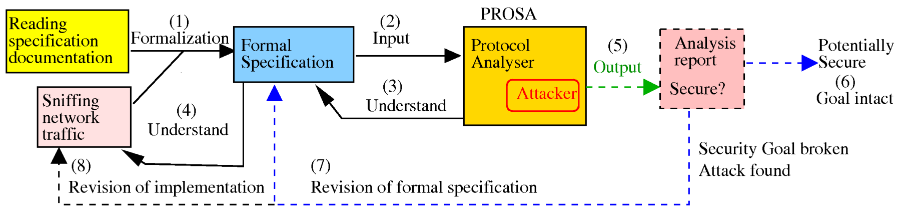


Fig. 2. Workflow for analysis of implementations

credentials. Hence, based on the Asterisk traces and SIP standard we constructed the formal models manually, and analysed the specifications in the the protocol analyser PROSA [8].

The analysis process is depicted in Fig. 2. In this process both IETF documentation and the traces are used to obtain a formal description (1) which typically enhances the understanding of the implementation (4). The formal specification is then validated by PROSA (2). If there are any errors or unreasonable elements found at this stage, the formal specification is revised (3). A correct protocol specification will then be subject to hand-crafted or automatically generated attacks. The correctness of manually constructed attacks will then be checked by validation in a similar way as for non-compromised protocol specifications, thereafter simulated in PROSA. Finally, the output of the analysis is a report that either confirms the initial security requirements or points at weaknesses that break some security goals (5). If no attacks are found then the protocol is considered preliminary secure (6). If an attack is found then the protocol is not secure and a revision is made (7). Since the formal specifications are derived from a concrete implementation, the report gives feedback onto the implementation (8).

2.2 The Protocol Analyser PROSA

PROSA [8] is a tool developed for the specification, static analysis and simulation of security protocols. PROSA consists of three main modules: (a) a specification language based on temporal epistemic logic; (b) a static analysis module; (c) a simulator for executing intended protocols and attacks on protocols.

The static analysis module consists of algorithms for *automated refinement* of both protocol specifications and attack descriptions. The automated refinement results in an explicit specification that contains local assumptions, i.e. pre- and postconditions, for each transmission clause. Refined specifications can then be *validated*. The validation process of a trace specification is performed in two steps in PROSA: First, a tool-supported refinement of the specification is generated. This will always give a longer specification that contains information about the agents beliefs and construction of credentials, like the generation of nonces, timestamps, assumptions about keys, and cryptographic operation like encryption, decryption and hashing. Secondly, the refined specification is validated to check whether a participant in the protocol setting possesses any beliefs that have not been legally obtained through communication or cryptography.

3 Formal Specification of SIP Call-Setup

SIP defines distinct functionality for registration, call setup and modification, call control and mid-call signalling. We refer to these parts as 'sub-protocols' since each of these parts requires its own sequence of message exchanges. We take a closer look at the sub-protocols *digest access authentication*, *call setup* and *call teardown*. *Call teardown* denotes the explicit event of terminating a call, specified by message 12-15 in Fig. 1. These are specified in a form commonly used in the literature. A protocol clause where "agent A sends a message M to the agent B" is of the form:

$$(P) \quad A \longrightarrow B \quad : \quad M$$

Messages in the protocols consist of basic entities as follows:

A, B, C, D, T, S, I, $I(A)$	agent terms
K_{AB}	symmetric key shared by A and B
N_A	nonce generated by A
W_A^Y	string containing the text Y related to A
X_A	miscellaneous entities related to A

We use three composition operators in the notation: concatenation of message content denoted by "," (comma), hashing $\mathsf{H}[M]$, and encryption denoted by $E(K : M)$, where K denotes a key and M a message content. The particular agent term $I(A)$ reads that "the intruder I impersonates as A".

3.1 Authentication

According to RFC3261 [14], there are three ways to configure SIP authentication: plain-text authentication, weak authentication, and strong authentication. Plain-text authentication sends the authentication credentials unprotected. Weak authentication is an adaptation of the HTTP digest access authentication [5] that requires a shared secret between the two participants. Strong authentication uses certificates in the same way as web browsers and servers use them. Weak authentication using a digest is by far the most common method. A typical application of digest access authentication is given by a challenger S requesting a client A to authenticate as described in the following protocol skeleton:

$$\begin{array}{ll}
(\mathrm{D}_1) & S \longrightarrow A : N_S \\
(\mathrm{D}_2) & A \longrightarrow S : W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N_A, W^{\mathrm{qop}}, \\
& \qquad \mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N_S, X_{\mathrm{nc}}, N_A, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{meth}}, W_A^{\mathrm{URI}}]]
\end{array}$$

Agents A and S share the symmetric key K_{AS}^{pwd}. Initially, the challenger S sends a nonce N_S to the client A. The client responds by sending the basic entities in plain text, except the password, and then the response itself. The entities involved in digest access authentication are as follows:

W_A^{uname}	username of A
W^{realm}	a protective domain
K_{AS}^{pwd}	shared password between A and S
W^{meth}	main method of message (like HTTP)
W_A^{URI}	Digest URI for client A
N_S	nonce of the challenger S
X_{nc}	nonce counter
N_A	A's nonce
W^{qop}	quality of protection

The authentication is one-way: A is authenticated to S, guaranteed by the secrecy of the shared key K_{AS}^{pwd}. Agent S can be certain that the message comes from A, since A is the only agent except S who possesses the key. Integrity of the message entities involved is provided by the fact that the hash could only be generated by A and freshness of the message is provided by the challenger nonce N_S.

3.2 The Teardown Sub-protocol

The trace described in Fig. 1 is one out many possible traces. *Teardown* of sessions can be performed at any stage in the session. Therefore the final four messages 12-15 in Fig. 1 and ($\mathrm{T}_{14} - \mathrm{T}_{17}$) in Fig. 4, can be considered as a teardown sub-protocol run by three agents. Instances of the teardown protocol might be running in parallel with the call setup protocol, which implies that a ‘BYE’ message received to any participant causes the host session of the call setup to be terminated. A SIP compliant specification where SIP methods are propagated correctly results in the following specification of teardown, extracted from the Wireshark protocol dump:

$$\begin{array}{llcl}
(\mathrm{TD}_1) & C \longrightarrow T & : & W^{\mathrm{BYE}}, D, W_D^{\mathrm{URI}}, W_C^{\mathrm{Contact}}, W_C^{\mathrm{URI}}, N_C^{\mathrm{callid}} \\
(\mathrm{TD}_2) & T \longrightarrow D & : & W^{\mathrm{BYE}}, C, W_C^{\mathrm{URI}}, N_D^{\mathrm{callid}} \\
(\mathrm{TD}_3) & D \longrightarrow T & : & W^{\mathrm{OK}}, W_D^{\mathrm{Contact}}, W_D^{\mathrm{URI}}, N_D^{\mathrm{callid}} \\
(\mathrm{TD}_4) & T \longrightarrow C & : & W^{\mathrm{OK}}, W_D^{\mathrm{Contact}}, W_D^{\mathrm{URI}}, N_C^{\mathrm{callid}}
\end{array}$$

where C denotes the role of the agent initiating the teardown, D denotes the responder agent, while T denotes the proxy server. Instances of the teardown protocol are started by the call setup protocol, while the call-setup is terminated by the teardown protocol.

3.3 Formalising Call Setup Permitting Arbitrary Teardown

Since we do not know which of the agents actively closes a session, we model that each agent starts a passive session of the teardown sub-protocol. In the example shown in Fig. 1 Alice actively closes the session, why she plays the role of B of

$$\begin{array}{lll}
(\mathrm{Q}_0) & A & : \ \mathrm{start}(\mathsf{teardown}, \mathsf{Role}(B, C), \mathsf{Role}(A, D), \mathsf{Role}(S, T)) \\
(\mathrm{P}_1) & A \longrightarrow S & : \ W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{Q}_1) & S & : \ \mathrm{start}(\mathsf{teardown}, \mathsf{Role}(A, C), \mathsf{Role}(B, D), \mathsf{Role}(S, T)) \\
(\mathrm{Q}_2) & S & : \ \mathrm{start}(\mathsf{teardown}, \mathsf{Role}(B, C), \mathsf{Role}(A, D), \mathsf{Role}(S, T)) \\
(\mathrm{P}_2) & S \longrightarrow A & : \ W^{\mathrm{PAR}}, W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, A, B, N_A^{\mathrm{callid}} \\
(\mathrm{P}_3) & A \longrightarrow S & : \ W^{\mathrm{ACK}}, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_4) & A \longrightarrow S & : \ W^{\mathrm{INVITE}}, A, B, N_S, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N'_A, W^{\mathrm{qop}}, N_A^{\mathrm{callid}} \\
 & \multicolumn{2}{l}{\quad \mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N'_A, X_{\mathrm{nc}}, N_S, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{INVITE}}, W_B^{\mathrm{URI}}]]} \\
(\mathrm{P}_5) & S \longrightarrow A & : \ W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_6) & S \longrightarrow B & : \ W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{Q}_3) & B & : \ \mathrm{start}(\mathsf{teardown}, \mathsf{Role}(B, D), \mathsf{Role}(A, C), \mathsf{Role}(S, T)) \\
(\mathrm{P}_7) & B \longrightarrow S & : \ W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{P}_8) & B \longrightarrow S & : \ W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{P}_9) & S \longrightarrow A & : \ W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_{10}) & B \longrightarrow S & : \ W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{P}_{11}) & S \longrightarrow A & : \ W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_{12}) & A \longrightarrow S & : \ W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{P}_{13}) & S \longrightarrow B & : \ W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
 & A \longleftrightarrow B & : \ \text{Media session}
\end{array}$$

Fig. 3. Call setup with flexible teardown based on RFC 3261

the teardown sub-protocol. Consequently, the Asterisk server acts in the role of S. Generally, both Alice and Bob might actively tear down a session by starting an instance of teardown in the role of A.

Combining the Wireshark protocol dump with the SIP specification we get Fig. 3. In order to model this we go beyond standard protocol notation and introduce the clauses $(\mathrm{Q}_0 - \mathrm{Q}_3)$, as seen in Fig. 3. The call setup involves the additional SIP methods and primitives:

W^{INVITE}	The INVITE method that indicates a request for phone call
W^{ACK}	An acknowledgement method
W^{PAR}	Proxy Authentication Required
W^{Trying}	100 Trying, receipt to a previous SIP message
W^{OK}	200 OK method notifies successful registration
W^{Ringing}	The responder's phone is ringing
W^{BYE}	Tearing down session (hang up phone)
$N_A^{\mathrm{callid}}, N_B^{\mathrm{callid}}$	Call identifiers for A (and S) and B (and S) respectively

Clause (Q_0) reads "agent A locally starts a session of the teardown protocol, such that agent A plays the responder role D, agent B plays the initiator role C, and the server S plays the server role". The notation $\mathsf{Role}(A, C)$ means that the agent A plays the C role in the given protocol, similar to procedure calls with parameter substitution in ordinary programming languages. The first local clause (Q_0) then says that A initially starts listening for a possible 'BYE' message from B. The server propagates the 'BYE' and 'OK' methods involved in the teardown sub-protocol. There are two cases: Clause (Q_1) starts a session where the server S is waiting for a 'BYE' from A, while in (Q_2), the server starts a similar session waiting for B sending a 'BYE'-message.

3.4 Deviations from the SIP Specification

The trace in Fig. 4 shows that the Asterisk implementation of SIP diverges from the specification described in Fig. 3 in three ways:

1. Alice's phone starts to ring (message T_7) *before* Bob is authenticated to the server. The meaning of an incoming 'Ringing' message received by Alice is that Bob has received an 'INVITE' message, and she is ready to start a call if Bob answers the call. Hence, in order to follow the SIP RFCs message (T_9) should come *before* message (T_7). Hence Alice is fooled to believe that Bob's phone is ringing, which is not the case. Therefore we simulated scenarios where the responder Bob was disconnected from the network just after receiving the 'INVITE' message. Alice still received a Ringing message. This behaviour is also confirmed in experiments using soft phones.
2. The acknowledgement received by Bob in message (T_{11}) arrives *before* Alice sends the message in clause (T_{13}). However, at this point Bob is mislead to believe that Alice has acknowledged the Ringing request from Bob.
3. After teardown initiated by Alice (T_{14}), the 'OK' message from S to Alice (T_{15}) is sent *before* the related 'OK' message is sent from Bob. This breaks the specification, since (T_{15}) would implicate that Bob has received the 'BYE' message, which is not the case in the implementation.

$$
\begin{array}{llll}
(\mathrm{T}_1) & A \longrightarrow S & : & W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_2) & S \longrightarrow A & : & W^{\mathrm{PAR}}, W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, A, B, N_A^{\mathrm{callid}} \\
(\mathrm{T}_3) & A \longrightarrow S & : & W^{\mathrm{ACK}}, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_4) & A \longrightarrow S & : & W^{\mathrm{INVITE}}, A, B, N_S, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N_A', W^{\mathrm{qop}}, N_A^{\mathrm{callid}} \\
 & \multicolumn{3}{l}{\quad \mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N_A', X_{\mathrm{nc}}, N_S, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{INVITE}}, W_B^{\mathrm{URI}}]]} \\
(\mathrm{T}_5) & S \longrightarrow A & : & W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_6) & S \longrightarrow B & : & W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_7) & S \longrightarrow A & : & W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_8) & B \longrightarrow S & : & W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_9) & B \longrightarrow S & : & W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{10}) & B \longrightarrow S & : & W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{11}) & S \longrightarrow B & : & W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{12}) & S \longrightarrow A & : & W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_{13}) & A \longrightarrow S & : & W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
 & A \longleftrightarrow B & & \text{Media session (RTP or SRTP)} \\
(\mathrm{T}_{14}) & A \longrightarrow S & : & W^{\mathrm{BYE}}, B, W_B^{\mathrm{URI}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_{15}) & S \longrightarrow A & : & W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(\mathrm{T}_{16}) & S \longrightarrow B & : & W^{\mathrm{BYE}}, A, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(\mathrm{T}_{17}) & B \longrightarrow S & : & W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}}
\end{array}
$$

Fig. 4. A trace of call setup and teardown using Asterisk as server S

This implementation can lead to unexpected results: An obvious attack targets the Ringing method: An intruder I could act as an eavesdropper until clause (T_7), then take over Bob's session entirely, kick Bob out of the call, and for the rest of the trace masquerade as Bob. Persons that are used to the particularly quick response (immediate ringing) from Asterisk based VoIP would not be alerted when the intruder I impersonates as Bob later in the session.

4 Attack on the Call Setup

An attack on the registration protocol of SIP has been found recently [9], yet in the following, we consider attacks that do not rely on successful registration

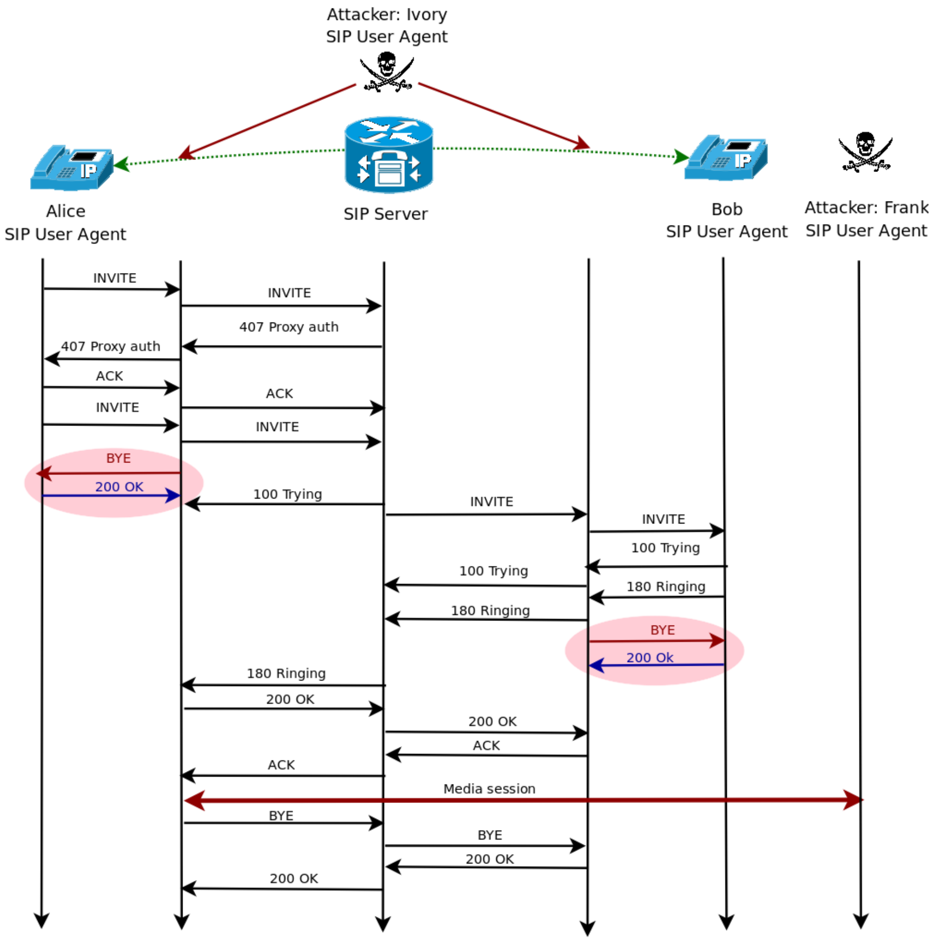


Fig. 5. Hijacking the initiator and the responder

attacks. We assume that the attacker I is as powerful as the Dolev Yao attacker [3] who controls the entire network, can intercept any message, impersonate as any other agent, and inject whatever entity it knows into SIP messages. Cryptography is assumed to be perfect, no brute force attacks on the underlying cryptographic algorithms are considered in this paper.

In the following we describe how it is possible for an attacker to hijack both the initiator and responder roles. In the initial part of the attack, described in Fig. 6, the intruder only passively listens in the authentication sub-protocol. In the protocol clauses $(P_{1.1.a})$ through $(P_{1.4.b})$ the intruder acts as a passive man-in-the-middle, obtaining information from plain text entities. From the knowledge gained during the initial eavesdropping, an attacker can perform a combined attack on both the caller and the callee.

In the attack, as shown in Fig. 5, the attacker Ivory (denoted I in Fig. 6) begins by eavesdropping the initial four messages concerned with establishing

$$
\begin{array}{lll}
(P_{1.1.a}) & A \longrightarrow I(S) & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{1.1.b}) & I(A) \longrightarrow S & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{1.2.a}) & S \longrightarrow I(A) & : W^{\mathrm{PAR}}, W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, A, B, N_A^{\mathrm{callid}} \\
(P_{1.2.b}) & I(S) \longrightarrow A & : W^{\mathrm{PAR}}, W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, N_S, A, B, N_A^{\mathrm{callid}} \\
(P_{1.3.a}) & A \longrightarrow I(S) & : W^{\mathrm{ACK}}, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{1.3.b}) & I(A) \longrightarrow S & : W^{\mathrm{ACK}}, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{1.4.a}) & A \longrightarrow I(S) & : W^{\mathrm{INVITE}}, A, B, N_S, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N'_A, W^{\mathrm{qop}}, N_A^{\mathrm{callid}} \\
 & \multicolumn{2}{l}{\mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N'_A, X_{\mathrm{nc}}, N_S, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{INVITE}}, W_B^{\mathrm{URI}}]]} \\
(P_{1.4.b}) & I(A) \longrightarrow S & : W^{\mathrm{INVITE}}, A, B, N_S, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, X_{\mathrm{nc}}, N'_A, W^{\mathrm{qop}}, N_A^{\mathrm{callid}} \\
 & \multicolumn{2}{l}{\mathsf{H}[\mathsf{H}[W_A^{\mathrm{uname}}, W^{\mathrm{realm}}, K_{AS}^{\mathrm{pwd}}], N'_A, X_{\mathrm{nc}}, N_S, W^{\mathrm{qop}}, \mathsf{H}[W^{\mathrm{INVITE}}, W_B^{\mathrm{URI}}]]} \\
(T_{2.3.2}) & I(S) \longrightarrow A & : W^{\mathrm{BYE}}, B, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(T_{2.3.3}) & A \longrightarrow I(S) & : W^{\mathrm{OK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{2.5}) & S \longrightarrow I(A) & : W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{2.6.a}) & S \longrightarrow I(B) & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(P_{2.6.b}) & I(S) \longrightarrow B & : W^{\mathrm{INVITE}}, A, B, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(P_{2.7.a}) & B \longrightarrow I(S) & : W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(P_{2.7.b}) & I(B) \longrightarrow S & : W^{\mathrm{Trying}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(P_{2.8.a}) & B \longrightarrow I(S) & : W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(P_{2.8.b}) & I(B) \longrightarrow S & : W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(T_{2.4.2}) & I(S) \longrightarrow B & : W^{\mathrm{BYE}}, A, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(T_{2.4.3}) & B \longrightarrow I(S) & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(P_{2.9}) & S \longrightarrow I(A) & : W^{\mathrm{Ringing}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{2.10}) & I(B) \longrightarrow S & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(P_{2.11}) & S \longrightarrow I(A) & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{2.12}) & I(A) \longrightarrow S & : W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(P_{2.13}) & S \longrightarrow I(B) & : W^{\mathrm{ACK}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
 & \mathbf{I(A) \longleftrightarrow F(B)} & : \textbf{Media session} \\
(T_{2.2.1}) & I(A) \longrightarrow S & : W^{\mathrm{BYE}}, B, W_B^{\mathrm{URI}}, W_A^{\mathrm{Contact}}, W_A^{\mathrm{URI}}, N_A^{\mathrm{callid}} \\
(T_{2.5.2}) & S \longrightarrow I(B) & : W^{\mathrm{BYE}}, A, W_A^{\mathrm{URI}}, N_B^{\mathrm{callid}} \\
(T_{2.2.3}) & I(B) \longrightarrow S & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_D^{\mathrm{callid}} \\
(T_{2.2.4}) & S \longrightarrow I(A) & : W^{\mathrm{OK}}, W_B^{\mathrm{Contact}}, W_B^{\mathrm{URI}}, N_A^{\mathrm{callid}}
\end{array}
$$

Fig. 6. Formal attack when hijacking the initiator and the responder

the call and authenticating the caller Alice to the server. Then Ivory tears down Alice's session prematurely by using a 'BYE' message, and thereafter terminates Bob's session before he has entered the media session. Before the media session, the attacker has taken over the call, and can start a conversation with agent Frank (denoted F in Fig. 6).

The attacker tears down the session after pretending to be Alice. The server S cannot discover that the two local sessions at each calling party are teared down. The attack effectively breaks the authenticity of the participants, and therefore results in an identity management problem. The consequence could be that the intruder I could set up an arbitrary call, that Alice is billed for. Another consequence of this attack could be that the logs, that telephony providers are obliged to carry out by legislation, are incorrect. Hence, the attack shows that non-repudiation is broken as well.

5 Discussion and Conclusions

Our method reveals concrete attacks that indicate where improvements in the protocol are necessary. We discovered that the well-known SIP implementation Asterisk deviates from the SIP specification, and found a severe call hijack attack, where intruders can completely take over a phone call.

Our work provides the designers of VoIP protocols with a set of tools for protocol analysis. The PROSA language and framework can be used to formally specify and analyse protocols and their specific implementations in a rigorous way. The use of traces, and the analysis with PROSA, can help to detect differences between protocol specification and implementation. The behaviour of implementations is therefore analysed, and treated as specification for a variant of the employed protocols. Attacks could be designed and tested rapidly compared to the traditional approaches as described in [13] and [4].

References

1. Arkko, J., Torvinen, V., Camarillo, G., Niemi, A., Haukka, T.: Security Mechanism Agreement for the Session Initiation Protocol (SIP). RFC 3329 (Proposed Standard) (January 2003)
2. Diab, W.B., Tohme, S., Bassil, C.: VPN analysis and new perspective for securing voice over VPN networks. ICNS 0, 73–78 (2008)
3. Dolev, D., Yao, A.C.-C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–207 (1983)
4. Endler, D., Collier, M.: Hacking Exposed VoIP: Voice over IP Security Secrets and Solutions. McGraw-Hill Osborne Media, New York (2006)
5. Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Stewart, L.: HTTP Authentication: Basic and Digest Access Authentication. RFC 2617 (Draft Standard) (June 1999)
6. Geneiatakis, D., Kambourakis, G., Dagiuklas, T., Lambrinoudakis, C., Gritzalis, S.: SIP Security Mechanisms: A state-of-the-art review. In: Proceedings of the Fifth International Network Conference (INC 2005), pp. 147–155 (July 2005)

7. Gupta, P., Shmatikov, V.: Security Analysis of Voice-over-IP Protocols. In: 20th IEEE Computer Security Foundations Symposium, 2007. CSF 2007, pp. 49–63 (2007)
8. Hagalisletto, A.M.: Automated Support for the Design and Analysis of Security Protocols. PhD thesis, University of Oslo (December 2007)
9. Hagalisletto, A.M., Strand, L.: Formal modeling of authentication in SIP registration. In: Second International Conference on Emerging Security Information, Systems and Technologies SECURWARE 2008, pp. 16–21 (August 2008)
10. Kuhn, D.R., Walsh, T.J., Fries, S.: Security Consideration for Voice over IP Systems. Sp 800-58, National Institute of Standards and Technology (NIST) (January 2005)
11. Meggelen, J., Smith, J., Madsen, L.: Asterisk: The Future of Telephony. O'Reilly Media, Sebastopol (2005)
12. Persky, D.: VoIP Security Vulnerabilities. Technical report, SANS Institute (2007)
13. Porter, T.: Practical VoIP Security. Syngress (March 2006)
14. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), Updated by RFCs 3265, 3853, 4320, 4916 (June 2002)
15. Salsano, S., Veltri, L., Papalilo, D.: SIP security issues: The SIP authentication procedure and its processing load. IEEE Network 16, 38–44 (2002)
16. Sinnreich, H., Johnston, A.B.: Internet communications using SIP: Delivering VoIP and multimedia services with Session Initiation Protocol, 2nd edn. John Wiley & Sons, Inc., New York (2006)
17. Xin, J.: Security issues and countermeasure for VoIP. Technical report, SANS Institute (2007)
18. Zhang, R., Wang, X., Yang, X., Jiang, X.: Billing Attacks on SIP-Based VoIP Systems. In: USENIX, First USENIX Workshop on Offensive Technologies (WOOT 2007) (August 2007)

Measuring Anonymity[⋆]

Xiaojuan Cai[1] and Yonggen Gu[2]

[1] BASICS Lab, Department of Computer Science,
Shanghai Jiao Tong University, Shanghai, China, 200240
cxj@sjtu.edu.cn
[2] Department of Computer Science,
Huzhou Teacher College, Zhejiang, China, 313000
gyg68@hutc.zj.cn

Abstract. Some systems offer probabilistic anonymity. The degree of anonymity is considered and defined by Reiter and Rubin [1]. In this paper metrics are proposed to measure anonymity of probabilistic systems. The metric induces a topology on probabilistic applied π processes, which are used to model anonymous systems. The degree of anonymity is formally defined, and as an illustrating example, Crowds – an anonymous system for web transaction – is analyzed.

Keywords: Probabilistic applied π calculus, anonymity, metric, Crowds.

1 Introduction

The importance of anonymity has increased over the past few years. Various systems have been proposed to implement anonymity for various kinds of network communication, such as FOO electronic voting protocol [2], untraceable electronic cash protocol [3] and so on. Some formal methods are applied to analyze their anonymity property, for example, Chothia analyzed the MUTE anonymous file-sharing system using the π calculus in [4].

At the same time, some probabilistic anonymous systems, which extend non-probabilistic anonymous systems with probability, have been proposed, such as Crowds [1] and some variants of DCP [5]. Crowds is a system developed by Reiter and Rubin [1] for protecting users' anonymity on the world-wide-web. It operates by grouping users into a large group called crowd. When the web server receives a request, neither web server nor collaborating crowd members can learn the true source of this request.

Many formal methods are built to analyze the anonymity of probabilistic protocols or systems, there are two groups of frameworks for analyzing anonymity:

– *Logic + Model checking.* Shmatikov uses PCTL to model Crowds system and applies PRISM to automatically check the anonymity in [6]; Halpern [7]

⋆ This work is supported by the National 973 Project (2003CB317005), the National Nature Science Foundation of China (60573002, 60703033) , the Nature Science Foundation of Huzhou City Zhejiang Province (2007YZ10) and the Sci-Tech Brainstorm Stress Project of Huzhou City Zhejiang Province (2007G22).

also gives the definition of anonymity in the probabilistic way, and some are corresponding to the definitions by Reiter in [1].
– *Shannon's entropy.* Diaz *et al* [8] give a measure system for the anonymity with Shannon's definition of entropy. However, the analysis cannot be automated and the capability of the attacker is limited.

The framework in this paper is not included in above two groups. We model anonymous systems in process calculi and use metrics to define anonymity. Process calculi provide a tool for the high-level description of interactions and communications between agents or processes. Using process calculi to analyze security protocols was first studied by Lowe in [9] with CSP. Later on the spi calculus [10] was proposed by extending the π calculus with cryptographic primitives, such as encryption and decryption. In [11] Abadi and Fournet introduced the applied π calculus, which is a simple extension of the π calculus with value passing, primitive functions and equations among terms. It has been used to model security protocols like Just Fast Keying [12]. Verification of authentication using the applied π calculus has been studied in [13,14]. It is natural and convenient to model protocols in process calculi.

In this paper, we use the probabilistic applied π calculus (PAπ for short) proposed by Goubault-Larrecq *et al* [15] as the framework for analyzing probabilistic anonymous systems. PAπ extends the applied π calculus with probabilistic choice($\oplus$). Instead of bisimulation, we use *pseudo-metric* to measure the difference between two processes. A pseudo-metric is a function which defines a distance between elements of a set, and the calculation can be automatically carried out. Some pioneering work about metrics on process calculi include [16,17] and [18]. The main contributions of this paper can be summarized as follows:

– We define pseudo-metric on pairs of processes. It equals to 0 if and only if two processes are strongly bisimilar. We first replace weak bisimilarity with strong one in analyzing security protocols because we found that the protocol and its specification behave very similarly to each other. We need not consider weak bisimilarity for its complicated definition in probabilistic process calculi. More discussions can be found in Section 3.
– According to the pseudo-metric, we give the formal definitions of *beyond suspicion, probable innocence* and *possible innocence* – three degrees of anonymity from strong to weak in [1].
– We model Crowds and analyze the anonymity degree. We study how the number of participants and the probability of forwarding influence the degree of anonymity.

The rest of the paper is organized as follows. Section 2 briefly introduces the probabilistic applied π calculus. Section 3 defines pseudo-metric between two probabilistic applied π processes, and demonstrates the correctness of the metric by showing when two processes are strongly bisimilar the metric is 0. Section 4 defines the degree of anonymity based on pseudo-metric. Section 5 analyzes Crowds system. Finally, we conclude in Section 6.

2 Probabilistic Applied π Calculus

This section covers the necessary background materials on PAπ. More details are referred to [15].

Given a signature $\Sigma = \{(f_1, a_1), \cdots, (f_n, a_n)\}$ consisting of a finite set of function symbols f_i each with an arity a_i, *terms, plain processes* and *extended processes* are defined by the following grammar:

(Terms)	$M, N ::= a, b, c, \cdots \mid x, y, z, \cdots \mid \mathsf{f}(M_1, \cdots, M_l)$
(Plain Process)	$P, Q ::= \mathbf{0} \mid \bar{u}\langle M\rangle.P \mid u(x).P \mid P + Q \mid P \oplus_p Q$
	$\mid P \mid Q \mid !P \mid \nu\, n.P \mid \mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$
(Extended Process)	$A, B ::= P \mid \{M/x\} \mid A \mid B \mid \nu\, n.A \mid \nu\, x.A$

We use $(p_1)P_1 + (1-p_1)P_2$ to denote $P_1 \oplus_{p_1} P_2$, so process $\Sigma_{i\in I}(p_i)P_i$ represents for $(p_1)P_1 + (p_2)P_2 + \cdots + (p_n)P_n$ with $\Sigma_{i\in I} p_i = 1$. $\Pi_{i\in\{1,2,\cdots,n\}} A_i$ is abbreviation for $A_1 \mid A_2 \mid \cdots \mid A_n$, and $\nu\,\widetilde{l}.A$ for $\nu\, l_1.\nu\, l_2.\cdots.\nu\, l_n.A$ where $\widetilde{l} = \{l_1, l_2, \cdots, l_n\}$. We use *Act* denotes the set of all possible actions in the labeled rules.

We write $fv(A)$, $bv(A)$, $fn(A)$, and $bn(A)$ for free and bound variables, free and bound names of A. The set of names and variables that occur in A is denoted as $n(A) \overset{\mathsf{def}}{=} bn(A) \cup fn(A)$ and $v(A) \overset{\mathsf{def}}{=} bv(A) \cup fv(A)$.

Equational theory which consists of a set of equations plays an important role in PAπ. For every signature Σ, we equip it with an equational theory E. We use $\Sigma \vdash M =_E N$ to denote terms M, N are equivalent in E. Sometimes we simply write $M = N$ if there is no ambiguity.

A *frame* is an extended process built up from $\mathbf{0}$ and active substitutions by parallel composition and restriction. The frame of an extended process A, denoted by ϕ_A, can be written in the form of $\phi_A \overset{\mathsf{def}}{=} \nu\,\widetilde{n}.\{\widetilde{M}/\widetilde{x}\}$ where $\widetilde{n} \subseteq n(\widetilde{M})$. If $\widetilde{M} = \langle M_1, M_2, \cdots, M_n\rangle$ and $\widetilde{x} = \langle x_1, x_2, \cdots, x_n\rangle$, then

$$\phi_A \equiv \nu\,\widetilde{n}.(\{M_1/x_1\} \mid \{M_2/x_2\} \mid \cdots \mid \{M_n/x_n\}).$$

Every extended process can be mapped to a frame. We write $dom(\phi)$ for the domain of ϕ, a set of variables which appear in active substitutions in ϕ but not under a variable restriction. The domain of an extended process is the domain of its frame.

Static equivalence is an equivalence relation built on closed extended processes, which depicts the indistinguishability of their frames. The following definitions were firstly introduced in [11].

Definition 1. *We say that two terms M and N are equal in the frame ϕ, and write $(M = N)\phi$, if and only if $\phi \equiv \nu\,\widetilde{n}.\sigma, M\sigma = N\sigma$, and $\{\widetilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$ for some names $\widetilde{n}$ and substitution σ.*

Definition 2. *We say that two closed frame ϕ and ψ are statically equivalent, and write $\phi \approx_s \psi$, when $dom(\phi) = dom(\psi)$ and when, for all terms M and N, we have $(M = N)\phi$ if and only if $(M = N)\psi$.*

Definition 3 (Static equivalence). *($\approx_s$) We say that two closed extended processes are statically equivalent, and write $A \approx_s B$, when their frames are statically equivalent.*

To denote a *discrete probability distribution* over a set X, we use a function $\mu : 2^X \longrightarrow [0,1]$ with the following properties: $\mu(X) = 1$ and $\mu(\cup_i X_i) = \Sigma_i \mu(X_i)$. We use δ_x to denote the dirac measure on x, and we define $\mu = \mu_1 +_p \mu_2$ as $\mu(Y) = p \cdot \mu_1(Y) + (1-p)\mu_2(Y)$. *Bisimulation* is a binary relation between state transition systems, associating systems which behave in the same way in the sense that one system simulates the other and vice-versa. It is very important in process calculi. Let A be a closed extended process, $\mathcal{C}$ be a set of closed extended processes, the probability of A performing action α and evolving into a process in set $\mathcal{C}$ is defined as follows:

$$Prob_A(\alpha, \mathcal{C}) = \sum_{A' \in \mathcal{C}} \mu(A'), \qquad \text{where } A \xrightarrow{\alpha} \mu$$

We now give the definition of *strong bisimulation.*

Definition 4 (Strong bisimulation). *Strong bisimulation ($\sim$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies:*

- *$A \approx_s B$;*
- *for any α, and any classes $\mathcal{C} \in \mathcal{A}/_{\mathcal{R}}$, $Prob_A(\alpha, \mathcal{C}) = Prob_B(\alpha, \mathcal{C})$.*

3 Metric

As bisimulation is not a robust concept in PAπ model, a pseudo-metric analogue of strong bisimulation is developed in this section.

3.1 Pseudo-metric

In mathematics, a metric or distance function is a function which defines a distance between elements of a set. A set with a metric is called a metric space. A metric induces a topology on a set but not all topologies can be generated by a metric.

Definition 5 (Metric). *A c-bound metric on a set X is a function (called the distance function or simply distance) $m : X \times X \to [0, c]$. For all x, y, z in X, this function is required to satisfy the following conditions:*

1. *$m(x, y) \geq 0$;*
2. *$m(x, y) = 0$ if and only if $x = y$;*
3. *$m(x, y) = m(y, x)$;*
4. *$m(x, z) \leq m(x, y) + m(y, z)$.*

In this paper, we only consider 1-bound metrics. If the second requirement is dropped, the function is called a *pseudo-metric*. Let the metric space $X = \mathcal{A}$, we define a pre-order on all 1-bound pseudo-metrics of the set of all closed extended processes.

Definition 6. *$\mathcal{M}$ is the class of 1-bounded pseudo-metrics on extended processes with the ordering:*
$m_1 \preceq m_2$ *if* $(\forall A, B \in \mathcal{A})[m_1(A,B) \geq m_2(A,B)]$.

We use $\top$ to denote the top element $\forall A, B.\top(A,B) = 0$; the bottom element is given by $\bot(A,B) = 1$ if $A \neq B$ otherwise 0. One can easily get that $(\mathcal{M}, \preceq)$ is a complete lattice.

Since the labeled transition system of PAπ is an LCMC (labeled concurrent Markov chain), every process evolves into a distribution on processes after a probabilistic transition. We lift the metric to the set of distributions.

Definition 7. *Suppose $m \in \mathcal{M}$, and μ, μ' be distributions on extended processes, then $m(\mu, \mu')$ is given by the solution of the following linear program:*

max: $\Sigma_i(\mu(A_i) - \mu'(A_i))(a_i - a_K)$

subject to: $\forall i. 0 \leq a_i, a_K \leq 1, \forall i, j. a_i - a_j \leq m(A_i, A_j)$

Definition 8. *For closed extended processes A and B. The action set of A corresponding to B, denoted $L(A,B)$, is the set:*

$$\{\alpha \mid \alpha \in Act \wedge fv(\alpha) \subseteq dom(A) \wedge bn(\alpha) \cap fn(B) = \emptyset\}$$

$L(A,B)$ is the set of actions A can perform and the bound names in these actions should be renamed in order not to clash with the free names of B. We now define a function F on $\mathcal{M}$ that closely resembles strong bisimulation.

Definition 9. *Given two closed extended processes A and B, a function F on $\mathcal{M}$ is defined as follows:*

- *if $A \not\approx_s B$, then $F(m)(A,B) = 1$;*
- *if $A \approx_s B$, $F(m)(A,B) =$*

$$\max\Big(\max_{\alpha \in L(A,B)} \sup_{A \xrightarrow{\alpha} \mu} \inf_{B \xrightarrow{\alpha} \mu'} m(\mu, \mu'), \max_{\alpha \in L(B,A)} \sup_{B \xrightarrow{\alpha} \mu'} \inf_{A \xrightarrow{\alpha} \mu} m(\mu, \mu')\Big)$$

where $\inf \emptyset = 1$, $\sup \emptyset = 0$.

We can easily get that F is monotone on $\mathcal{M}$, so F has a maximum fixed point which is given by $m_{max} = \sqcup_i m_i$ where $m_0 = \top$ and $m_{i+1} = F(m_i)$. In the rest paper, we mean $m_{max}(A,B)$ when we say "the metric between A, B".

We give some examples about how to calculate the metric between two processes. In each example, we use the following equational theory:

$$\mathrm{fst}((x,y)) = x$$
$$\mathrm{snd}((x,y)) = y$$
$$\mathrm{dec}(\mathrm{enc}(x,y),y) = x$$

Example 1. $m(P \mid \{m/x\}, Q \mid \{n/x\}) = 1$.

Since P, Q are plain processes, and we have $x\{m/x\} = m$ but $x\{n/x\} \neq m$, so the two processes are not statically equivalent.

Example 2.

$$A \overset{\mathsf{def}}{=} (\bar{c}\langle m\rangle + \bar{d}\langle n\rangle) \oplus_{0.5} \bar{d}\langle n\rangle$$
$$B \overset{\mathsf{def}}{=} \bar{c}\langle m\rangle \oplus_{0.25} \bar{d}\langle n\rangle$$

We have $A \approx_s B$, therefore $m(A, B)$ is the maximum fixed point of $F(m)$:

$$\begin{aligned} F(m) &= \max(\max\nolimits_{\alpha\in L(A,B)} \sup\nolimits_{A\xrightarrow{\alpha}\mu} \inf\nolimits_{B\xrightarrow{\alpha}\mu'} m(\mu,\mu'), \\ &\qquad \max\nolimits_{\alpha\in L(B,A)} \sup\nolimits_{B\xrightarrow{\alpha}\mu'} \inf\nolimits_{A\xrightarrow{\alpha}\mu} m(\mu,\mu')) \\ &= \max(m(\mu,\mu'), m(\mu',\mu)) \\ &= m(\mu,\mu') \end{aligned}$$

where $\alpha = \tau$, $\mu = (\bar{c}\langle m\rangle + \bar{d}\langle n\rangle) : 0.5, \bar{d}\langle n\rangle : 0.5$ and $\mu' = \bar{c}\langle m\rangle : 0.25, \bar{d}\langle n\rangle : 0.75$.
$m(\mu, \mu')$ is the solution to the following linear program:

$$\begin{array}{l} \max[(0.5-0)(a_1-a_K)+(0.5-0.75) \\ (a_2-a_K)+(0-0.25)(a_3-a_K)] \\ \textit{subject to}: 0 \le a_1, a_2, a_3, a_K \le 1 \\ \qquad\qquad |a_1-a_2| \le m(\bar{c}\langle m\rangle + \bar{d}\langle n\rangle, \bar{d}\langle n\rangle) = 1 \\ \qquad\qquad |a_1-a_3| \le m(\bar{c}\langle m\rangle + \bar{d}\langle n\rangle, \bar{c}\langle m\rangle) = 1 \\ \qquad\qquad |a_2-a_3| \le m(\bar{c}\langle m\rangle, \bar{d}\langle n\rangle) = 1 \end{array}$$

We can finally get that $m_{max}(A, B) = 0.5$.

This example gives us the institution that nondeterminism choice is different from the probabilistic one. If we apply the uniform distribution on the nondeterministic choice and get $A' \overset{\text{def}}{=} (\bar{c}\langle m\rangle \oplus_{0.5} \bar{d}\langle n\rangle) \oplus_{0.5} \bar{d}\langle n\rangle$, then $m(A', B) = 0$. Another important observation is that $A \not\sim B$ and $A' \sim B$ by the definition of strong bisimulation.

Example 3.

$$A \overset{\mathsf{def}}{=} \nu\, m, k.\bar{a}\langle \mathrm{enc}(m,k)\rangle.\bar{b}\langle m\rangle$$
$$B \overset{\mathsf{def}}{=} \nu\, m, n, k.(\bar{a}\langle \mathrm{enc}(m,k)\rangle.\bar{b}\langle n\rangle \oplus_{0.5} \bar{a}\langle \mathrm{enc}(n,k)\rangle.\bar{b}\langle n\rangle)$$

We can get $m_{max}(A, B) = 0$ though they look quite different.

The point of this example is that

$$\nu\, k, m.\{\mathrm{enc}(m,k)/x, m/y\} \approx_s \nu\, k, m, n.\{\mathrm{enc}(m,k)/x, n/y\},$$
$$\nu\, k, m.\{\mathrm{enc}(m,k)/x, m/y\} \approx_s \nu\, k, m, n.\{\mathrm{enc}(n,k)/x, n/y\}.$$

The following theorem demonstrates the correctness of the definition of metric.

Theorem 1. $A \sim B$ *iff* $m_{max}(A, B) = 0$.

3.2 Strong Bisimulation v.s. Weak Bisimulation

When analyzing security protocols with process calculi, we usually use observational equivalence to denote that the environment cannot distinguish two instances of a protocol (anonymity), or cannot tell the difference between the protocol and its specification (secrecy, authenticity). As observable equivalence coincides with weak bisimulation in process calculi such as π, Applied π and PAπ, weak bisimulation is used instead of observational equivalence for its convenience in proof. However, the proof of weak bisimulation is much more complicated than that of strong bisimulation. We know that strong bisimulation implies weak bisimulation, so can we replace weak bisimulation with strong one in analysis of security protocols?

To our best knowledge, the specification and the protocol itself behaves very similar in the definitions of most security properties. In [10] Abadi and Gordon define the authenticity and secrecy security protocol as follows:

Authenticity: $Inst(M) \simeq Inst_{spec}(M),\ for\ any M;$

Secrecy: $Inst(M) \simeq Inst(M')\ if\ F(M) \simeq F(M'),\ for\ any\ M, M'.$

where $\simeq$ is testing equivalence in Spi calculus and $Inst_{spec}$ is the specification of $Inst$. In all the given example of [10], process $Inst_{spec}$ only changes some messages of $Inst$. It is easy to get that $Inst(M) \approx Inst_{spec}(M)$ if and only if $Inst(M) \sim Inst_{speic}(M)$ in applied π calculus. The secrecy is similar. We find the same result in the definition of anonymity (non-probabilistic anonymity) [19,14]. So we can use strong bisimulation to eliminate the complexity caused by τ actions in weak bisimulation.

4 Anonymity Degree

Anonymity is a general concept, which can be refined into more precise definitions. In [20], three types of anonymous communication properties are proposed: sender anonymity, receiver anonymity and unlinkability of sender and receiver. Sender anonymity means that the identity of the party who sent a message is hidden, while the identity of the receiver might not be. Receiver anonymity similarly means that the identity of the receiver is hidden. Unlinkability of sender and receiver means that though the sender and receiver can each be identified, but the relation between them cannot be identified.

For each type of anonymity, Reiter and Rubin defined six degrees of anonymity in [1]. Three of them attracted more attention. Take sender anonymity as an example:

– *Beyond suspicion*: if though the attacker can see evidence of a sent message, the sender appears no more likely to be the originator of that message than any other potential sender in the system.
– *Probable innocence*: if from the attacker's point of view the sender appears no more likely to be the originator than to not be the originator.

– *Possible innocence*: if from the attacker's point of view, there is a nontrivial probability that the real sender is someone else.

Probable innocence is weaker than beyond suspicion because the attacker may have reason to expect that the sender is more likely to be responsible than any other potential sender, but it still appears at least as likely that the sender is not responsible. Possible innocence is even weaker because the sender is almost exposed. But it is still not exactly exposed, that is to say, the probability of someone to be the actual sender is strictly less than 1.

For anonymous systems, global adversary can make the so called "intersection attack". Because not all users are active all the time, the owner of the initial message can be detected by global adversary. As a result of it, recently people add dummy traffic [21] into anonymous systems to prevent such attacks. In this paper we will not consider global attackers, but think about the local corrupted parties each of which is a local active attacker.

Suppose A is a probabilistic anonymous system, then $A(s, r)$ denotes an instance of A that the sender is s and receiver is r. Let $\widetilde{k_i}$ denote the private knowledge of some participant i, such as the private keys, newly generated random numbers and etc. Chn_i denotes the set of all the channels related to i. For a set of corrupted participants T, we write $A^T(s, r)$ for the instance in which the protocol $A(s, r)$ runs with all the private knowledge of corrupted parties in T being revealed to the environment and all the private channels being under the control of the environment. In other words, these collaborated participants have become attackers. By applying the structural rules, we have $A(s, r) \equiv \nu\widetilde{n}.A'(s, r)$ where $\widetilde{n} = fn(A(s, r))$. Let $\widetilde{K} = \cup_{i \in T} k_i$,

$$A^T(s, r) \overset{\mathsf{def}}{=} \nu\,\widetilde{n'}.(A'(s, r) \mid \{\widetilde{k}/\widetilde{x_j}\})$$

where $\widetilde{n'} = \widetilde{n} \setminus (\cup_{i \in T} Chn_i)$. Now we are ready to give the formal definition of the degree of anonymity.

Let d_s^T denotes the maximum metric between two instances with different senders and same receivers with the set of corrupted parties T, and let d_r^T means the maximum metric between two instances with same senders and different receivers with the set of corrupted parties T. Formally,

$$d_s^T \overset{\mathsf{def}}{=} \max_{r \in R} \max_{s, s' \in S \setminus T} \{m_{max}(A^T(s, r), A^T(s', r))\}$$

$$d_r^T \overset{\mathsf{def}}{=} \max_{s \in S} \max_{r, r' \in R \setminus T} \{m_{max}(A^T(s, r), A^T(s, r'))\}$$

Definition 10. *For any anonymous system A, whose possible sender set is S, the possible receiver set is R, we say A satisfies,*

– beyond suspicion sender anonymity *under corrupted parties T, if $d_s^T = 0$,*
– probable innocence sender anonymity *under corrupted parties T, if $d_s^T < 0.5$,*
– possible innocence sender anonymity *under corrupted parties T, if $d_s^T < 1$;*
– *Similarly, we use d_r^T to replace d_s^T in above statements to define the degree of receiver anonymity.*

5 Analysis of Crowds

In this section, we first briefly introduce Crowds system; then we model crowds and analyze its anonymity.

5.1 Crowds

Crowd is a system developed by Reiter and Rubin [1] for protecting users' anonymity on the world-wide-web. It groups users into a large group called crowd. A user is represented in a crowd by a process on his/her computer called a *jondo*. All communication between any two jondos is encrypted using the symmetric key between them. A request will be sent to the destination with the following steps:

- When receiving the request from the browser, the proxy jondo, say I picks a jondo, say A_1 (possibly himself) at random, and forwards the request to it.
- When A_1 receives the request, it flips a biased coin to determine whether or not to forward the request to another jondo. The coin indicates to forward with probability p_f.
- If the result is to forward, then the jondo selects a random jondo and forward the request to it, otherwise, submits the request to the destined server.

5.2 Modeling

In this paper, we only consider the sender anonymity of Crowds. To simplify the analysis, we assume there is one server, denoted by r, who wants to find the identity of the real sender. Since we do not consider the global attacker, we may assume the channels are private and the messages may not be encrypted. The environment only can listen to the channels of corrupted parties.

Assume there are n jondos in this Crowd. We use $J = \{1, 2, \cdots, n\}$ to denote the set of jondos, and use c_{ij} with $i < j$ to express the channel between jondo i and j. We write $CJ = \bigcup_{i,j \in J \text{ and } i<j} \{c_{ij}\}$ for the set of channels between jondos, $CR = \bigcup_{i \in J} \{c_{ir}\}$ for the channels between a jondo and the server r. $CJ \cup CR$ is denoted by C. We use the process $Crowds(s, r)$ in Figure 1 to model an instance of Crowd system where the sender is the jondo s.

A jondo is either an *Initiator*(the real sender) or a *NonInitiator*. The *Initiator* only can *Forward* the request, while the *NonInitiator* can forward the request with probability p_f or submit it with $(1 - p_f)$. When forwarding the request, there are three possibilities: first, the jondo chose another jondo whose index number smaller than his; second,the jondo chose another jondo whose index number larger than his; and third the jondo chose himself. Because choices are made uniformly random, so we model these three possibilities in the process *Forward* with uniform distribution. Jondos in Crowds can initiate and forward the requests many times, which may happen in parallel, and this is the reason why we use the replication operator ! in the processes in Figure 1.

$$Forward(i, req) \overset{\mathsf{def}}{=} \Sigma_{j\in J, i<j}(\tfrac{1}{n})\overline{c_{ij}}\langle req\rangle + \Sigma_{j\in J, j<i}(\tfrac{1}{n})\overline{c_{ji}}\langle req\rangle + (\tfrac{1}{n})SmtOrFwd(i, req)$$
$$SmtOrFwd(i, req) \overset{\mathsf{def}}{=} (p_f)Forward(i, req) + (1-p_f)\overline{c_{ir}}\langle req\rangle$$
$$Initator(i, s) \overset{\mathsf{def}}{=} \nu\, req.Forward(i, req)$$
$$NonInitiator(i) \overset{\mathsf{def}}{=} !(\Pi_{j\in J, i<j}c_{ji}(x).SmtOrFwd(i, x) \mid \Pi_{j\in J, j<i}c_{ji}(x).SmtOrFwd(i, x))$$
$$Server(r) \overset{\mathsf{def}}{=} \Pi_{i\in J}c_{ir}(x)$$
$$Crowds(s, r) \overset{\mathsf{def}}{=} \nu\, C.(Initiator(s) \mid \Pi_{i\in J}NonInitiator(i) \mid Server(r))$$

Fig. 1. The Crowds system in Probabilistic Applied π calculus

5.3 Analyzing

We first consider the simplest case: two jondos (s_1, s_2). In this case the only corrupted party is the server because if one of the two jondos is an attacker, the sender must be another one. As the server is the attacker, we publicize the channels in CR:

$$Crowds^r(s, r) \overset{\mathsf{def}}{=} \nu\, CJ.(Initiator(s) \mid \Pi_{i\in J}NonInitiator(i) \mid Server(r))$$

The sender anonymity degree d_s^r equals to $m(Crowds^r(s_1, r), Crowds^r(s_2, r)$.

Theorem 2. *When there are only two senders, $d_s^r = 0$ for any $p_f \in [0, 1]$, which means in such situation Crowds system satisfies beyond suspicion sender anonymity.*

Theorem 3. *If all the senders are honest, then Crowds system satisfies beyond suspicion sender anonymity for any $p_f \in [0, 1]$.*

This may go against our intuition that p_f should influence the anonymity of crowds. However, in Crowds when an initiator gets a request, he will first choose one jondo with the probability $1/n$ and forward to him. So it is not difficult to interpret such phenomenon.

Let us consider there are n jondos $(s_1, \cdots, s_n)$ and part of them are corrupted parties, w.l.o.g. we may assume the corrupted parties $T = \{s_1, \cdots, s_c, r\}$. We write CK to denote the channels of $s_1, \cdots, s_c$, let $C' = CJ \setminus CK$, then an instance $Crowds^T(s, r)$ is of the following form

$$\nu\, C'.(Initiator(s) \mid \Pi_{i\in J}NonInitiator(i) \mid Server(r))$$

We get some data in Table 1 where n denotes the number of jondos, and c the number of corrupted parties.

We found that for any $n \geq 3$ and $c \geq 1$, if $p_f \leq 0.5$, then the degree $d_s^T > 0.5$, which means Crowds system satisfies possible innocence sender anonymity. If $p_f > 0.5$ and n is large enough, it satisfies probable innocence ($d_s^T < 0.5$). For the same p_f and c, the larger n is, the more anonymous Crowds system is.

Table 1. Seder anonymity of Crowds

Jondos n	**Corrupted jondos** c	**Forwarding probability** p_f	**Degree** d_s^T
3	1	0.5	0.833
3	1	1	0.667
6	1	0.8	0.467
10	2	0.8	0.44
20	2	0.8	0.32

6 Conclusion

In this paper we propose a metric on pairs of processes to measure the degree of difference between them under the framework of PAπ. It decreases to 0 when the two processes are strongly bisimilar. Based on metric, the anonymity degree is defined in PAπ framework. We then analyze the Crowds system.

As to the future work, we will consider applying our approach to analyze some other probabilistic anonymous systems such as some electronic cash, electronic voting protocols. We would also like to develop an automatic tool to calculate metric between any two closed PAπ processes.

References

1. Reiter, M., Rubin, A.: Crowds: anonymity for Web transactions. ACM Transactions on Information and System Security 1(1), 66–92 (1998)
2. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. Auscrypt. 92, 244–251 (1993)
3. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990)
4. Chothia, T.: Analysing the Mute Anonymous File-sharing System Using the Pi Calculus. In: Proceedings of the 26th Conference on Formal Methods for Networked and Distributed Systems. LNCS. Springer, Heidelberg (2006)
5. Deng, Y., Palamidessi, C., Pang, J.: Weak Probabilistic Anonymity. Electronic Notes in Theoretical Computer Science 180(1), 55–76 (2007)
6. Shmatikov, V.: Probabilistic model checking of an anonymity system. Journal of Computer Security 12(3/4), 355–377 (2004)
7. Halpern, J.: Anonymity and Information Hiding in Multiagent Systems. Journal of Computer Security 13(3), 483–514 (2005)
8. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)
9. Lowe, G.: Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. Software - Concepts and Tools 17(3), 93–102 (1996)
10. Abadi, M., Gordon, A.: A Calculus for Cryptographic Protocols: The Spi Calculus. In: Proceedings of 4th ACM Conference on Computer and Communications Security, pp. 36–47 (1997), http://www.research.digital.com/SRC/~abadi

11. Abadi, M., Fournet, C.: Mobile Values, New Names, and Secure Communication. In: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 104–115. ACM Press, New York (2001)
12. Abadi, M., Blanchet, B., Fournet, C.: Just fast keying in the pi calculus. In: Schmidt, D. (ed.) ESOP 2004. LNCS, vol. 2986, pp. 340–354. Springer, Heidelberg (2004)
13. Fournet, C., Abadi, M.: Hiding names: Private authentication in the applied pi calculus. In: Okada, M., Pierce, B.C., Scedrov, A., Tokuda, H., Yonezawa, A. (eds.) ISSS 2002. LNCS, vol. 2609, pp. 317–338. Springer, Heidelberg (2003)
14. Luo, Z., Cai, X., Pang, J., Deng, Y.: Analyzing an electronic cash protocol using applied pi calculus. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 87–103. Springer, Heidelberg (2007)
15. Goubault-Larrecq, J., Palamidessi, C., Troina, A.: A probabilistic applied pi–calculus. In: Shao, Z. (ed.) APLAS 2007. LNCS, vol. 4807, pp. 175–190. Springer, Heidelberg (2007)
16. Deng, Y., Chothia, T., Palamidessi, C., Pang, J.: Metrics for Action-labelled Quantitative Transition Systems. ENTCS 153(2), 79–96 (2006)
17. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Metrics for labelled Markov processes. Theoretical Computer Science 318(3), 323–354 (2004)
18. Desharnais, J., Jagadeesan, R., Gupta, V., Panangaden, P.: The metric analogue of weak bisimulation for probabilistic processes. In: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science, pp. 413–422 (2002)
19. Kremer, S., Ryan, M.D.: Analysis of an electronic voting protocol in the applied pi calculus. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 186–200. Springer, Heidelberg (2005)
20. Pfitzmann, A., Köhntopp, M.: Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 1–9. Springer, Heidelberg (2001)
21. Berthold, O., Langos, H.: Dummy traffic against long term intersection attacks. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 110–128. Springer, Heidelberg (2003)

A Hybrid E-Voting Scheme

Kun Peng

Institute for Infocomm Research, Singapore
dr.kun.peng@gmail.com

Abstract. There are two existing solutions to secure e-voting: homomorphic tallying and shuffling, each of which has its own advantages and disadvantages. The former supports efficient tallying but depends on costly vote validity check and does not support complex elections. The latter supports complex elections and dose not need vote validity check but depends on costly shuffling operations in the tallying operation. In this paper, the two techniques are combined to exploit their advantages and avoid their disadvantages. The resulting e-voting scheme is called hybrid e-voting, which supports complex elections, employs efficient vote validity check and only needs shuffling with a very small scale. So it is more efficient than the existing e-voting schemes, especially in complex elections.

1 Introduction

There are two main solutions to secure electronic voting. The first one is homomorphic tallying, which does not decrypt the encrypted votes separately but exploit homomorphism of the employed encryption algorithm to collectively open the encrypted votes using a small number of decryptions. There are two kinds of homomorphic tallying: additive homomorphic tallying and multiplicative homomorphic tallying. The former [1,16,7,17,18,12] employs an additive homomorphic encryption algorithm[1] like Paillier encryption and recovers the sum of the voters' selections. The latter [21] employs an multiplicative homomorphic encryption algorithm[2] like ElGamal encryption and recovers the product of the voters' selections. With homomorphic tallying, each vote must be in a special format, so that the number of every possible selection can be counted. More precisely, with additive homomorphic tallying every vote contains one or more selections (each corresponding to a candidate or a possible choice) and every selection must be one of two pre-defined integers (e.g. 0 and 1), each representing support or rejection of a candidate or choice; with multiplicative homomorphic tallying every vote contains only one selection, which corresponds to the chosen candidate or preferred choice and must be one of a few pre-defined integers (1 or e.g. small primes), each representing a candidate or choice. With such special vote formats,

[1] An encryption algorithm with decryption function $D()$ is additive homomorphic if $D(c_1) + D(c_2) = D(c_1c_2)$ for any ciphertexts c_1 and c_2.

[2] An encryption algorithm with decryption function $D()$ is multiplicative homomorphic if $D(c_1)D(c_2) = D(c_1c_2)$ for any ciphertexts c_1 and c_2.

usually the election rule is not complex with homomorphic tallying. When the number of candidates or possible choices is large, additive homomorphic tallying needs a large number of selections in a vote and thus compromises efficiency while the multiplicative homomorphic tallying must use large integers to represent the selections and thus the product of the selections easily overflows the multiplicative modulus. For example, in many political elections (e.g. in Australia) there are multiple candidates and each vote must indicate a complete preferential order of all the candidates. In such preferential voting applications, the existing homomorphic tallying solutions are impractical. Another drawback of homomorphic tallying is also caused by special vote format: each vote must be publicly checked to be valid (in a certain special format). Vote validity check usually depends on costly zero knowledge proof operations and corresponding verification operations.

In this paper, we are more interested in multiplicative homomorphic tallying [21], which is much less well-known than additive homomorphic tallying but has much simpler vote format. Besides simpler vote format and therefore more efficient vote sealing, multiplicative homomorphic tallying has another advantage over additive homomorphic tallying: it employs encryption algorithms depending on hardness of discrete logarithm problem like ElGamal encryption instead of the encryption algorithms depending on hardness of factorization problem. More precisely, there is no efficient method to implement key generation in a distributed way for any practical additive homomorphic encryption algorithm (e.g. Paillier encryption, all of which use factorization problem as a trapdoor) while centralized key generation for them through a trusted dealer [10] requires too strong a trust and is impractical in applications like e-voting. Note that although a modified ElGamal encryption in [18] is additive homomorphic, it is not practical in most applications as it does not support efficient decryption. There exist some distributed key generation mechanisms for RSA [3,23,8], which is also factorization based. However, they (especially [3,23]) are inefficient. [8] improves efficiency to some extent by loosening the requirements on the parameters and using additional security assumptions, but is still inefficient compared to distributed key generation of DL based encryption algorithms like ElGamal [9]. So they cannot provide an efficient solution to distributed key generation for additive homomorphic encryption, not to mention the relatively more efficient mechanism among them may not satisfy the parameter requirements with reasonable security assumptions when applied to additive homomorphic encryption based e-voting. In comparison, distributed key generation for multiplicative homomorphic encryption (e.g. ElGamal encryption) can be implemented very efficiently using the technique in [9].

With multiplicative homomorphic tallying, the vote space contains 1 and some small primes and each vote contains one integer in the vote space. In the tallying phase, product of all the votes are recovered and then factorized to re-construct the votes. A key point is that the product cannot overflow the multiplicative modulus, otherwise factorization fails. For example, when there are 10 possible choices in the election, even if the vote space contains 1 and the nine smallest primes, a vote selection can still be as large as 23. In the current security standard, the multiplicative modulus is usually 1024 bits long when ElGamal encryption is employed. That means even if there are only several hundred voters the product of their votes will probably overflow the multiplicative modulus. There are two countermeasures in [21] to avoid the overflow. Firstly, the election must be simple and thus the vote space only contains very small primes. For example, the election application in [21] only support YES/NO election and thus the vote space only contains 1 and 2. Secondly, the votes are divided into multiple groups and the size of the groups are so small that the product of the votes in each group will not overflow the multiplicative modulus. A factorization is needed in each group to reconstruct the votes. There are two concerns in the existing multiplicative homomorphic tallying [21]. Firstly, the supported election rule is too simple and not suitable for many election applications. Secondly, its grouping mechanism weakens privacy of e-voting as each voter is known to have cast one of the recovered votes in his group. Unlike other e-voting schemes (which hide every voter's choice among all the recovered votes), the multiplicative-homomorphic-tallying-based e-voting scheme in [21] only achieves weaker and incomplete privacy.

Due to the drawbacks of homomorphic tallying, shuffling [11,19,14,22,15,20,13] is often employed in e-voting when complex election rules like preferential election is used. A shuffling operation re-encrypts (or partially decrypts) the encrypted votes and re-orders them. Multiple shuffling operations are employed and each of them is performed by a different tallier such that the votes are untraceable if at least one tallier is honest. Finally, the repeatedly shuffled votes are decrypted to recover all the votes. Even if one of the most efficient shuffling primitives with satisfactory security [11,15,20,13] is employed, in every shuffling operation for each vote several exponentiations are needed for re-encryption and several exponentiations are needed for public proof of validity of shuffling. So the shuffling operations in e-voting are costly, especially when the number of voters is large.

In this paper, multiplicative homomorphic tallying and shuffling are modified respectively so that they can be combined to exploit their advantages and avoid their disadvantages. On one hand, the vote space of multiplicative homomorphic tallying is enlarged to hold all the possible choices for a voter even in a complex election. For example, when there are several candidates and preferential election is used, there are several hundred possible choices for a voter. In this case, we can use the 10-bit primes and with a 1024-bit multiplicative modulus each group for factorization operation can contain around 100 votes. With this modification, each vote only contains one integer even if preferential election is used and only one decryption is needed in each group to recover the votes. On the other hand, to prevent the grouping mechanism with small group size from weakening privacy a modified shuffling mechanism is employed to shuffle the groups (instead of the votes). Any existing shuffling protocol supporting ElGamal encryption can be employed on the conditions that it is properly modified to suit the new application. Firstly, it only shuffles the groups. Secondly, as shown in Section 4.1, recovery of the shuffled messages in the end of the shuffling and analysis of soundness needs adjusting. As the number of groups is much smaller than the number of votes (e.g. the former may be 1% of the latter), the modified shuffling is very efficient. In the new e-voting scheme, vote validity check is still needed to guarantee that every vote is in the vote space such that the factorization operations can recover the votes. As the group size is not very small, vote validity check is costly if existing techniques are employed to implement it. To solve this problem, a new vote validity check technique is designed for the voters to efficiently prove validity of their votes.

The resulting e-voting scheme is called hybrid e-voting, which supports simple vote format, efficient vote sealing, complex elections and efficient distributed key generation, employs efficient vote validity check and only needs shuffling with a very small scale. So it is more efficient than existing e-voting schemes, especially in complex elections. A property sometimes desired in e-voting, receipt-freeness (or called coercion-resistance in some literature), is not the focus of this paper, so is not discussed in detail. Either of the two existing solutions to receipt-freeness, deniable encryption [4] and re-encryption with untransferable zero knowledge proof of correctness by a third party (in the form of a trusted authority or a tamper-resistent hardware) [18] can be employed to achieve it if needed.

2 Parameters and Denotions

The symbols and notations to be used in this paper are listed as follows and in Table 1.

Table 1. Notations

P	the prover in a proof protocol.
V	the verifier in a proof protocol.
$a \in_R S$	a is a integer randomly chosen from set S.
$\lfloor x \rfloor$	the largest integer no larger than x.
$\lceil x \rceil$	the smallest integer no smaller than x.
$E()$, $RE()$ and $D()$	encryption, re-encryption and decryption respectively.
$P\ [\ x_1, x_2, \ldots, x_n \mid A(x_1, x_2, \ldots, x_n)\ ]$	the probability of event A distributed over variants $x_1, x_2, \ldots, x_n$.

For simplicity, it is supposed that when ElGamal encryption is employed it is as follows.

- Parameter setting
  A security parameter K is chosen (e.g. set to be 1024) and a K-bit prime p is chosen such that $p-1 = 2q$ and q is a large prime. G is the cyclic subgroup in Z_p with order q and g is a generator of G.
- Key setting
  The private key is an integer x in Z_q. It is generated in a distributed way using the key generation technique in [9] such that it is shared by multiple parties (e.g talliers in e-voting). x can be reconstructed only if the number

of cooperating share holders is over a certain threshold. The public key is $y = g^x \bmod p$.
- Encryption
  A message M is encrypted into $E(M) = (a, b) = (g^r \bmod p,\ My^r \bmod p)$ where r is randomly chosen from Z_q.
- Re-encryption
  A ciphertext $u = (a, b)$ is re-encrypted into $RE(u) = (a', b') = (ag^{r'} \bmod p,\ by^{r'} \bmod p)$ where r' is randomly chosen from Z_q.
- Decryption
  When the the number of cooperating share holders is over the threshold, given a ciphertext $u = (a, b)$, they cooperate to calculate $D(u) = b/a^x \bmod p$.
- Product of ciphertexts
  $u_1u_2 = (a_1a_2, b_1, b_2)$ where $u_1 = (a_1, b_1)$ and $u_2 = (a_2, b_2)$.

3 Background

Important cryptographic primitives in existing e-voting schemes are recalled in this paper. In comparison with them, more advanced and efficient designs will be employed in this paper.

3.1 Shuffling in E-Voting

In a shuffling protocol, a shuffling node re-encrypts and reorders multiple input ciphertexts to some output ciphertexts such that the messages encrypted in the output ciphertexts are a permutation of the messages encrypted in the input ciphertexts. The shuffling node has to publicly prove validity of its shuffling. Shuffling is usually employed to build up anonymous communication channels and its most important application is e-voting. Suppose the input ciphertexts are $u_1, u_2, \ldots, u_n$ are inputs to a shuffling node, who outputs ciphertexts $u'_1, u'_2, \ldots, u'_n$ where $u'_i = RE(u_{\pi(i)})$ and $\pi()$ is a secret permutation of $1, 2, \ldots, n$. The shuffling node has to publicly prove that $D(u'_1), D(u'_2), \ldots, D(u'_n)$ is a permutation of $D(u_1), D(u_2), \ldots, D(u_n)$ without revealing $\pi()$.

Even in the most computationally efficient shuffling protocols [20,13], re-encryption of each ciphertext costs several exponentiations and zero knowledge proof of validity of shuffling of n ciphertexts cost $O(n)$ exponentiations. When n is large, it is a high cost, not to mention multiple instances of shuffling are needed in e-voting.

3.2 Vote Validity Check

Suppose the vote space is $S = \{p_1, p_2, \ldots, p_m\}$ and a voter submits an ElGamal ciphertext $u = (a, b) = (g^r \bmod p,\ My^r \bmod p)$ as his encrypted vote, he has to publicly prove that $D(u)$ is in S without revealing it. The normal technique to implement this proof is a special zero knowledge proof primitive called "proofs of partial knowledge" [6] detailed in Figure 1.

For simplicity, suppose $D(u) = p_m$.

1. $P \rightarrow V$: $a_i = g^{w_i} a^{c_i} \bmod p$ for $i = 1, 2, \ldots, m-1$
   $b_i = y^{w_i} (b/p_i)^{c_i} \bmod p$ for $i = 1, 2, \ldots, m-1$
   $a_m = g^s \bmod p$
   $b_m = y^s \bmod p$
   where $c_i \in_R Z_q$ for $i = 1, 2, \ldots, m-1$, $w_i \in_R Z_q$ for $i = 1, 2, \ldots, m-1$ and $s \in_R Z_q$.
2. $V \rightarrow P$: $c \in_R Z_q$
3. $P \rightarrow V$: $c_1, c_2, \ldots, c_m, \quad w_1, w_2, \ldots, w_m$
   where $w_m = s - c_m r \bmod q$ and $c_m = c - \sum_{i=1}^{m-1} c_i \bmod q$.

Verification: $a_i = g^{w_i} a^{c_i} \bmod p$ for $i = 1, 2, \ldots, m$
$b_i = y^{w_i} (b/p_i)^{c_i} \bmod p$ for $i = 1, 2, \ldots, m$
$c = \sum_{i=1}^{m} c_i \bmod q$

Fig. 1. ZK proof and verification of a vote encrypted with ElGamal

This solution is highly inefficient. It costs $O(m)$ full-length exponentiations. When it is applied to vote validity check in an e-voting with n voters, the cost for vote validity check is $O(nm)$. In a large scale election with a large number of voters and a larger number of choices for them, it is a very high cost. Especially, it is too costly for the common people to verify validity of the votes and correctness of the election result.

4 The New Solution

The new solution is a hybrid technique, which modifies and combines multiplicative homomorphic tallying and shuffling to exploit their advantages and avoid their disadvantages. Moreover, an efficient vote validity check mechanism is proposed to improve efficiency.

4.1 Combination of Multiplicative Homomorphic Tallying and Shuffling

Multiplicative homomorphic tallying and shuffling are modified and combined to design a new e-voting scheme as follows.

1. Determining the vote space
   According to the election rule, the number of possible choices for a voter is calculated. For example, in a preferential election with l candidates, there are $l!$ possible choices. When $l = 5$ or 6 as in many practical preferential elections, there are several hundred possible choices. Suppose there are m possible choices in the election application, the vote space $S = \{p_1, p_2, \ldots, p_m\}$ contains 1 and the $m-1$ smallest odd primes, each standing for a choice.

2. Vote submission
   Suppose there are n voters. Each voter chooses his vote v_i from S and submit $u_i = (a_i, b_i) = (g^{r_i}, v_i y^{r_i})$ where r_i is randomly chosen from Z_q. Each voter proves validity of his vote using batch proof and verification.
3. Grouping the encrypted votes
   Suppose $p_1 = 1$ and p_j is the $j-1^{th}$ smallest odd primes in S. The size of each group is $\alpha = \lfloor \log_{p_m} p \rfloor$, so that the product of the votes in any group will not overflow p. The n encrypted votes are divided into $\beta = \lceil n/\alpha \rceil$ groups, while u_i is in the $(\lfloor i/\alpha \rfloor + 1)^{th}$ group.
4. Shuffling the groups
   The β groups are denoted as $G_1, G_2, \ldots, G_\beta$. For $j = 1, 2, \ldots, \beta$, the talliers calculate $U_j = (A_j, B_j) = (\prod_{u_i \in G_j} a_i,\ \prod_{u_i \in G_j} b_i)$. $U_1, U_2, \ldots, U_\beta$ are repeatedly shuffled by multiple talliers and the last shuffling operation outputs $U'_1, U'_2, \ldots, U'_\beta$. Any efficient shuffling protocol supporting ElGamal encryption (e.g. [14,13]) can be employed to shuffle the groups as follows.
   (a) After a shuffling tallier receives ciphertexts $W_1, W_2, \ldots, W_\beta$, each representing a group shuffled by the last shuffling tallier, he re-encrypts and re-orders them and output ciphertexts $W'_1, W'_2, \ldots, W'_\beta$ such that $W'_j = RE(W_{\pi(j)})$ for $j = 1, 2, \ldots, \beta$ where $\pi()$ is a permutation of $1, 2, \ldots, \beta$ chosen by the tallier.
   (b) After $W'_1, W'_2, \ldots, W'_\beta$ are published, random θ-bit challenges $t_1, t_2, \ldots, t_\beta$ are chosen (e.g. by pseudo-random functions like hash function or by multiple parties with a threshold trust on them) where $2^\theta \leq q$.
   (c) The shuffling tallier proves that

$$\prod_{j=1}^{\beta} W_j^{t_j} = \prod_{j=1}^{\beta} {W'}_j^{t_{\pi(j)}} \bmod p \tag{1}$$

   and the permutation $\pi()$ is committed and unchanged using a zero knowledge protocol (see [13] for more details).
   (d) A verifier verifies the proof of (1) and is convinced that the shuffling is valid if and only if (1) is proved to be satisfied. Unfortunately, in the existing shuffling protocols employing this idea (e.g [14,13]) it is not formally explained why (1) guarantees that $D(W'_1), D(W'_2), \ldots, D(W'_\beta)$ is a permutation of $D(W_1), D(W_2), \ldots, D(W_\beta)$. When ElGamal encryption is employed like in our application, (1) actually implies that

$$\prod_{j=1}^{\beta} D(W_j)^{t_j} = \prod_{j=1}^{\beta} D({W'}_j)^{t_{\pi(j)}} \bmod p. \tag{2}$$

   Although according to the formal security analysis in [20] $D(W'_1), D(W'_2), \ldots, D(W'_\beta)$ must be a permutation of $D(W_1), D(W_2), \ldots, D(W_\beta)$ when $\sum_{j=1}^{\beta} D(W_j)t_j = \sum_{j=1}^{\beta} D({W'}_j)t_{\pi(j)}$ is satisfied, it only covers shuffling employing additive homomorphic encryption algorithms like Paillier encryption. As multiplicative homomorphic tallying is exploited in the new e-voting scheme, multiplicative homomorphic encryption algorithms like ElGamal encryption must be employed and only (2) can be guaranteed by the shuffling protocol. To the best of our knowledge, there is no existing formal analysis (in [14,13] or other schemes) to guarantee that (2) implies correctness of shuffling. Actually, satisfaction of (2) cannot guarantee that $D(W'_1), D(W'_2), \ldots, D(W'_\beta)$ is a permutation of $D(W_1), D(W_2), \ldots, D(W_\beta)$. As formally illustrated in Theorem 1, satisfaction of (2) only guarantees that $|D(W'_1)|, |D(W'_2))|, \ldots, |D(W'_\beta)|$ is a permutation of $D(W_1), D(W_2), \ldots, D(W_\beta)$ where function $| \; |$ is defined as follows.

Definition 1. *For an integer x in Z_p, $|x| = x$ if x is factorized into α primes in the vote space where α is the group size; otherwise, $|x| = p - x$.*

5. Decryption and factorization
The talliers cooperate to decrypt $U'_1, U'_2, \ldots, U'_\beta$. For $j = 1, 2, \ldots, \beta$, $V_j = D(U'_j)$. Each participating tallier proves that his part of decryption is correct using zero knowledge proof of equality of discrete logarithms [5]. Then $V_1, V_2, \ldots, V_\beta$ are factorized as follows.
   (a) Factorize each group represented by V_j for $j = 1, 2, \ldots, \beta$. According to Theorem 1, either V_j or $p - V_j$ should be factorized. Suppose α is the group size. If V_j is factorized into α primes in the vote space, V_j is the product of the votes in a shuffled group and the factorization is accepted. Otherwise, the product of the votes in a shuffled group should be recovered as $p - V_j$, which is then factorized. As p is an odd prime, it is impossible that both V_j and $p - V_j$ are factorized into only the odd primes in the vote space, so the factorization function is decisional.
   (b) The number of votes choosing p_2 is counted as the number of p_2 found in the β factorizations; the number of votes choosing p_3 is counted as the number of p_3 found in the β factorizations; ......; the number of votes choosing p_m is counted as the number of p_m found in the β factorizations. The number of votes choosing p_1 is counted as n minus the sum of other choices.

Theorem 1. *Suppose $x_1, x_2, \ldots, x_\beta$ are integers in Z_p^* and $y_1, y_2, \ldots, y_\beta$ are primes in Z_p^* and $t_1, t_2, \ldots, t_\beta$ are θ-bit integers. If $\prod_{j=1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\beta} y_j^{t_{\pi(j)}} \bmod p$ with a probability larger than $2^{-\theta}$ and $\pi()$ is a committed and unchanged permutation of $1, 2, \ldots, \beta$, then there is a choice of implementation of $[\;]$ for each $[x_j]$ such that $[x_1], [x_2], \ldots, [x_\beta]$ is a permutation of $y_1, y_2, \ldots, y_\beta$ where $[x_j] = x_j$ or $p - x_j$.*

Proof. Given any integer γ in $\{1, 2, \ldots, \beta\}$, there must exist integers $t_1, t_2, \ldots, t_{\gamma-1}, t_{\gamma+1}, \ldots, t_n$ in $\{0, 1, \ldots, 2^\theta - 1\}$ and two different integers t_γ and $\hat{t}_\gamma$ in $\{0, 1, \ldots, 2^\theta - 1\}$ such that the following two equations are correct.

$$\prod_{j=1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\beta} y_j^{t_{\pi(j)}} \bmod p \tag{3}$$

$$\prod_{j=1}^{\gamma-1} x_j^{t_j} + x_\gamma^{\hat{t}_\gamma} + \prod_{j=\gamma+1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\gamma-1} y_j^{t_{\pi(j)}} + y_\gamma^{\hat{t}_{\pi(\gamma)}} + \prod_{j=\gamma+1}^{\beta} y_j^{t_{\pi(j)}} \bmod p \tag{4}$$

Otherwise, for any combination of $t_1, t_2, \ldots, t_{\gamma-1}, t_{\gamma+1}, \ldots, t_\beta$ there is at most one t_γ to satisfy equation $\prod_{j=1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\beta} y_j^{t_{\pi(j)}} \bmod p$. This deduction implies that among the $2^{\beta\theta}$ possible combinations of $t_1, t_2, \ldots, t_\beta$, equation $\prod_{j=1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\beta} y_j^{t_{\pi(j)}} \bmod p$ is correct for at most $2^{(\beta-1)\theta}$ combinations. This conclusion leads to a contradiction: given random integers t_j from $\{0, 1, \ldots, 2^\theta - 1\}$ for $j = 1, 2, \ldots, \beta$ and $\pi()$, a committed and unchanged permutation of $j = 1, 2, \ldots, \beta$, equation $\prod_{j=1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\beta} y_j^{t_{\pi(j)}} \bmod p$ is correct with a probability no larger than $2^{-\beta}$.

(3) and (4) respectively implies

$$\prod_{j=1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\beta} y_{\pi^{-1}(j)}^{t_j} \bmod p \tag{5}$$

$$\prod_{j=1}^{\gamma-1} x_j^{t_j} + x_\gamma^{\hat{t}_\gamma} + \prod_{j=\gamma+1}^{\beta} x_j^{t_j} = \prod_{j=1}^{\gamma-1} y_{\pi^{-1}(j)}^{t_j} + y_{\pi^{-1}(\gamma)}^{\hat{t_\gamma}} + \prod_{j=\gamma+1}^{\beta} y_{\pi^{-1}(j)}^{t_j} \bmod p \tag{6}$$

(5) divided by (6) yields

$$x_\gamma^{t_\gamma - \hat{t}_\gamma} = y_{\pi^{-1}(\gamma)}^{t_\gamma - \hat{t}_\gamma} \bmod p$$

Namely

$$(x_\gamma / y_{\pi^{-1}(\gamma)})^{t_\gamma - \hat{t}_\gamma} = 1 \bmod p$$

As x_γ and $y_{\pi^{-1}(\gamma)}$ are in Z_p^*, $x_\gamma / y_{\pi^{-1}(\gamma)}$ is in Z_p^* as well. As $t_1, t_2, \ldots, t_\beta$ are θ-bit integers and $2^\theta \le q$, the absolute value of $t_\gamma - \hat{t}_\gamma$ is smaller than q. So, as $p-1$ has only two dividers, 2 and q,

$$x_\gamma / y_{\pi^{-1}(\gamma)} = \pm 1 \bmod p$$

Namely,

$$x_\gamma = \pm y_{\pi^{-1}(\gamma)} \bmod p$$

Note that γ can be any integer in $\{1, 2, \ldots, \beta\}$. Therefore, there is a choice of implementation of [ ] for each $[x_j]$ such that $[x_1], [x_2], \ldots, [x_\beta]$ is a permutation of $y_1, y_2, \ldots, y_\beta$. $\square$

Theorem 1 guarantees that any valid vote can be correctly recovered, while the privacy functionality of the shuffling guarantees that no matter how small the group size is the grouping mechanism does not compromise privacy of the e-voting scheme.

5 Properties and Comparison

In the new e-voting scheme, both homomorphic tallying and shuffling are employed. However, both techniques are optimised. In homomorphic tallying, efficiency of vote validity check is improved. When L is 40, soundness is strong enough and only a small constant number of full length exponentiations are needed. Bellare *et al* [2] have illustrated that an exponentiation with a 40-bit exponent is much more efficient than a full length exponentiation. So a satisfactory trade-off between security and efficiency is achieved. Although shuffling is employed in the new e-voting scheme, it is in a much smaller scale than in the existing application of shuffling to e-voting. For example, in a normal case the number of possible choices is no more than several hundred in an election. In this case, the primes in the vote space are no larger than a couple of thousand and with a 1024-bit multiplicative modulus each group for factorization operation can contain around 100 votes. So usually the number of groups is much smaller than the number of votes (e.g. the former may be 1% of the latter). As the cost of shuffling operation (including re-encryption and proof or validity) is linear in the number of shuffled objects, the the modified shuffling mechanism is much more efficient than the existing shuffling protocols in e-voting.

Although a grouping mechanism is employed in the new e-voting scheme, it does not compromise privacy of the election as all the groups are shuffled before the votes in them are recovered through collective decryption and factorization. If at lease one of the shuffling talliers is honest, the groups are untraceable and no vote is known to be in any group. As semantically secure encryption is used and the employed shuffling protocol and the vote validity check mechanism achieve zero knowledge privacy, privacy of the new e-voting scheme is strong. The advantages of the new e-voting scheme over the existing e-voting schemes are demonstrated in Table 2. It is clearly illustrated that the new e-voting scheme is secure, rule-flexible and more efficient.

Table 2. Comparison of e-voting schemes

e-voting	election rule	distributed key generation	vote sealing	vote validity check	shuffling
additive homomorphic	simple rule only	inefficient	inefficient	inefficient	no need
multiplicative homomorphic	simple rule only	efficient	efficient	inefficient	no need
shuffling based	supporting complex rule	can be efficient	efficient	no need	inefficient
new hybrid scheme	supporting complex rule	efficient	efficient	efficient	efficient

6 Conclusion

The new e-voting scheme proposed in this paper is a hybrid solution combining multiplicative homomorphic tallying and shuffling. It exploits advantages of the two techniques and avoids their disadvantages. As a result, it has a lot of merits. It supports simple vote format, efficient vote sealing, complex elections and efficient distributed key generation. It employs efficient vote validity check and only needs shuffling with a very small scale. So it is more efficient than existing e-voting schemes, especially in complex elections.

References

1. Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Twentieth Annual ACM Symposium on Principles of Distributed Computing, pp. 274–283 (2001)
2. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
3. Boneh, D., Franklin, M.: Efficient generation of shared rsa keys. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 425–439. Springer, Heidelberg (1997)
4. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
5. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
6. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
7. Damgaård, 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)
8. Damgård, I.B., Koprowski, M.: Practical threshold RSA signatures without a trusted dealer. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 152–165. Springer, Heidelberg (2001)
9. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science, pp. 427–437 (1987)
10. 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)
11. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
12. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
13. Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 377–392. Springer, Heidelberg (2007)
14. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002)
15. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Transactions 88-A(1), 172–188 (2005)
16. Katz, J., Myers, S., Ostrovsky, R.: Cryptographic counters and applications to electronic voting. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 78–92. Springer, Heidelberg (2001)
17. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 141–158. Springer, Heidelberg (2002)

18. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
19. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security 2001, pp. 116–125 (2001)
20. Peng, K., Boyd, C., Dawson, E.: Simple and efficient shuffling with provable correctness and ZK privacy. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 188–204. Springer, Heidelberg (2005)
21. Peng, K., Aditya, R., Boyd, C., Dawson, E., Lee, B.: Multiplicative homomorphic E-voting. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 61–72. Springer, Heidelberg (2004)
22. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.: A correct, private, and efficient mix network. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 439–454. Springer, Heidelberg (2004)
23. MacKenzie, P., Frankel, Y., Yung, M.: Robust efficient distributed rsa-key generation. p. 320 (1998)

A Framework for Authenticated Key Exchange in the Standard Model*

Shuhua Wu and Yuefei Zhu

Department of Networks Engineering,
Zhengzhou Information Science Technology Institute,
Zhengzhou 450002, China
wushuhua726@sina.com.cn

Abstract. We first introduce the new notion of the so-called target-independent smooth projective hashing (TISPHash) based on computationally-hiding commitments. Based on it and a class of pseudo-random functions (PRFs), we propose a framework for (PKI-based) authenticated key exchange protocols without random oracles and prove it to be secure in the (currently) strongest security definition, the extended Canetti-Krawczyk security definition. Our protocol is actually an abstraction of the efficient key exchange protocol of T. Okamoto. The abstracted protocol enjoys efficient instantiations from any secure encryption scheme that admits an efficient construction of TISPHash and allows a simple and intuitive understanding of its security. In some sense, our construction generalizes the design of T. Okamoto.

Keywords: authenticated key exchange, Standard Model.

1 Introduction

A central problem in cryptography is that of enabling parties to communicate secretly and reliably in the presence of an adversary. This is often achieved by having the parties run a protocol for generating a mutual and secret session key (*authenticated key exchange*). During the last decades, the design of 2-party authenticated key exchange(AKE) has been explored intensively. Although some efficient two-pass AKE protocols such as HMQV[1], NAXOS and CMQV[2] were presented in the literature, they usually achieved provable security in the random oracle model. In this model, all parties are assumed to have oracle access to a totally random (universal) function [3]. The common interpretation of such results is that security is likely to hold even if the random oracle is replaced by a ("reasonable") concrete function known explicitly to all parties (e.g., SHA-2). However, it has been shown that it is impossible to replace the random oracle in a generic manner with any concrete function [4]. Thus, the proofs of security of these protocols are actually heuristic in nature. Therefore designing provably secure AKE without random oracles get much attention in the current research.

* This work was partially supported by a grant from the National High Technology Research and Development Program of China (863 Program) (No. 2007AA01Z471).

Most recently, T. Okamoto presented a new paradigm to realize AKE without random oracles under three assumptions: the decisional Diffie-Hellman (DDH) assumption, target collision resistant (TCR) hash functions and a class of pseudo-random functions (PRFs), πPRFs (PRFs with pairwise-independent random sources)[5]. They proposed a (PKI-based) AKE protocol that was comparably as efficient as the existing most efficient protocols like MQV[6] and that was secure without random oracles (under these assumptions). Moreover, their protocol achieved provable security in the (currently) strongest security definition, the extended Canetti-Krawczyk (eCK) security definition introduced by LaMacchia, Lauter and Mityagin[7].

In this paper, we first introduce the new notion of the so-called target-independent smooth projective hashing(TISPHash) based on computationally-hiding commitments. Based on it and πPRF, we propose a framework for (PKI-based) AKE protocols without random oracles and prove it to be secure in the (currently) strongest security definition, the extended Canetti-Krawczyk (eCK) security definition. Our protocol is actually an abstraction of the key exchange protocol of T. Okamoto[5]. The abstracted protocol enjoys efficient instantiations from any secure encryption scheme that admits an efficient construction of TISPHash and allows a simple and intuitive understanding of its security. In some sense, our construction generalizes the design of [5]. Finally, to provide the reader with an idea of how efficient target-independent smooth projective hash functions are, we give an example of TISPHash based on the El-Gamal encryption scheme. A protocol that is almost identical to that of [5] can be derived when our protocol is instantiated with it.

2 Extended Canetti-Krawczyk Security Definition [5]

This section outlines the extended Canetti-Krawczyk (eCK) security definition for two pass PKI-based AKE protocols that was introduced by LaMacchia, Lauter and Mityagin [7], and follows the description in [2,5].

In the eCK definition, we suppose there are n parties which are modeled as probabilistic polynomial-time Turing machines.We assume that some agreement on the common parameters in the AKE protocol has been made among the parties before starting the protocol. The mechanism by which these parameters are selected is out of scope of the AKE protocol and the (eCK) security model.

Each party has a static public-private key pair together with a certificate that binds the public key to that party. $\hat{A}(\hat{B})$ denotes the static public key $A(B)$ of party $\mathcal{A}(\mathcal{B})$ together with a certificate. We do not assume that the certifying authority (CA) requires parties to prove possession of their static private keys, but we require that the CA verifies that the static public key of a party belongs to the domain of public keys.

Here, two parties exchange static public keys A, B and ephemeral public keys X, Y ; the session key is obtained by combining A, B, X, Y and possibly session identities. A party $\mathcal{A}$ can be activated to execute an instance of the protocol called a session. Activation is made via an incoming message that has one of the following forms: $(\hat{A}, \hat{B})$ or $(\hat{B}, \hat{A}, X)$. If $\mathcal{A}$ was activated with $(\hat{A}, \hat{B})$, then $\mathcal{A}$ is called the session initiator, otherwise the session responder. Session initiator $\mathcal{A}$ creates ephemeral public-private key pair, (X, x) and sends $(\hat{B}, \hat{A}, X)$ to session responder $\mathcal{B}$. $\mathcal{B}$ then creates ephemeral public-private key pair, (Y, y) and sends $(\hat{A}, \hat{B}, X, Y)$ to $\mathcal{A}$.

The session of initiator $\mathcal{A}$ with responder $\mathcal{B}$ is identified via session identifier $(\hat{A}, \hat{B}, X, Y)$, where $\mathcal{A}$ is said the owner of the session, and $\mathcal{B}$ the peer of the session. The session of responder $\mathcal{B}$ with initiator $\mathcal{A}$ is identified as $(\hat{B}, \hat{A}, Y, X)$, where $\mathcal{B}$ is the owner, and $\mathcal{A}$ is the peer. Session $(\hat{B}, \hat{A}, Y, X)$ is said a matching session of $(\hat{A}, \hat{B}, X, Y)$. We say that a session is completed if its owner computes a session key.

The adversary $\mathcal{M}$ is modeled as a probabilistic polynomial-time Turing machine and controls all communications. Parties submit outgoing messages to the adversary, who makes decisions about their delivery. The adversary presents parties with incoming messages via Send(*message*), thereby controlling the activation of sessions. In order to capture possible leakage of private information, adversary $\mathcal{M}$ is allowed the following queries:

– EphemeralKeyReveal(*sid*) The adversary obtains the ephemeral private key associated with session *sid*.
– SessionKeyReveal(*sid*) The adversary obtains the session key for session *sid*, provided that the session holds a session key.
– StaticKeyReveal(*pid*) The adversary learns the static private key of party *pid*.
– EstablishParty(*pid*) This query allows the adversary to register a static public key on behalf of party. In this way the adversary totally controls that party.

If party *pid* is established by EstablishParty(*pid*) query issued by adversary $\mathcal{M}$, then we call the party dishonest. If party is not dishonest, we call the party honest.

The aim of adversary $\mathcal{M}$ is to distinguish a session key from a random key. Formally, the adversary is allowed to make a special query Test(*sid**), where *sid** is called the target session. The adversary is then given with equal probability either the session key held by *sid** or a random key of equal length. The adversary wins the game if he guesses correctly whether the key is random or not. To define the game, we need the notion of fresh session as follows:

Definition 1. *(fresh session) Let sid be the session identifier of a completed session, owned by an honest party $\mathcal{A}$ with peer $\mathcal{B}$, who is also honest. Let $\overline{sid}$ be the session identifier of the matching session of sid, if it exists. Define session sid to be fresh if none of the following conditions hold:*

– $\mathcal{M}$ *issues a* SessionKeyReveal(*sid*) *query or a* SessionKeyReveal($\overline{sid}$) *query (if $\overline{sid}$ exists),*
– $\overline{sid}$ *exists and $\mathcal{M}$ makes either of the following queries:*
both StaticKeyReveal($\mathcal{A}$) *and* EphemeralKeyReveal(*sid*)*, or*
both StaticKeyReveal($\mathcal{B}$) *and* EphemeralKeyReveal($\overline{sid}$)*,*

– $\overline{sid}$ *does not exist and* $\mathcal{M}$ *makes either of the following queries: both* StaticKeyReveal($\mathcal{A}$) *and* EphemeralKeyReveal(sid)*, or* StaticKeyReveal($\mathcal{B}$)*.*

We are now ready to present the eCK security notion.

Definition 2. *(eCK security) Let* sk^* *be a session key of the target session* sid^* *that should be "fresh", and* b^* *is a bit chosen uniformly at random in* $\{0,1\}$*. As a reply to Test(*sid^**) query by* $\mathcal{M}$*,* sk^* *is given to* $\mathcal{M}$ *if* $b^* = 0$*; a random key of the same size is given otherwise. Finally* $\mathcal{M}$ *guesses the bit* b^* *and outputs this guess* b*. We define* $AdvAKE_{\mathcal{M}}(l) = 2\left|Pr[b = b^*] - \frac{1}{2}\right|$.

A key exchange protocol is secure if the following conditions hold:

– *If two honest parties complete matching sessions, then they both compute the same session key (or both output indication of protocol failure).*
– *For any probabilistic polynomial-time adversary* $\mathcal{M}$*,* $AdvAKE_{\mathcal{M}}(l)$ *is negligible in* l*.*

This security definition is stronger than CK-security [8] and it simultaneously captures all the known desirable security properties for authenticated key exchange including resistance to key-compromise impersonation attacks, weak perfect forward secrecy, and resilience to the leakage of ephemeral private keys.

3 Decisional Diffie-Hellman Assumption-DDH

Let G be a large cyclic group of prime order p. We consider the following two distributions:

– Given a Diffie-Hellman quadruple g, g^x, g^y and g^{xy}, where $x, y \in Z_p$, are random strings chosen uniformly at random;
– Given a random quadruple g, g^x, g^y and g^r, where $x, y, r \in Z_p$, are random strings chosen uniformly at random.

An algorithm that solves the Decisional Diffie-Hellman problem is a statistical test that can efficiently distinguish these two distributions. Decisional Diffie-Hellman assumption means that there is no such a polynomial statistical test.

4 Basic Tools

The design of our protocol mainly builds on three basic tools: πPRFs, computationally-hiding commitments and TISPHash, where the so-called TISPHash is first introduced by us and is a specific class of smooth projective hashing recently introduced by Cramer and Shoup [9]. Our usage of these tools is to a large extent inherited from [10]. In this section we review the main definitions and results necessary for the sequel.

4.1 Computationally-Hiding Commitments

The first essential component of Gennaro and Lindell's construction[10] and of our proposal are non-interactive computationally-hiding commitment schemes [11]. In this section we will describe such schemes, as well as a subtle issue that arises regarding these schemes.

Roughly speaking, they should fulfill the following requirement: A commitment c to a value m is of no help in gaining any information about m for any chosen-plaintext adversary of computationally-bounded resources(i. e., *Computationally-hiding* under CPA). In this paper, we simply call it computa- tionally-hiding.

Let $\mathcal{C}$ be a non-interactive computationally-hiding commitment scheme as above; such schemes are known to exist in the common reference string model[11]. Any public-key encryption scheme that is semantically secure under chosen plaintext attack can be used as a non-interactive computationally-hiding scheme in the common reference string model[11].

We denote by $C_\rho(m;r)$ a commitment to m using random-coins r and common reference string ρ. Such a commitment scheme is the basis for our hard problem. Let C_ρ denote the space of all strings that may be output by the commitment scheme $\mathcal{C}$ when the CRS is ρ, and let M denote the message space. We note that actually, C_ρ and M must be supersets of these spaces that are efficiently recognizable; the actual definition of the supersets depends on the specific commitment scheme used. Next, define the following sets:

- $X_\rho = C_\rho \times M$.
- $L_\rho = \{(c,m) | \exists\ r(\text{called a witness}) \colon c = C_\rho(m;r)\}$

Furthermore, define the partitioning of X_ρ to be by index m (i.e., consider $X_\rho(m) = C_\rho \times m$ and $L_\rho(m) = \{(c,m)|\exists\ r : c = C_\rho(m;r)\}$ (i.e., $L_\rho(m)$ is the language of all commitments to m using ρ). The distribution $D(L_\rho(m))$ is defined by choosing a random r and outputting $(C_\rho(m;r),m)$. In contrast, the distribution $D(X_\rho(m)\backslash L_\rho(m))$ is defined by choosing a random r and outputting $(C_\rho(0^{|m|};r),m)$. Clearly, by the hiding property of $\mathcal{C}$, it holds that for every m, random elements chosen from $D(L_\rho(m))$ are computationally indistinguishable from random elements chosen from $D(X_\rho(m)\backslash L_\rho(m))$. This therefore constitutes a hard partitioned subset membership problem. We refer to [10] for more information about it.

4.2 Smooth Projective Hashing

We briefly recall Gennaro and Lindell's proposal for constructing smooth projective hash families[10], given a suitable commitment scheme as above $\mathcal{C}$, and then introduce the new notion of TISPHash.

We are interested in a smooth projective hash function family defined with respect to (X_ρ, L_ρ). Loosely speaking a smooth projective hash function is a function with two keys. The first key maps the entire set $X_\rho(m)$ to some set G. The second key (called the projection key) is such that it can be used to correctly compute the mapping of L_ρ to G. However, it gives no information about the mapping of $X_\rho \backslash L_\rho$ to G. In fact, given the projection key, the distribution over the mapping of $X_\rho \backslash L_\rho$ to G is statistically close to uniform (or "smooth"). We now present the formal definition. A family of smooth projective hash functions $\mathcal{HASH} = (\alpha, SmthHash, ProjHash)$ associated with C_ρ consists of three algorithms. Let K be the key space, and k be a key chosen at random uniformly in K. We call k a hash key. Via $s \leftarrow \alpha(k)$, the key projection algorithm produces projected hash keys $s \in S$ for a hash key k, where S is the space of projected hash keys. Via $g \leftarrow SmthHash(k, c, m)$, the hashing algorithm computes the hash value $g \in G$ of (c, m) using the hash key k. Via $g \leftarrow ProjHash(s, c, m, r)$, the projected hashing algorithm computes the hash value $g \in G$ of (c, m) using the projected hash key s and a witness r (i.e. $c = C_\rho(m, r)$). Formally, the above system defines a smooth projective hash system if the following conditions are satisfied:

- For all $(c, m) \in L_\rho$ and $k \in K$, the hash values $SmthHash(k, c, m)$ and $ProjHash(\alpha(k), c, m, r)$ are the same(**Projective**);
- For all $(c, m) \in X_\rho \backslash L_\rho$ and $k \in K$, the hash value $g = SmthHash(k, c, m)$ is statistically close to uniform in G and independent of the values $(\alpha(k), c, m)$(**Smooth**).

The usual smooth projective hash function actually does not suffice for our application. Now, we define a new notion— target-independent smooth projective hashing (TISPHash). We say a smooth projective hash function family is target-independent if one more condition is satisfied:

- For $(c, m) \leftarrow D(X_\rho \backslash L_\rho)$ (where an appropriate witness r is not known), no ppt adversary can find a different pair of $(c^{'}, m^{'})$ such that the value $SmthHash(k, c, m)$ is dependent on the value $SmthHash(k, c^{'}, m^{'})$ with non-negligible probability, given the projection $\alpha(k)$(**Target-Independent**).

The so-called target-independent smooth projective hashing is just a specific smooth projective hash function. Actually, the example given in [10] based on the El-Gamal encryption scheme is smooth projective hashing but not target-independent.

4.3 Pseudo-Random Function (PRF)

The concept of a pseudo-random function (PRF) is defined in [12] by Goldreich, Goldwasser and Micali.

Let $l \in N$ be a security parameter. A pseudo-random function (PRF) family $\mathbf{F}$ associated with $\{\Sigma_l\}_{l \in N}$, $\{\mathcal{D}_l\}_{l \in N}$ and $\{\mathcal{R}_l\}_{l \in N}$ specifies a family of pseudo-random functions indexed by l, where each such function F^l_σ takes a random seed drawn by σ uniformly distributed over Σ_l and then maps an element of $\mathcal{D}_l$ to an element of $\mathcal{R}_l$.

Let $\mathcal{M}$ be a probabilistic polynomial-time machine with oracle access to $\mathcal{O}$(output an element in $\mathcal{R}$ with an element in $\mathcal{D}$). For all l, we define two experiments, $\text{Exp}^F_l(\mathcal{M})$ and $\text{Exp}^R_l(\mathcal{M})$, where $\mathcal{O}$ is answered with F^l_σ in the first experiment while it is answered with a truly random function $RF : \mathcal{D}_l \rightarrow \mathcal{R}_l$ in the other experiment. $\mathbf{F}$ is a pseudo-random function (PRF) family if there is no such probabilistic polynomialtime adversary $\mathcal{M}$ that can efficiently distinguish the above two experiments.

πPRFs, firstly introduced in [5], is a specific class of PRFs with pairwise-independent random sources. Let $(\sigma_0, \sigma_1, \cdots, \sigma_{t(l)})$ be random variables uniformly distributed over Σ_l and σ_0 be pairwisely independent from other variable, where $t(l)$ is a polynomial of l.

Let $\mathcal{M}$ be a probabilistic polynomial-time machine with oracle access to $(\mathcal{O}_0, \cdots, \mathcal{O}_{t(l)})$. For all l, we define two experiments, $\mathrm{Exp}_l^F(\mathcal{M})$ and $\mathrm{Exp}_l^R(\mathcal{M})$, where $\mathcal{O}_j$ is answered respectively with $F^l_{\sigma_j}$ for $j = 1, \cdots, t(l)$ in the both experiments and the only difference is that $\mathcal{O}_0$ is answered with $F^l_{\sigma_0}$ in the first experiment while it is answered with a truly random function $RF : \mathcal{D}_l \rightarrow \mathcal{R}_l$ in the second experiment. $\mathbf{F}$ is a πPRF family if there is no such probabilistic polynomialtime adversary $\mathcal{M}$ that can efficiently distinguish the above two experiments.

5 The Proposed Framework for AKE

The idea of our framework is inspired in that of the Gennaro-Lindell protocol[10], which itself is an abstraction of the password-based key exchange protocol of Katz, Ostrovsky, and Yung [13,14]. Our protocol is also an abstraction of the AKE protocol of T. Okamoto [5]. Our protocol builds on three basic tools: PRFs including πPRFs, non-interactive computationally-hiding commitment scheme $\mathcal{C}$ and target-independent smooth projective hash family $\mathcal{HASH}$. The last two can be implemented in the common reference string (CRS) model.

For a pictorial description of our protocol, please refer to Fig. 1. For the sake of readability, we do not explicitly refer to instances of users. The protocol description below omits many implementation details which are important for the proof of security to hold. Most important is for both parties to perform a "validity check" on the messages they receive. Let $e \overset{U}{\longleftarrow} E$ denote that e is uniformly selected from a set E. The formal description follows.

Description. Let $C_\rho(m, r)$ denote a commitment to the message m using random coin tosses r and the CRS ρ, where m can be simply set to be 1 in our protocol. Let $\mathcal{HASH} = (\alpha, SmthHash, ProjHash)$ be a family target-independent smooth projective hash functions based on C_ρ. In particular, we assume the image of the hash functions $SmthHash$ and $ProjHash$ to be contained in a finite abelian group G with g_1 as its generator. Recall that the key projection function α is defined as a function of K. Let $\hat{\mathbf{F}}$ and $\tilde{\mathbf{F}}$ be PRF families and $\mathbf{F}$ a family of πPRFs. The corresponding Σ, $\mathcal{D}$ and $\mathcal{R}$ can be inferred based on the description below and thus are omitted here. $\hat{F}$, $\tilde{F}$ and F are the member functions of them respectively with respect to the security parameter l. Our protocol has a total of two rounds of communication and works as follows.

Initialization. Party $\mathcal{A}$ chooses uniformly at random a value $a \in K$ as its static private key, computes its static public key A via α with a as input and

$$
\begin{array}{lcl}
\mathcal{A} & & \mathcal{B} \\
a \overset{U}{\longleftarrow} K & & b \overset{U}{\longleftarrow} K \\
A \leftarrow \alpha(a) & & B \leftarrow \alpha(b) \\
 & & \\
(x_1, x_2) \overset{U}{\longleftarrow} \{0,1\}^l \times \{0,1\}^l & & \\
(x, x_3) \leftarrow \hat{F}_{x_1}(1^l) + \tilde{F}_a(x_2) & & \\
X \leftarrow C_\rho(m, x) & & \\
X_3 \leftarrow g_1^{x_3} & \xrightarrow{(\hat{B}, \hat{A}, X, X_3)} & \\
 & & (y_1, y_2) \overset{U}{\longleftarrow} \{0,1\}^l \times \{0,1\}^l \\
 & & (y, y_3) \leftarrow \hat{F}_{y_1}(1^l) + \tilde{F}_b(y_2) \\
 & & Y \leftarrow C_\rho(m, y) \\
 & \xleftarrow{(\hat{A}, \hat{B}, X, X_3, Y, Y_3)} & Y_3 \leftarrow g_1^{y_3} \\
\sigma \leftarrow SmthHash(a, Y) \times & & \sigma \leftarrow SmthHash(b, X) \times \\
\quad ProjHash(B, X, x) \times Y_3^{x_3} & & \quad ProjHash(A, Y, y) \times X_3^{y_3} \\
sk \leftarrow F_\sigma(sid) & & sk \leftarrow F_\sigma(sid)
\end{array}
$$

Here, $sid \leftarrow (\hat{A}, \hat{B}, X, X_3, Y, Y_3)$, and (A, B) is confirmed indirectly through the certificates.

Fig. 1. The Proposed Framework for AKE

then register $A = \alpha(a)$ to CA. Similarly, Party $\mathcal{B}$ generates its static private key b and computes its static public key B and then register $B = \alpha(b)$ to CA.

Round 1. Party $\mathcal{A}$ selects an ephemeral private key $(x_1, x_2) \overset{U}{\longleftarrow} \{0,1\}^l \times \{0,1\}^l$ and computes $(x, x_3) \leftarrow \hat{F}_{x_1}(1^l) + \tilde{F}_a(x_2)$, $X \leftarrow C_\rho(m, x)$ and $X_3 \leftarrow g_1^{x_3}$. Then $\mathcal{A}$ erase (x, x_3) and the whole computation history of the ephemeral public key and Send $(\hat{B}, \hat{A}, X, X_3)$ to $\mathcal{B}$.

Round 2. Upon receiving the message from $\mathcal{A}$, Party $\mathcal{B}$ selects an ephemeral private key $(y_1, y_2) \overset{U}{\longleftarrow} \{0,1\}^l \times \{0,1\}^l$ and computes $(y, y_3) \leftarrow \hat{F}_{y_1}(1^l) + \tilde{F}_b(y_2)$, $X \leftarrow C_\rho(m, y)$ and $Y_3 \leftarrow g_1^{y_3}$. Then $\mathcal{B}$ erases (y, y_3) and the whole computation history of the ephemeral public key and Send $(\hat{A}, \hat{B}, X, X_3, Y, Y_3)$ to $\mathcal{A}$.

Finalization. To compute the session key, $\mathcal{A}(\mathcal{B})$ computes $\sigma \leftarrow SmthHash(a, Y) \times ProjHash(B, X, x) \times Y_3^{x_3}$ (resp. $\sigma \leftarrow SmthHash(b, X) \times ProjHash(A, Y, y) \times X_3^{y_3}$) and derives the session key $sk \leftarrow F_\sigma(sid)$, where $sid = (\hat{A}, \hat{B}, X, X_3, Y, Y_3)$.

Correctness. The correctness follows from the fact that, in an honest execution of the protocol, $Y_3^{x_3} = X_3^{y_3}$, $SmthHash(a, Y) = ProjHash(A, Y, y)$, $SmthHash(b, X) = ProjHash(B, X, x)$. It is easy to verify that both parties in the protocol will terminate by accepting and computing the same values for σ, sid, and $sk = F_\sigma(sid)$.

Rationale. One might wonder why the two parties have to construct X_3 and Y_3 and include them in the derivation function of σ. This is motivated by the following passive attack on the protocol deriving the value of σ simply from $SmthHash(a, Y) \times SmthHash(b, X)$, that allows to break the security of the protocol: The adversary eavesdrops an honest execution of the protocol between $\mathcal{A}$ and $\mathcal{B}$, where a pair of matchable sessions $(sid, \overline{sid})$ are established. When the execution is completed, the adversary queries StaticKeyReveal on the two participants, giving him the static private keys a and b, computes σ and thus easily learns the session key sk. Note, both the session sid and its matching session $\overline{sid}$ are still fresh since the adversary never queries EphemeralKeyReveal on either sid or $\overline{sid}$. Therefore, the adversary is allowed to Test on sid or $\overline{sid}$. Through this attack, the adversary could correctly guesses the bit b involved in the Test query with probability 1 and thus the protocol would not fulfill the security in Definition 2.

One might also wonder why the ephemeral private keys can not be immediately used to compute the ephemeral public key, e.g., $X \leftarrow C_\rho(m, x_1)$ and $X_3 \leftarrow g_1^{x_2}$. Instead they are derived from another pair of value (x, x_3) output by $\hat{F}$ and $\tilde{F}$. Note that the value of (x, x_3) can only be computed in a computation process with the knowledge of ephemeral and static private keys. Reconsider the above attack where the adversary queries EphemeralKeyReveal on the two sessions sid and $\overline{sid}$ but never queries StaticKeyReveal on any of the two participants instead. He can break the security of the protocol if the ephemeral private keys are immediately used to compute the ephemeral public key in the protocol.

Security. The intuition behind the security of our protocol is quite simple. We assume sid^* is the target session chosen by adversary. Let us remember the conditions of a fresh session (i.e., restrictions on sid^*). If the adversary mounts a passive attack on sid^* in which he eavesdrops on the honest execution among the participants, i.e., there exists a matching session, $\overline{sid^*}$ of target session sid^*, σ^* involved in sid^* would be uniform and independent for him due to the hardness of the DDH problem in G. If the adversary mounts an active attacks on sid^* in which he interacts with an honest party to establish session sid^*, i.e., there exists no matching session of target session sid^*, σ^* involved in sid^* would be pairwisely independent from any other σ involved in whichever session (not equivalent to session sid^*) the attacker can establish with an honest party due to the target-independence of the smooth projective hashing from computationally-hiding commitment in use. As long as the security notion of PRFs and πPRFs in use is met, an honest party is able to implicitly authenticate his partner and safely share a session key with him. As the following theorem shows, the protocol described in Fig. 1 is a secure authenticated key exchange protocol without random oracles.

Theorem 1. *Let $\hat{\mathbf{F}}$ and $\tilde{\mathbf{F}}$ be PRF families and $\mathbf{F}$ a family of πPRFs, let $\mathcal{C}$ be a non-interactive computationally-hiding commitment scheme and $\mathcal{HASH}$ be a family of target-independent smooth projective hash functions associated with $\mathcal{C}$. Then, the proposed AKE protocol is secure (in the sense of Definition 2) if the DDH assumption holds over G.*

Due to the limitation of the paper length, the complete proof is omitted here.

Note the commitment scheme in our framework is just a basis for the hard problem upon which we can build a family of target-independent smooth projective hash functions $\mathcal{HASH}$. Therefore, non-malleability is not needed in our construction. As a result, a public-key encryption scheme that is semantically secure under chosen plaintext attack can be used to instantiate the commitment scheme and m can be simply set to be 1 in our protocol. The intuition behind that is quite simple. Let us assume the adversary impersonates $\mathcal{A}$ to interact with $\mathcal{B}$. He sends a correct commitment X with a known x and thus knows $ProjHash(B, X, x)$ but he can never knows $SmthHash(a, Y)$, where a is $\mathcal{A}$'s static private key(the corresponding public key is A, which is included in $\mathcal{A}$'s certificate, and its validness can be checked in a usual way in PKI-based system) and Y is honestly generated by $\mathcal{B}$. So he can gain no information of the session key. This is true even when the adversary replays the message Y in another session since $\mathcal{HASH}$ in use is a family of target-independent smooth projective hash functions.

6 An Example for Efficient Constructions

To provide the reader with an idea of how efficient target-independent smooth projective hash functions are, we give an example based on the El-Gamal encryption scheme [10]. Note a non-interactive computationally-hiding commitment scheme can be constructed from the El-Gamal encryption encryption scheme. Our example is a little similar to that given in [5] based on the El-Gamal encryption scheme but the latter is not target-independent.

The El-Gamal scheme works as follows. Let G be a cyclic group of prime order p where p is large, and let g_1 be a generator g_1 of G. The key generation algorithm chooses a random $z \in Z_p$ with $z \neq 0$. The secret-key is then defined to be z and the public-key is defined to be $g_2 = g_1^z$. To encrypt $m \in G$, a random $x \in Z_p$ is chosen and the ciphertext is defined to be $X = (X_1, X_2) = (g_1^x, g_2^x \cdot m)$. Upon input X, decryption is carried out by computing $m = X_2/X_1^z$. It is well known that under the Decisional Diffie-Hellman Assumption over G, the El-Gamal scheme is semantically secure against chosen-plaintext attacks.

The smooth projective hashing for the El-Gamal encryption scheme is then defined as follows. The key space is defined by $K = Z_p^4$ (i.e., a key is a tuple $a = (a_1, a_2, a_3, a_4)$, with $a_i \in Z_p$). The key projection function α is defined by $\alpha(a) = A = (A_1, A_2) = (g_1^{a_1} g_2^{a_2}, g_1^{a_3} g_2^{a_4})$. The hash function SmthHash on input (a, X, m) outputs $\sigma = X_1^{a_1+ca_3}(X_2/m)^{a_2+ca_4}$, and the projective hash function ProjHash on input (A, X, m, x) simply outputs $\sigma = A_1^x A_2^{cx}$, where $c = H(X)$ and H is a TCR hash function. Projectiveness follows from the fact that $SmthHash(a, X, m) = ProjHash(A, X, m, x)$ holds for all $X \in L$. We now prove the smoothness property. Consider $X = (X_1, X_2) = (g_1^x, g_2^{x'} \cdot m) \neq L$. Then this implies that $x \neq x'$. Then (A, σ) are expressed by the following three equations:

$$\log_{g_1} A_1 = a_1 + za_2 \bmod p \tag{1}$$
$$\log_{g_1} A_2 = a_3 + za_4 \bmod p \tag{2}$$
$$\log_{g_1} \sigma = x(a_1 + ca_3) + zx'(a_2 + ca_4) \bmod p \tag{3}$$

Since $\log_{g_1} \sigma$ is linearly independent from $\log_{g_1} A_1$ and $\log_{g_1} A_2$ when $x \neq x'$, i.e., for every choice of A and σ, there exists a tuple $a = (a_1, a_2, a_3, a_4)$ that fulfills the above equations, σ is uniformly distributed over G, given A. We conclude that the projective hash function is smooth.

It now remains to prove the target-independence property. Consider any such $Y = (Y_1, Y_2) = (g_1^y, g_2^{y'} \cdot m) \neq X$. Then $\delta = SmthHash(a, Y, m) = Y_1^{a_1+da_3}(\frac{Y_2}{m})^{a_2+da_4}$ is expressed by the following equation:

$$\log_{g_1}\delta = y(a_1 + ca_3) + zy'(a_2 + da_4) \bmod p \qquad (4)$$

where $d = H(Y)$. If $Y \in L$, i.e. $y = y'$, the value of σ is (information theoretically) independent from δ since, given δ and A, σ is still uniformly distributed in G if a_2, a_4 is chosen uniformly at random in Z_p^2. Then, we consider the case that $Y \notin L$, i.e. $y \neq y'$. We hereafter assume $c \neq d \bmod p$. We can safely do so because H is a TCR hash function. We can show the value of σ is (information theoretically) independent from δ since $z^2(x - x')(y - y')(c - d) \neq 0 \bmod p$ and thus the following matrix:

$$\begin{bmatrix} 1 & z & 0 & 0 \\ 0 & 0 & 1 & z \\ x & zx' & cx & zcx' \\ y & zy' & dy & zdy' \end{bmatrix}$$

is regular. Actually, for every choice of A, σ and δ, there exists a tuple $a = (a_1, a_2, a_3, a_4)$ that fulfills the above equations (1)-(4). Therefore, σ is uniformly distributed over G, given A and δ. We conclude that the projective hash function is target-independent.

We remark that a protocol that is almost identical to that of [5] can be derived when our protocol is instantiated with the El-Gamal encryption scheme based on the DDH assumption, combined with the above construction of a target-independent smooth projective hash function (which is thus implicit in the work of [5]). As it is the case with Gennaro and Lindell's construction[10], our protocol enjoys efficient instantiations based on the decisional Diffie-Hellman, quadratic residuosity, and N-residuosity assumptions (see [10]).

7 Conclusion

We have proposed a framework for (PKI-based) authenticated key exchange protocols without random oracles and proved it to be secure in the (currently) strongest security definition. An example of efficient instantiations of our protocol also is given and and the resulting protocol is almost identical to that of T. Okamoto, which is quite efficient [5]. Our protocol is actually an abstraction of the key exchange protocol of T. Okamoto. The abstracted protocol not only inherits all its attractive features(including in efficiency) but also enjoys efficient instantiations from any secure encryption scheme that admits an efficient construction of TISPHash and allows a simple and intuitive understanding of its security. In some sense, our construction generalizes the design of T. Okamoto.

References

1. Krawczyk, H.: HMQV: A high-performance secure diffie-hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005), http://eprint.iacr.org/2005/176
2. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS, Cryptology ePrint Archive, Report 2006/073 (2006), http://eprint.iacr.org/2007/123
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: 1st Conf. on Computer and Communications Security, pp. 62–73. ACM, New York (1993)
4. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. In: 30th STOC, pp. 209–218 (1998)
5. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474–484. Springer, Heidelberg (2007)
6. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography 28, 119–134 (2003)
7. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange, Cryptology ePrint Archive, Report 2006/073 (2006), http://eprint.iacr.org/2006/073
8. 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, p. 453. Springer, Heidelberg (2001), http://eprint.iacr.org/2001/040
9. 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)
10. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003), http://eprint.iacr.org/2003/032.ps.gz
11. Di Crescenzo, G., Katz, J., Ostrovsky, R., Smith, A.: Efficient and non-interactive non-malleable commitment. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 40–59. Springer, Heidelberg (2001)
12. Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. Journal of the ACM 33(4), 792–807 (1986)
13. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
14. Katz, J., Ostrovsky, R., Yung, M.: Forward secrecy in password-only key exchange protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 29–44. Springer, Heidelberg (2003)

Secret Handshake: Strong Anonymity Definition and Construction

Yutaka Kawai, Kazuki Yoneyama*, and Kazuo Ohta

The University of Electro-Communications 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan
{kawai,yoneyama,ota}@ice.uec.ac.jp

Abstract. Secret handshake allows two members in the same group to authenticate each other secretly. In previous works of secret handshake schemes, two types of anonymities against the group authority (GA) of a group G are discussed: 1)*Even* GA cannot identify members, namely nobody can identify them (No-Traceability), 2)*Only* GA can identify members (Traceability). In this paper, first the necessity of tracing of the identification is shown. Second, we classify abilities of GA into the ability of identifying players and that of issuing the certificate to members. We introduce two anonymities *Co-Traceability* and *Strong Detector Resistance*. When a more strict anonymity is required ever for GA, the case 2) is unfavorable for members. Then, we introduce *Co-Traceability* where even if $\mathcal{A}$ has GA's ability of identifying members or issuing the certificate, $\mathcal{A}$ cannot trace members identification. However, if a scheme satisfies Co-Traceability, GA may be able to judge whether handshake players belong to the own group. Then, we introduce *Strong Detector Resistance* where even if an adversary $\mathcal{A}$ has GA's ability of identifying members, $\mathcal{A}$ cannot make judgments whether a handshaking player belongs to G. Additionally, we propose a secret handshake scheme which satisfies previous security requirements and our proposed anonymity requirements by using group signature scheme with message recovery.

Keywords: Secret handshake, anonymity, traceability, privacy.

1 Introduction

1.1 Background

A secret handshake scheme (SHS), introduced by Balfanz et al. [3], allows two members of the same group to authenticate each other secretly, in the sense that each party reveals his affiliation information to the other only if the other party belongs to the same group. For example scenario: a CIA agent Alice wants to authenticate to Bob, but only if Bob is also a CIA agent. In addition, if Bob is not a CIA agent, Alice does not want to reveal her affiliation information, whether Alice is a CIA agent or not, for Bob.

* Supported by JSPS Research Fellowships for Young Scientists.

Balfanz, et al. [3] constructed a 2-party SHS by adapting the key agreement protocol of Sakai, et al. [10]. Its security rests on the hardness of the Bilinear Diffie Hellman (BDH) problem. Subsequently, Castelluccia, et al. [6] developed a more efficient 2-party SHS through the use of so-called CA-oblivious encryption. It is secure under Computational Diffie Hellman (CDH) assumption.

Both previous works satisfy the basic security properties of secret handshake system; *correctness, impersonator resistance,* and *detector resistance.* Recently, the *unlinkability* is added to the basic security requirement. Unlinkability means that two occurrences of handshaking by the same party cannot be linked with each other by anyone. In [3,6], however, the member sends one's ID information in a handshake protocol. Thus, the construction of [3,6] does not satisfy *unlinkability* unless members use one-time certificate (i.e. member change IDs whenever members execute handshake protocol).

Xu and Yung [11] constructed a secret handshake scheme which achieves weaker unlinkability "k-anonymity" with reusable certificate, instead of one-time certificates. Members can reuse their certificate because they always authenticate as someone among k users.

The work of [1] presented the first construction of SHS with unlinkability using reusable certificate in the standard model. The scheme in [1] allows each participant to specify the role and group of the other party and thus add flexibility to the authentication rules. Moreover, they achieve attribute-based secret handshakes using fuzzy identity-based-encryption.

1.2 Anonymity against Group Authority

Motivation: Let us consider the case in which whistle-blowing the system of company is troublesome. In this case the GA is a manager of company and the members are employees of the company and the authorized persons of the company (e.g. lawyer). When the employee uses this system, he executes a secret handshake with the lawyer in order to pass on the whistle-blowing. The employee wants to tell to only his company's lawyer about exposure. The lawyer should only know the fact that the employee belongs to his company. In above scenario, a secret handshake scheme is very useful.

However, if previous secret handshake schemes are used as above whistle-blowing system, there is a problem. In previous secret handshake schemes, GA can identify handshake players (player's names, IDs, etc). In this scenario, the employees, who want to exposure, want to execute a secret handshake to remain anonymous. If an employee is not guaranteed anonymity, he will not blow the whistle in the fear of the dismissal etc. However, for example in [6], a handshake player must send own ID to a handshake partner. IDs are registered to GA, when a member joins to the group. So, GA can identify handshake players by referring ID list and watching the transcript of handshake.

From the above reason, the previous schemes are not applicable to this case. So in this paper, we will define new security requirements about anonymities against GA. We introduce two new anonymities *Co-Traceability* and *Strong Detector*

Resistance. Strong Detector Resistance means that even if an adversary $\mathcal{A}$, who does not belong to G, has GA's ability of identifying members, $\mathcal{A}$ cannot make judgments whether a handshaking player belongs to G. *Co-Traceability* means that GA alone cannot reveal members identification.

Co-Traceability can never be satisfied against GA with all of abilities, since if U belongs to G, U might be revealed identification executing a handshake protocol with a dummy member D whom GA creates using an ability to issue a user. Then, GA can identify U by using the information of D and the ability of identifying members. Also, Strong Detector Resistance can never be satisfied against GA with all of the abilities. The reason is that if U belongs to G, U outputs accept certainly executing a handshake protocol with a dummy member D whom GA creates using an ability to issue a user.

Therefore, in order to discuss the anonymity against GA, we split the role of GA into the two; issue authority and trace authority. The issue authority (IA) issues the certificate to users. The trace authority (TA) has an ability to identify a member. Co-Traceability is defined against an adversary who has an ability of either TA or IA. Strong Detector Resistance is defined against an adversary who has an ability of TA.

Co-Traceability: If a secret handshake scheme adopt whistle-blowing, handshake players do not want to reveal own identification against even GA. However, tracing of handshake players is useful for the handshake player in case the evidence of handshaking is required.

So, we introduce new security requirement Co-Traceability. Intuitively, Co-Traceability means that TA alone cannot identify handshake players, but TA can identify players by cooperating with another. In our proposed secret handshake scheme, another is a handshake player. If Alice and Bob execute a secret handshake, TA alone cannot identify Alice and Bob, but if TA cooperates with Alice, TA could identify Bob.

At the same time, in this paper, we will define this algorithm "$\mathsf{SHS.Co\text{-}Trace}$". This new algorithm is useful in the situation which a handshake player wants to know the other handshake partner when the output of the protocol is *accept*. In previous secret handshake schemes, identifications of handshake players always reveal against everyone. By using $\mathsf{SHS.Co\text{-}Trace}$, a handshake player can execute a handshake protocol hiding own identification and can know the identification of the other handshake player by revealing own identification to TA.

Strong Detector Resistance: In previous works, in Secret Handshake system, information about the group of the player must not be leaked, if players does not belong to the same group. There are the cases that a player wants to hide the information of his own group from TA as well as members and users. If TA can know whether a handshake player belongs to his own group G or not, GA might examine the frequency that a member of group G executes a handshake protocol. This consideration leads to the new security property, *Strong Detector Resistance.*

Table 1. Levels of trust in authorities for each of security requirements

Previous Security	Issuer Authority	Trace Authority
Impersonator Resistance	Uncorrupted	Uncorrupted
Detector Resistance	Uncorrupted	Uncorrupted
Unlinkability	Uncorrupted	Uncorrupted
Proposed Security	**Issuer Authority**	**Trace Authority**
Co-Traceability (against TA)	Uncorrupted	Corrupted
Co-Traceability (against IA)	Corrupted	Uncorrupted
Strong Detector Resistance	Uncorrupted	Corrupted

In this paper, we discuss the anonymity and groups which members belong to in the case TA or IA is corrupted. Here, "TA (IA) is corrupted" means that an adversary $\mathcal{A}$ can get a secret key of TA (IA). The levels of trust for each authority for each requirement are summarized in Table 1.

In order to realize *Strong Detector Resistance* and *Co-Traceability*, we introduce a new algorithm SHS.Co-Trace. SHS.Co-Trace, given a group public key, a group authority secret key, and "an internal information of a player", outputs ID of the other player. Concretely, we realize SHS.Co-Trace and *Co-Traceability* with a secret key to execute SHS.Co-Trace separately by GA and a handshake player.

2 Definition of Secret Handshake

2.1 Entity

In SHS, there exist three entities in the group G as follows.

User: the entity which does not belong to the group. $A \notin G$ means that the user A does not belong to the group G.

Member: the entity which is made belong to the group, by the Group Authority. $A \in G$ means that the member A belongs to the group G.

GA (Group Authority): the manager of a group. GA is responsible for adding users into the group he manages. GA maintains a list $\mathcal{L}$, which includes certificates and registration data of all members.

2.2 The Model of Secret Handshake

A secret handshake scheme SHS consists of the following five algorithms:

SHS.Setup: generates the public parameters *param* which is common to all groups.

SHS.CreateGroup: generates a key pair *gpk* (group public key) and *gsk* (private key for GA), using *param*. SHS.CreateGroup is run by GA.

SHS.AddMember: is executed between a player U and a GA of some group. Inputs of SHS.AddMember are gsk, $param$, and gpk, and outputs are a membership certificate ($cert_U$) and a secret key (sk_U)

SHS.Handshake: is the authentication protocol executed between U and V, based on the public input $param$. The group public keys (gpk_U, gpk_V) and certificates (($cert_U, sk_U$), ($cert_V, sk_V$)) of U and V are input to handshake protocol. The result of the algorithm is either reject or accept. $U \overset{Handshake}{\longleftrightarrow} V$ means the above situation.

SHS.Trace: , given gpk, gsk and a transcript $\mathcal{T}$ of the handshaking of U and V, outputs the member U (or V).

2.3 Security Properties of SHS Scheme

A secret handshake protocol must satisfy the following *basic* security properties: **Correctness, Impersonator Resistance, Detector Resistance, Unlinkability** [6].

Correctness: If honest members U,V of the same group run handshake protocol, then both players always output "*accept*".

Impersonator Resistance (IR): IR demands that the adversary $\mathcal{A}$, who does not belong to a group G, is not able to authenticate an honest who belongs to G.

Detector Resistance (DR): DR demands that the adversary $\mathcal{A}$, who does not belong to a group G, is not able to distinguish whether some honest is a member of some group G.

Unlinkability (Unlink): Unlink demands that the adversary $\mathcal{A}$, who does not belong to a group G, is not able to decried whether two executions of the handshake protocol were performed by the same party or not, even if both of them were successful.

3 Anonymity against Group Authority

In previous works of secret handshake schemes, anonymities against GA have not been discussed. In this section we will define two new anonymity requirements against GA, *Co-Traceability* and *Strong Detector Resistance* of secret handshake schemes.

3.1 Issue Authority and Trace Authority

In this subsection, let us classify the GA from the view point of abilities. GA has the two abilities. First ability is to issue the certificate to users. Second ability is to trace handshake players. We call the authority with the former ability *Issue authority* (IA). and the authority with the latter ability *Trace authority* (TA). IA has a public and secret key (ipk, isk) and TA has a public and secret key (tpk, tsk)[1]

[1] In previous definition, $gpk = (ipk, tpk)$, $gsk = (isk, tsk)$.

3.2 Co-traceability

In previous works of secret handshake schemes, two types of anonymities against the GA are discussed from the view point of revealing the identification of handshake players: 1)*Even* GA cannot identify players, namely nobody can identify them, 2)*Only* GA can identify players.

Tracing of handshake players is useful for the handshake player in case the evidence of handshaking is required. However when the anonymity is preferred as the case of prosecution from inside, a more strict anonymity is required ever for GA.

In the way of SHS.Trace, members except TA cannot join SHS.Trace process. From the view point of power of TA, this situation could be troublesome for members. Then, we introduce a new security requirement *Co-Traceability.* Intuitively, *Co-traceability* means that TA alone cannot identify handshake players.

Co-Traceability can never be satisfied against GA with all abilities, since if U belongs to G, U might be revealed identification executing a handshake protocol with a dummy member D who GA creates using an ability to issue a user. So, two different types of adversarys, $\mathcal{A}_T$ and $\mathcal{A}_I$, should be concerned. The difference between $\mathcal{A}_T$ and $\mathcal{A}_I$ is the input. The input of $\mathcal{A}_T$ is *param*, *gpk* and *tsk*. The input of $\mathcal{A}_I$ is *param*, *gpk* and *isk*. The formal definition of Co-traceability is as follow:

Definition 1 (Co-Traceability). Co-Traceability (Co-Trace) means that an adversary $\mathcal{A}$ who has all GA's secret key (isk, tsk) cannot identify the player which execute SHS.Handshake even though $\mathcal{A}$ is not a member of G. Formally, we say that SHS guarantees Co-Trace if the following function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k)$is negligible. Let a member which executes according to the protocol be honest. We consider a PPT adversary $\mathcal{A}$ that can access to the following oracles $\mathcal{O}$.

Formally, we say that SHS guarantees Co-Trace if the following function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k)$ is negligible for any PPT adversary $\mathcal{A} = (\mathcal{A}_T, \mathcal{A}_I)$.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k) = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k) = 1]$$

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{Co-Trace}}(k)$
    $param \leftarrow \mathsf{SHS.Setup}(k)$
    $(ipk, tpk, isk, tsk) \leftarrow \mathsf{SHS.CreateGroup}(param)$;
    $(cert_U, sk_U) \leftarrow \mathsf{SHS.AddMember}(param, ipk, tpk, isk, U)$
    $\mathsf{honest}(param, gpk, sk_U, cert_U) \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}$
    $U' \leftarrow \mathcal{A}^{\mathcal{O}}$
    If $U = U'$, outputs 1.        Otherwise outputs 0.
```

3.3 Strong Detector Resistance

If a secret handshake scheme satisfies Co-Traceability, TA alone cannot identify handshake players. However, GA may be able to identify whether handshake players belong to the own group. Then, GA can know the frequency that members of own group execute handshake. In order to achieve high anonymity against GA, even the frequency of handshake should be hidden for GA.

We then introduce the concept of *Strong Detector Resistance* in order to cope with this requirement. Intuitively, Strong Detector Resistance means that even if an adversary $\mathcal{A} \notin G$ has all the ability of revealing the identity of members, $\mathcal{A}$ cannot make judgment whether a handshaking player belongs to G or not.

Strong Detector Resistance never satisfies against IA, since if U belongs to G, U outputs *accept* executing a handshake protocol with a dummy member who GA creates using an ability to issue a user. The formal definition of Strong Detector Resistance is as follow:

Definition 2 (Strong Detector Resistance). Strong Detector Resistance (SDR) demand that an adversary $\mathcal{A}$ who have "the all ability of revealing the identity of members" (e.g. trace key, member-list $\mathcal{L}$) cannot distinguish whether some honest player, who is a member of group G, even though $\mathcal{A}$ is not a member of G. SDR is easier security goal than SDR for an adversary $\mathcal{A}$.

Formally, we say that SHS guarantees SDR if the following function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SDR}}(k)$ is negligible for any polynomially-bounded adversary $\mathcal{A}$.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SDR}}(k) = \left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{SDR}}(1,k)=1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{SDR}}(0,k)=1]\right|$$

SDR has a close relationship to the SHS.Trace algorithm. If GA can reveal an identify ID of a member [6,3], an adversary can break SDR by executing SHS.Trace. On the other hand, if SHS does not have algorithm revealing an identify [1], SDR is equivalent to DR.

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{SDR}}(b,k)$
    $param \leftarrow \mathsf{SHS.Setup}(k)$
    $(ipk, tpk, isk, tsk) \leftarrow \mathsf{SHS.CreateGroup}(param)$;
    $(cert_U, sk_U) \leftarrow \mathsf{SHS.AddMember}(param, ipk, tpk, isk, U)$
    $\mathbf{player}_b := \mathsf{honest}(param, gpk, sk_U, cert_U)$;
    $\mathbf{player}_{1-b} := \mathsf{SIM}(param)$
      $\mathbf{player}_0 \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk, tsk)$
      $\mathbf{player}_1 \overset{Handshake}{\longleftrightarrow} \mathcal{A}^{\mathcal{O}}(param, gpk, tsk)$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}}(param, gpk, tsk)$        Return $b'$
```

4 Proposed Scheme

4.1 Group Signature with Message Recovery (GSMR) [12]

Our proposed SHS, including SHS.Co-Trace algorithm and satisfied Co-Traceability and Strong Detector Resistance, is based on the SHS using GSMR (Group Signature with Message Recovery) [12].

A group signature, first introduce by Chaum and van Heyst [7] and followed by [2,8], allows a member which belong to a group to sign messages on behalf of the group without a member reveal own identity. In group signature system, as the same secret handshake systems, there exists a manager (authority) of group. A manager, in the case of a dispute, can reveal an identity of any group signature make valid group signatures A standard group signature scheme involves consists of five algorithms $\mathsf{GS.KeyGen}$, $\mathsf{GS.Join}$, $\mathsf{GS.Sign}$, $\mathsf{GS.Verify}$, $\mathsf{GS.Open}$. $\mathsf{GS.KeyGen}$,is a key generation algorithm, is given security parameter and outputs a group public key gpk and a group manager secret key $gmsk$. $\mathsf{GS.Join}$, given $gmsk$ and a member secret key sk, outputs membership certificate. $\mathsf{GS.Sign}$, given gpk, a member secret key sk and a message m, outputs a group signature σ. $\mathsf{GS.Verify}$, given gpk, m, and σ, outputs accept if σ is valid for m with respect to gpk. $\mathsf{GS.Open}$, given m, σ, and $gmsk$, outputs signer's identity. The security requirements of a group signature scheme are traceability, anonymity, and non-frameability[4].

Intuitively, [12] achieve Unlink by using group signature (more precisely, anonymity of group signatures) and realized $\mathsf{SHS.Trace}$ by $\mathsf{SHS.Open}$. In [12], a group signature with message recovery (GSMR) is constructed from the standard group signature[2]. In order to apply GSMR to SHS, GSMR can be forged a signature corresponding to an arbitrary message using a valid signature and gpk. If GSMR does not this property, anyone can be convinced of handshake players' groups by carrying out $\mathsf{GS.Verify}$.

4.2 Construction of Proposed Scheme

We show the construction of proposed SHS scheme. Our GSMR scheme is based on [8]. We assume that $\mathbb{G}_1 = \langle G_1 \rangle, \mathbb{G}_2 = \langle G_2 \rangle$ of prime order p that have a bilinear map e and $e(G_1, G_2)$ generates $\mathbb{G}_T$. Here, $(\mathbb{G}_1, \mathbb{G}_2)$ is bilinear groups.

$\mathsf{SHS.Setup}$: Given a security parameter k, generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e)$ and chooses hash functions $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$ and $\mathcal{G} : \{0,1\}^* \to \mathbb{Z}_p$.

$\mathsf{SHS.CreateGroup}$: First, IA chooses $w \leftarrow_R \mathbb{Z}_p$ and $H \leftarrow_R \mathbb{G}_1$ and generates $W = wG_2$. IA outputs $isk = w, ipk = (H, W)$. Next, TA chooses $(t, s) \leftarrow_R (\mathbb{Z}_p)^2$ and generates T, S such that $H = sS = tT$. TA outputs $tsk = (t, s), tpk = (T, S)$.

$\mathsf{SHS.AddMember}$: First, user U, who wants to join to the group, chooses $(x', z) \leftarrow_R (\mathbb{Z}_p)^2$ and generates $H' = x'H + zG_1$. U sends H' to IA. Next, IA chooses $(x'', z') \leftarrow_R (\mathbb{Z}_p)^2$ and sends (x'', z') to U. U generates $x_U = x'x'' + z'$ and $H_U = x_U H$. U sends H_U to IA. U proves in zeroknowlege to IA the knowledge of x_U and $x''z$ satisfying $H_U = x_U H$ and $x''H' + z'H - H_U = x''zG_1$. Finally, IA chooses $y_U \leftarrow_R \mathbb{Z}_p$ and generates $A_U = \frac{1}{w+y_U}(G_1 - H_U)$. IA sends (A_U, y_U) to U. IA adds (U, A_U, y_U) to the group member-list $\mathcal{L}$.
$\mathsf{GSMR}(param, gpk, sk, cert, m) \to \sigma$:

1. U chooses s', t', S', T', H' s.t. $s'S' = t'T' = H'$ $((s', t') \in (\mathbb{Z}_p)^2$, $(S', T', H') \in (\mathbb{G}_1)^3)$. If U will want to execute $\mathsf{SHS.Co\text{-}Trace}$, U has to memorize their parameters.

2. U chooses $(\alpha, \beta, \alpha', \beta') \leftarrow_R \mathbb{Z}_p^*$ and generates $R_1 = \alpha T, R_2 = \beta S, R_3 = \alpha' T', R_4 = \beta' S'$ and $R_5 = (\alpha + \beta)H + (\alpha' + \beta')H' + A_U$
3. U chooses $(r_x, r_y, r_\alpha, r_\beta, r_{y\alpha}, r_{y\beta}, r_{\alpha'}, r_{\beta'}, r_{y\alpha'}, r_{y\beta'}) \leftarrow_R (\mathbb{Z}_p^*)^{10}$ and generate $R'_1 = r_\alpha T, R'_2 = r_\beta S, R'_3 = r_{\alpha'} T', R'_4 = r_{\beta'} S'$,
$R'_5 = e(R_5, G_2)^{r_y} e(H, W)^{-(r_\alpha + r_\beta)} e(H', W)^{-(r_{\alpha'} + r_{\beta'})} e(H, G_2)^{-(r_{y\alpha} + r_{y\beta}) + r_x}$
$e(H', G_2)^{-(r_{y\alpha'} + r_{y\beta'})}, R'_6 = r_y R_1 - r_{y\alpha} T, R'_7 = r_y R_2 - r_{y\beta} S, R'_8 = r_y R_3 - r_{y\alpha'} T'$ and $R'_9 = r_y R_4 - r_{y\beta'} S'$
4. U generates $c' = \mathcal{H}(param, gpk, R_1, R_2, R_3, R_4, R_5, R'_1, R'_2, R'_3, R'_4, R'_5, R'_6, R'_7, R'_8, R'_9)$ and $c = c' \oplus m$.
5. U generates $s_x := r_x + c x_U, s_y := r_y + c y_U, s_\alpha := r_\alpha + c\alpha, s_\beta := r_\beta + c\beta s_{y\alpha} := r_{y\alpha} + c(y\alpha), s_{y\beta} := r_{y\beta} + c(y\beta), s_{\alpha'} := r_{\alpha'} + c\alpha', s_{\beta'} := r_{\beta'} + c\beta', s_{y\alpha'} := r_{y\alpha'} + c(y\alpha')$ and $s_{y\beta'} := r_{y\beta'} + c(y\beta')$
6. U outputs $\sigma = (R_1, R_2, R_3, R_4, R_5, s_x, s_y, s_\alpha, s_\beta, s_{y\alpha}, s_{y\beta}, s_{\alpha'}, s_{\beta'}, s_{y\alpha'}, s_{y\beta'}, c)$

MR:$(param, gpk, \sigma) \rightarrow m$

1. V is given $param, gpk$, and σ.
2. V generates $R'_1 = s_\alpha T - cR_1, R'_2 = s_\beta S - cR_2, R'_3 = s_\alpha T' - cR_3, R'_4 = s_\beta S' - cR_4, R'_5 = e(R_5, G_2)^{s_y} e(H, W)^{-(s_\alpha + s_\beta)} e(H, G_2)^{-(s_{y\alpha} + s_{y\beta}) + s_x}$
$e(H', W)^{-(s_{\alpha'} + s_{\beta'})} e(H, G_2)^{-(s_{y\alpha'} + s_{y\beta'})} \left(\frac{e(R_5, G_2)}{e(G_1, G_2)}\right)^c, R'_6 = s_y R_1 - s_{y\alpha} T,$
$R'_7 = s_y R_2 - s_{y\beta} S, R'_8 = s_y R_3 - s_{y\alpha'} T'$, and $R'_9 = s_y R_4 - s_{y\beta'} S'$
3. V generates $c' = \mathcal{H}(param, gpk, R_1, R_2, R_3, R_4, R_5, R'_1, R'_2, R'_3, R'_4, R'_5, R'_6, R'_7, R'_8, R'_9)$ and $m = c \oplus c'$. V outputs m.

SHS.Handshake: Suppose the member U with certificate $cert_U = (A_U, y_U)$ and secret key $sk_U = x_U$, and the member V with certificate $cert_V = (A_V, y_V)$ and secret key $sk_V = x_V$, engage in handshake protocol.

1. U and V generates $(s'_U, t'_U, S'_U, T'_U, H'_U)$, $(s'_V, t'_V, S'_V, T'_V, H'_V)$
s.t. $s'_U S_U = t'_U T = H'_U, s'_V S_V = t'_V T = H'_V$
$(s'_U, t'_U, s'_V, t'_V \in (\mathbb{Z}_p)^4, S'_U, T'_U, H'_U, S'_V, T'_V, H'_V \in \mathbb{G}_1)$.
2. U chooses $r_U \leftarrow_R \mathbb{Z}_p^*$ and generates $m_U := r_U G_1$ and
$\sigma_U \leftarrow \mathsf{GSMR}(param, gpk_U, sk_U, cert_U, m_U)$. U send σ_U to VD
3. V chooses $r_V \leftarrow_R \mathbb{Z}_p^*$ and generates $m_V := r_V G_1$,
$\sigma_V \leftarrow \mathsf{GSMR}(param, gpk_V, sk_V cert_V, m_V)$ and $m'_U \leftarrow \mathrm{MR}(param, gpk_V, \sigma_U)$. V send σ_V to UD
4. U generates $m'_V \leftarrow \mathrm{MR}(param, gpk_U, \sigma_V)$ and $resp_U := \mathcal{G}(r_U m'_V, m_U)$ and send $resp_U$ to V.
5. V generates $resp_V := \mathcal{G}(r_V m'_U, m_V)$. If $resp_U = \mathcal{G}(r_V m'_U, m'_U)$, V outputs *accept* and send $resp_V$ to U. Otherwise V outputs *reject*.
6. If $resp_V = \mathcal{G}(r_U m'_V, m'_V)$, U outputs *accept*. Otherwise U outputs *reject*.

SHS.Co-Trace: Suppose the member U with $cert_U = (A_U, y_U)$, secret key $sk_U = x_U$, and the parameters $(s'_U, t'_U, S'_U, T'_U, H'_U)$ used in SHS.Handshake.

1. TA is given $cert_U = (A_U, y_U), sk_U = x_U$ from U.
If $e(A, W)e(H, G_2)^x e(A, G_2)^y = e(G_1, G_2)$, TA executes the following. Otherwise TA outputs $\perp$.

Table 2. Comparison among SHS

Scheme	[3]	[6]	[1]	Proposed Scheme
Underlying Assumption	CBDH	CDH	SXDH and BDH	DL and DLDH
Number of rounds	3	4	2	4
Communication complexity (bits)	640	8512	684	5804
IR security	Yes	Yes	Yes	Yes
DR security	Yes	Yes	Yes	Yes
Unlink security	No	No	Yes	Yes
SDR security	No	No	Yes	Yes
Co-Trace security	No	No	No	Yes
SHS.Trace algorithm	Yes	Yes	No	No
SHS.Co-Trace algorithm	No	No	No	Yes

2. TA is given $\sigma_V(= R_{V1}, R_{V2}, R_{V3}, R_{V4}, R_{V5}, s_{Vx}, s_{Vy}, s_{V\alpha}, s_{V\beta}, s_{Vy\alpha}, s_{Vy\beta}, s_{V\alpha'}, s_{V\beta'}, s_{Vy\alpha'}, s_{Vy\beta'}, c_V)$ (in the transcript of SHS.Handshake) and computes $R_{V5} - (tR_{V1}sR_{V2} + t'R_{V3}s'R_{V4}) = A_V$ and outputs V such that $(V, A_V, y_V) \in \mathcal{L}$.

Among SHS, we show the comparisons with respect to important factors in Table 2. [3], [1] and proposed scheme use bilinear maps. We assume that $\mathbb{G}_1 \neq \mathbb{G}_2$ such that the representation of G_1 can be a 172-bit prime when $|p| = 171$. Also, we assume that [6] are instantiated on a 160-bits prime order subgroup of a prime finite field of 1024 bits. In [3], [6] and proposed scheme, a member has own ID. On the other hand, in [1], a member does not have ID, so [1] can not include both SHS.Trace and SHS.Co-Trace.

Our proposed scheme is secure under discrete logarithm assumption and Decisional Linear Diffie-Hellman (DLDH) assumption[5]. Proofs of the following theorems are given in [9].

Theorem 1. *The proposed scheme has Impersonator Resistance property, if the discrete logarithm problem is hard to solve.*

Theorem 2. *The proposed scheme has Strong Detector Resistance, Unlinkability and Co-Traceability property, if the Decisional Linear Diffie-Hellman (DLDH) problem is hard to solve.*

References

1. Ateniese, G., Blanton, M., Kirsch, J.: Secret handshakes with dynamic and fuzzy matching. In: Network and Distributed System Security Symposium (2007)
2. Ateniese, G., Camenisch, J.L., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)

3. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.C.: Secret handshakes from pairing-based key agreements. In: IEEE Symposium on Security and Privacy, 2003, pp. 180–196. IEEE Computer Society, Los Alamitos (2003)
4. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
6. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret handshakes from CA-oblivious encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
7. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
8. Furukawa, J., Imai, H.: An efficient group signature scheme from bilinear maps. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 455–467. Springer, Heidelberg (2005)
9. Kawai, Y., Yoneyama, K., Ohta, K.: Secret handshake: Strong anonymity definition and construction. In: IACR ePrint Archive, 2009 (2009)
10. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: The Symposium on Cryptography and Information Security, SCIS 2000 (2000)
11. Xu, S., Yung, M.: k-anonymous secret handshakes with reusable credentials. In: Proceedings of the 11th ACM conference on Computer and communications security, pp. 158–167. ACM, New York (2004)
12. Yamashita, T., Tada, M.: How to construct secret handshake schemes. In: Computer Security Symposium 2007, CSS 2007, pp. 321–326 (2006) (in japanese)

An Extended Authentication and Key Agreement Protocol of UMTS

Farshid Farhat, Somayeh Salimi, and Ahmad Salahi

Iran Telecommunication Research Center, North Karegar, Tehran, Iran
{farhat,ssalimi,salahi}@itrc.ac.ir

Abstract. Identification, authentication and key agreement protocol of UMTS networks have some weaknesses to provide DoS-attack resistance, mutual freshness, and efficient bandwidth consumption. In this article we consider UMTS AKA and some other proposed schemes. Then we explain the known weaknesses in the previous frameworks suggested for UMTS AKA protocol. After that we propose a new UMTS AKA protocol (called EAKAP) for UMTS mobile network that combines identification stage and AKA stage of UMTS AKA protocol as well as eliminating disadvantages of related works and bringing some new features to improve the UMTS AKA mechanism such as reducing the interactive rounds of the UMTS AKA protocol.

Keywords: Identification, Authentication and Key Agreement, UMTS, Mobile Network, Security Protocol.

1 Introduction

The wireless communications advances cause the ease of access to wireless services for individuals. Because of the air interface between the user and the network, the physical security of users' media is in serious danger with respect to the wired infrastructure. Physical layer security like spread spectrum methods for commercial usages is so expensive, hence wireless providers try to secure higher layers to obtain privacy and confidentiality features for their subscribers.

The most commonly used wireless communications are cellular communications. In the first-generation (1G) analog systems, security services were not addressed. Proper authentication of subscribers is an important feature for operators to charge them correctly. So in the second-generation (2G) digital cellular systems, the GSM communications, authentication and also confidentiality were taken into account and security measures were designed for these goals in 2G.

Some of the GSM security weaknesses are active attack by fake BTS, no secure communication between BTS and BSC and also between BSC and MSC, no data integrity check and weak stream cipher algorithm (A5/1,2) for confidentiality. These weaknesses were concentrated in the security design phase of UMTS, a major standard for the third-generation (3G). So, an enhanced authentication and key agreement protocol was considered for UMTS and integrity was added as well as using strong algorithms.

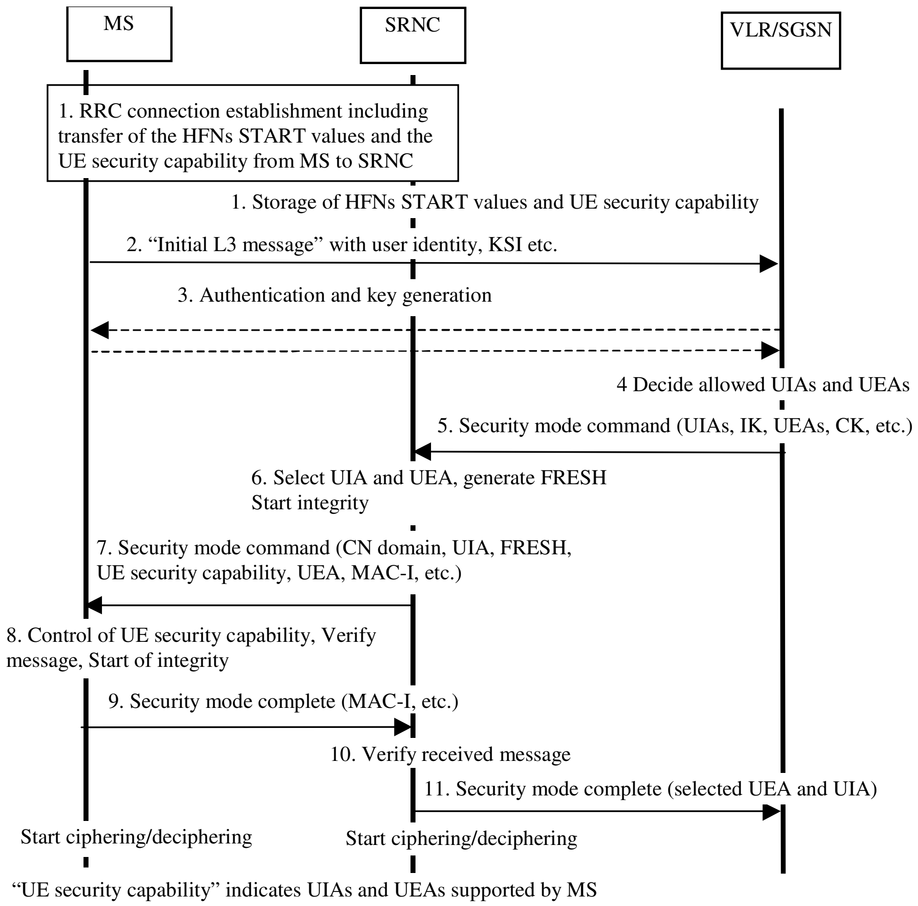

MS
SRNC
VLR/SGSN
1. RRC connection establishment including transfer of the HFNs START values and the UE security capability from MS to SRNC
1. Storage of HFNs START values and UE security capability
2. "Initial L3 message" with user identity, KSI etc.
3. Authentication and key generation
4 Decide allowed UIAs and UEAs
5. Security mode command (UIAs, IK, UEAs, CK, etc.)
6. Select UIA and UEA, generate FRESH Start integrity
7. Security mode command (CN domain, UIA, FRESH, UE security capability, UEA, MAC-I, etc.)
8. Control of UE security capability, Verify message, Start of integrity
9. Security mode complete (MAC-I, etc.)
10. Verify received message
11. Security mode complete (selected UEA and UIA)
Start ciphering/deciphering
Start ciphering/deciphering
"UE security capability" indicates UIAs and UEAs supported by MS

Fig. 1. The UMTS Identification and AKA procedure [1]

Between features mentioned above, the authentication and key agreement protocol has vital importance which the most prominent security features are based on. In this protocol, other than authentication, user and network agree on the cipher and integrity keys CK and IK respectively. If existence of any vulnerability in this protocol, other than these keys, the subscriber secret key K may be compromised and so, we focus on this protocol for the purpose of improving it.

This paper is organized as follows; in section II we explain the UMTS authentication and key agreement (AKA) procedure. Section III outlines related challenges to improve the security and performance of the UMTS AKA protocol. In section IV the new proposed protocol is described that covers the previous known weaknesses. In section V the EAKAP is analyzed and evaluated from the security point of view. We conclude the paper in section VI. The descriptions of abbreviations are given in appendix.

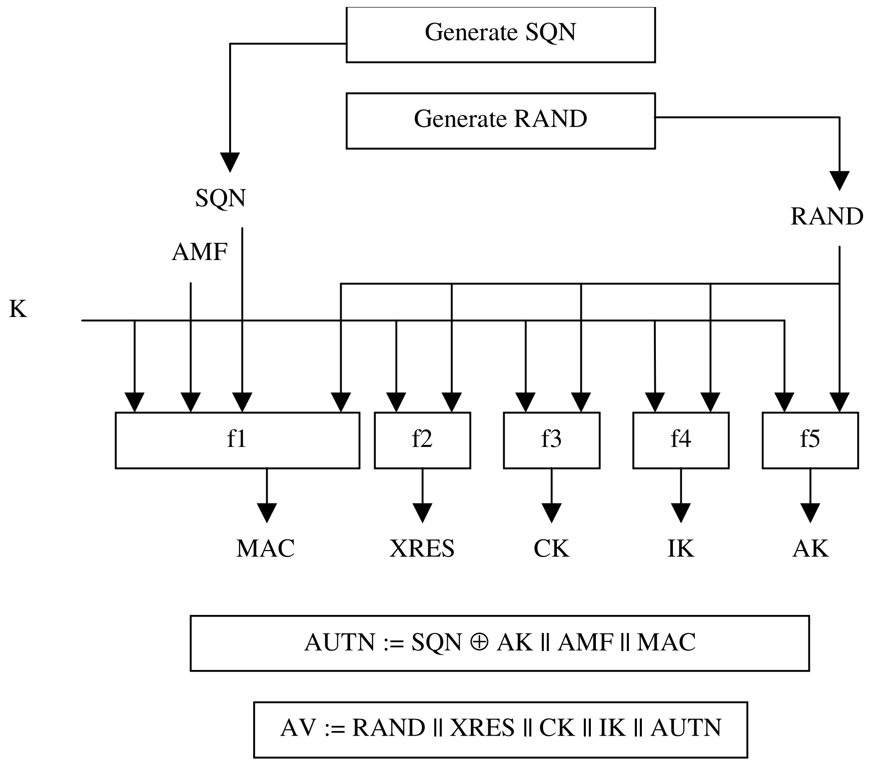

Generate SQN
Generate RAND
SQN
AMF
RAND
K
f1
f2
f3
f4
f5
MAC
XRES
CK
IK
AK
AUTN := SQN ⊕ AK || AMF || MAC
AV := RAND || XRES || CK || IK || AUTN

Fig. 2. Generation of Authentication Vectors [1]

2 UMTS AKA Description

The purpose of UMTS AKA is to authenticate the user and network to each other and also establish a new pair of cipher and integrity keys between the VLR/SGSN and the USIM [1]. MS, VLR/SGSN, and HLR/AuC (HE) are involved in UMTS AKA protocol. At identification stage, MS sends its identity to VLR/SGSN via SRNC.

The secret key (*K*) and cryptographic Algorithms including f_1, f_1^*, f_2, f_3, f_4, f_5, and f_5^*, shared between MS and HE, are used for UMTS AKA process. Furthermore HE and MS track the value of a counter (SQN_{HE}) and (SQN_{MS}) respectively. These sequence numbers are used for the purpose of freshness checking of the received messages. As shown in figure1, UMTS Identification, distribution of authentication data, and AKA procedure are performed as follows.

Identification

1- The MS sends the initial L3 message including its TIMSI and the KSI to VLR/SGSN via SRNC. By this message, the MS requests for services like Location Update, CM Service and Routing Area Update.

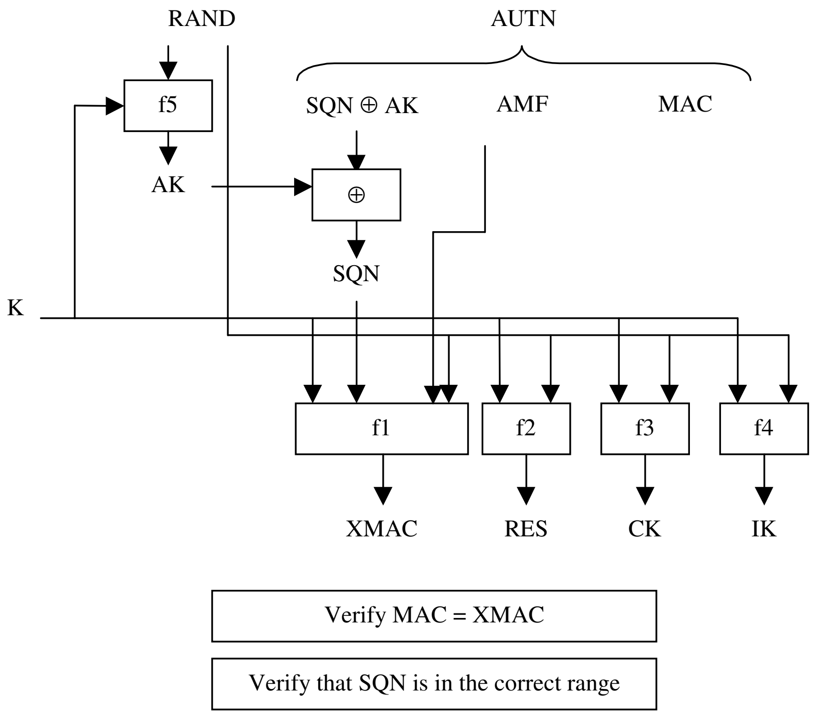


Fig. 3. User Authentication Function in the MS [1]

Distribution of Authentication Data

2- VLR/SGSN identifies MS by its TMSI and then sends the authentication data request including IMSI and requesting node type (PS or CS) to the HE.

3- Upon the receipt of the authentication data request, the HE sends an authentication data response back to the VLR/SGSN which contains an ordered array of n authentication vectors *AV (1...n)*. Each *AV* includes parameters *RAND*, *XRES*, *CK*, *IK*, and *AUTN*. Generation of *AV* is shown in figure2.

Authentication and Key Management

4- The VLR/SGSN chooses the next unused *AV* from the ordered array of *AV*s in the VLR/SGSN database on the basis of first-in/first-out. Then the VLR/SGSN sends to the MS the random challenge (*RAND*) and an authentication token (*AUTN*) from the chosen *AV*.

5- When the MS receives *RAND*||*AUTN*, it proceeds as illustrated in figure3. The MS first computes the anonymity key $AK=f5_K(RAND)$ and retrieves the sequence number. Then the MS computes $XMAC=f1_K(SQN||RAND||AMF)$ and compares it with MAC included in *AUTN*. If they are the same, the MS verifies if the *SQN* is in the correct range. Then, the MS calculates $RES=f2_K(RAND)$ and sends it to the VLR/SGSN. The VLR/SGSN compares the received *RES* with *XRES*. If they match, then the authentication process of the MS is successfully completed.

3 Related Works

In this section we consider the security analysis of UMTS AKA Protocols. Many protocols have been suggested for UMTS AKA improvement, but we choose some protocols that have novelty in their design and use symmetric algorithms. Nevertheless they have some security or performance weaknesses in their structure that we try to explain.

The UMTS X-AKA protocol [7] applies a temporary key mechanism with timestamp instead of the sequence number. The function *f5* is used for generating temporary keys. The UMTS X-AKA protocol consists of two procedures. First, the user registers on HN and then HN distributes temporary key (TK) and authentication information to SN. Second, the authentication and key agreement procedure is executed between SN and MS. The SN uses TK and authentication information to carry out the mutual authentication between SN and MS and then an agreed key and a cipher key are provided. The UMTS X-AKA protocol uses timestamp to manage freshness of the messages. The timestamp usage needs a robust time synchronization infrastructure. Time synchronization structure of the network has no security feature, so the usage of an independent structure with no security to refresh the exchanged messages is hazardous. Also the HN could not recognize the shared session keys between MS and SN, because SN generates the pseudo-random number needed to construct the session keys.

In [8] an AKA protocol with robust user privacy protection has been proposed. In this scheme, temporary key mechanism to authenticate MS and prevent the location privacy attack is used. In addition, it has lower overhead on VLR. Since MS can easily compute the temporary key through the shared secret key, VLR can be authenticated by MS successfully. In this protocol, the VLR initiates the authentication process by sending a nonce to the MS without any MAC, so DoS attack is probable to be imposed on the MS. Also the protocol has seven steps without identification and security mode set-up stages.

J. Al-Saraireh and S. Yousef proposed an AKA protocol [9] in which, the MS generates the AVs sending to the network. They provided an efficient bandwidth consuming framework with minimal ways for the AKA procedure, but the proposed protocol doesn't support mutual authentication i.e. only the network authenticates the MS. The protocol has 3 steps. The man in the middle attack scenario on interworking of UMTS and GSM [10] could be applied on this protocol [9], because the MS doesn't recognize the validity of the network. Furthermore, the DoS attack on the MS is possible, because the MS could only verify the network until a MAC is received from the network. With some modification in this protocol, it will be mutual. If the VLR/SGSN sends the RES received from the HE to the MS, the mutual authentication would be satisfied by checking the XRES and RES in the MS side.

The security of the wireless network access has been enhanced by Harn and Hsin [11] that uses timestamp and hash chain to provide non-repudiation and freshness. As mentioned earlier, using the timestamp needs independent secure infrastructure. Also the hash chain construction consumes much computation load at the end user side. Furthermore, the number of the protocol ways is six and the protocol doesn't contain the identification and security mode set-up stages.

An extension of UMTS AKA protocol has been proposed by J. Al-Saraireh and S. Yousef in [12] that provides mutual freshness of the MS and the HE but it doesn't use sequence number mechanism and instead, both the MS and HE generates random numbers. So verifying the freshness of the messages could be done by searching in a large database that contains all of the previous random numbers generated by the parties. Applying such a huge database is more expensive and complex than using sequence number method. Besides, the power of DoS attack diminishes when the parties can check the integrity and freshness of the messages faster.

M. Zhang and Y. Fang enhanced the security of the 3GPP AKA by a protocol called AP-AKA. They have projected a special scenario of the redirection attack that UMTS AKA is weak towards it and AP-AKA is robust against it. But the MS traffic redirection by a virtual relay to the neighbor VLR could charge the MS more than usual because the location of the MS has been virtually changed. The first step of the AP-AKA has not integrity protection, so it could be forged. Also the attack [10] can be executed on AP-AKA while interworking with GSM because the VLR initiates the AKA procedure without integrity check. Furthermore, the identification and security mode set-up stages are not considered to provide a better performance for the six-way AP-AKA protocol.

4 UMTS Extended AKA Protocol

In this section, we propose a new authentication protocol of UMTS mobile networks. The stages of the proposed protocol are illustrated in figure4 and figure5. UMTS AKA has some weaknesses to provide simple DoS-attack resistance. Also previous proposed schemes (described in section III) do not provide strong mutual freshness. Furthermore, the steps of the pre-proposed protocols could be reduced to save bandwidth consumption.

The new proposed UMTS AKA protocol, named EAKAP, combines identification stage and AKA stage with security mode set-up of described UMTS protocol. EAKAP is done by a 5-way handshake protocol between the MS, the VLR/SGSN, and the HE. Most of the previous schemes were done by a 5-way handshake in the phase of AKA without security mode set-up, so we could enhance bandwidth efficiency. The EAKAP applies the secret key (K) and the cryptographic algorithms that are used in UMTS AKA protocol shared between the MS and the VLR/SGSN. But the usage of the *f2* algorithm in EAKAP is not necessary because EAKAP does not generate the *RES* or the *XRES*.

Both the MS and the HE have sequence number and random number generator to provide complete freshness of their messages. On the basis of the *f5* structure [2], we could improve the confidentiality of the *SQN* by changing its encryption method. To protect VLR/SGSN against DoS attack, we assume some security capabilities at VLR/SGSN side. These security features contain the shared cryptographic algorithms (*fc* and *fi*) between MS and VLR/SGSN. The *fc* algorithm is used for ciphering and the *fi* algorithm is used to generate integrity check. The EAKAP procedure works as follows (shown in figure4).

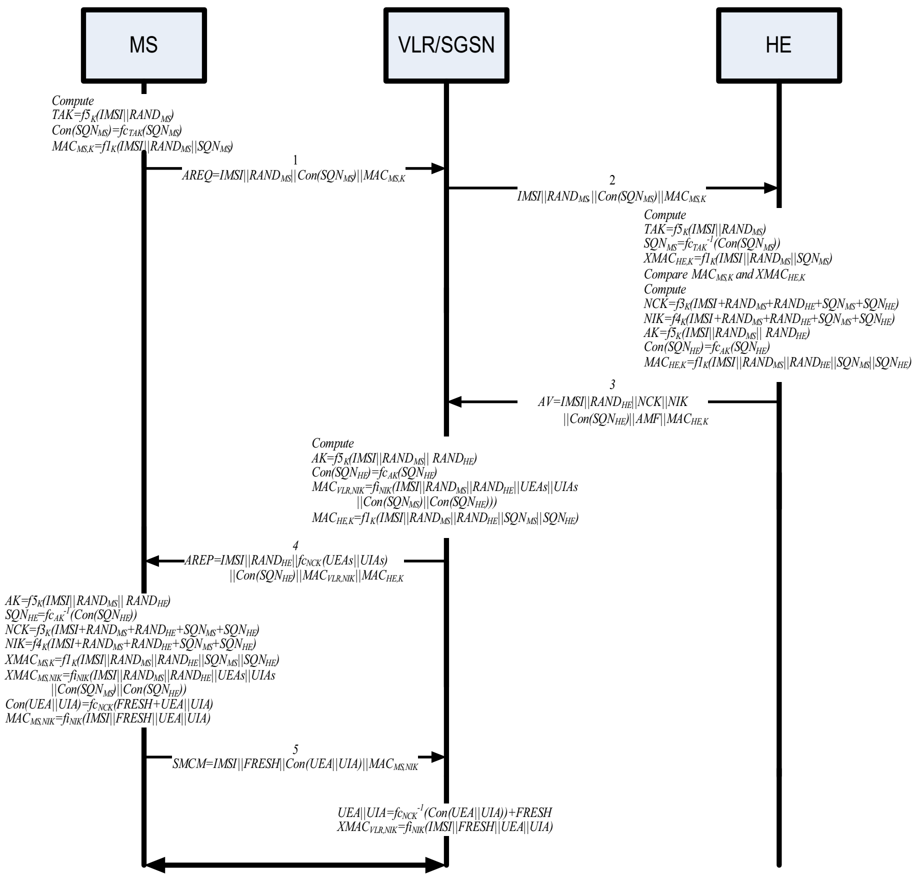

MS
VLR/SGSN
HE
Compute
$TAK=f5_K(IMSI||RAND_{MS})$
$Con(SQN_{MS})=fc_{TAK}(SQN_{MS})$
$MAC_{MS,K}=f1_K(IMSI||RAND_{MS}||SQN_{MS})$
1
$AREQ=IMSI||RAND_{MS}||Con(SQN_{MS})||MAC_{MS,K}$
2
$IMSI||RAND_{MS}||Con(SQN_{MS})||MAC_{MS,K}$
Compute
$TAK=f5_K(IMSI||RAND_{MS})$
$SQN_{MS}=fc_{TAK}^{-1}(Con(SQN_{MS}))$
$XMAC_{HE,K}=f1_K(IMSI||RAND_{MS}||SQN_{MS})$
Compare $MAC_{MS,K}$ and $XMAC_{HE,K}$
Compute
$NCK=f3_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$NIK=f4_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$AK=f5_K(IMSI||RAND_{MS}||RAND_{HE})$
$Con(SQN_{HE})=fc_{AK}(SQN_{HE})$
$MAC_{HE,K}=f1_K(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE})$
3
$AV=IMSI||RAND_{HE}||NCK||NIK$
$||Con(SQN_{HE})||AMF||MAC_{HE,K}$
Compute
$AK=f5_K(IMSI||RAND_{MS}||RAND_{HE})$
$Con(SQN_{HE})=fc_{AK}(SQN_{HE})$
$MAC_{VLR,NIK}=fi_{NIK}(IMSI||RAND_{MS}||RAND_{HE}||UEAs||UIAs$
$||Con(SQN_{MS})||Con(SQN_{HE})))$
$MAC_{HE,K}=f1_K(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE})$
4
$AREP=IMSI||RAND_{HE}||fc_{NCK}(UEAs||UIAs)$
$||Con(SQN_{HE})||MAC_{VLR,NIK}||MAC_{HE,K}$
$AK=f5_K(IMSI||RAND_{MS}||RAND_{HE})$
$SQN_{HE}=fc_{AK}^{-1}(Con(SQN_{HE}))$
$NCK=f3_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$NIK=f4_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$XMAC_{MS,K}=f1_K(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE})$
$XMAC_{MS,NIK}=fi_{NIK}(IMSI||RAND_{MS}||RAND_{HE}||UEAs||UIAs$
$||Con(SQN_{MS})||Con(SQN_{HE}))$
$Con(UEA||UIA)=fc_{NCK}(FRESH+UEA||UIA)$
$MAC_{MS,NIK}=fi_{NIK}(IMSI||FRESH||UEA||UIA)$
5
$SMCM=IMSI||FRESH||Con(UEA||UIA)||MAC_{MS,NIK}$
$UEA||UIA=fc_{NCK}^{-1}(Con(UEA||UIA))+FRESH$
$XMAC_{VLR,NIK}=fi_{NIK}(IMSI||FRESH||UEA||UIA)$

Fig. 4. EAKAP Structure when MS meets a new VLR/SGSN

Authentication Request

1- If no TMSI was set before, the MS generates the *AREQ* message as follows:

$$AREQ=IMSI||RANDMS||Con(SQN_{MS})||MAC_{MS,K} \tag{1}$$

Where

$$TAK=f5_K(IMSI||RAND_{MS}) \tag{2}$$

$$Con(SQN_{MS})=fc_{TAK}(SQN_{MS}) \tag{3}$$

$$MAC_{MS,K}=f1_K(IMSI||RAND_{MS}||SQN_{MS}) \tag{4}$$

If the VLR/SGSN has allocated a TMSI encrypted by previous *CK* (*OCK*) paired with previous *IK* (*OIK*) to the MS, the MS would generate the *AREQ* message as follows:

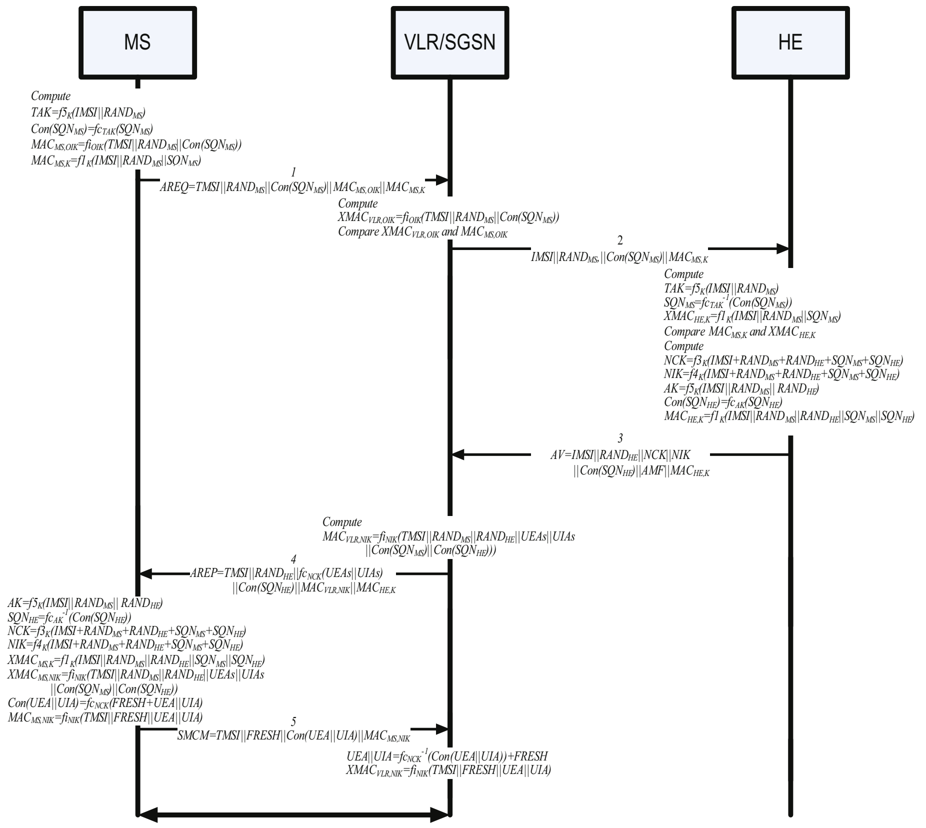

MS
VLR/SGSN
HE
Compute
$TAK=f5_K(IMSI||RAND_{MS})$
$Con(SQN_{MS})=fc_{TAK}(SQN_{MS})$
$MAC_{MS,OIK}=fi_{OIK}(TMSI||RAND_{MS}||Con(SQN_{MS}))$
$MAC_{MS,K}=f1_K(IMSI||RAND_{MS}||SQN_{MS})$
1
$AREQ=TMSI||RAND_{MS}||Con(SQN_{MS})||MAC_{MS,OIK}||MAC_{MS,K}$
Compute
$XMAC_{VLR,OIK}=fi_{OIK}(TMSI||RAND_{MS}||Con(SQN_{MS}))$
Compare $XMAC_{VLR,OIK}$ and $MAC_{MS,OIK}$
2
$IMSI||RAND_{MS},||Con(SQN_{MS})||MAC_{MS,K}$
Compute
$TAK=f5_K(IMSI||RAND_{MS})$
$SQN_{MS}=fc_{TAK}^{-1}(Con(SQN_{MS}))$
$XMAC_{HE,K}=f1_K(IMSI||RAND_{MS}||SQN_{MS})$
Compare $MAC_{MS,K}$ and $XMAC_{HE,K}$
Compute
$NCK=f3_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$NIK=f4_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$AK=f5_K(IMSI||RAND_{MS}||RAND_{HE})$
$Con(SQN_{HE})=fc_{AK}(SQN_{HE})$
$MAC_{HE,K}=f1_K(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE})$
3
$AV=IMSI||RAND_{HE}||NCK||NIK$
$||Con(SQN_{HE})||AMF||MAC_{HE,K}$
Compute
$MAC_{VLR,NIK}=fi_{NIK}(TMSI||RAND_{MS}||RAND_{HE}||UEAs||UIAs$
$||Con(SQN_{MS})||Con(SQN_{HE})))$
4
$AREP=TMSI||RAND_{HE}||fc_{NCK}(UEAs||UIAs)$
$||Con(SQN_{HE})||MAC_{VLR,NIK}||MAC_{HE,K}$
$AK=f5_K(IMSI||RAND_{MS}||RAND_{HE})$
$SQN_{HE}=fc_{AK}^{-1}(Con(SQN_{HE}))$
$NCK=f3_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$NIK=f4_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE})$
$XMAC_{MS,K}=f1_K(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE})$
$XMAC_{MS,NIK}=fi_{NIK}(TMSI||RAND_{MS}||RAND_{HE}||UEAs||UIAs$
$||Con(SQN_{MS})||Con(SQN_{HE}))$
$Con(UEA||UIA)=fc_{NCK}(FRESH+UEA||UIA)$
$MAC_{MS,NIK}=fi_{NIK}(TMSI||FRESH||UEA||UIA)$
5
$SMCM=TMSI||FRESH||Con(UEA||UIA)||MAC_{MS,NIK}$
$UEA||UIA=fc_{NCK}^{-1}(Con(UEA||UIA))+FRESH$
$XMAC_{VLR,NIK}=fi_{NIK}(TMSI||FRESH||UEA||UIA)$

Fig. 5. EAKAP Structure when MS meets an old VLR/SGSN

$$AREQ=TMSI||RAND_{MS}||Con(SQN_{MS})||MAC_{MS,OIK}||MAC_{MS,K} \tag{5}$$

Where

$$TAK=f5_K(IMSI||RAND_{MS}) \tag{6}$$

$$Con(SQN_{MS})=fc_{TAK}(SQN_{MS}) \tag{7}$$

$$MAC_{MS,OIK}=fi_{OIK}(TMSI||RAND_{MS}||Con(SQN_{MS})) \tag{8}$$

$$MAC_{MS,K}=f1_K(IMSI||RAND_{MS}||SQN_{MS}) \tag{9}$$

Then the MS sends the calculated *AREQ* to the VLR/SGSN.

Distribution of Authentication Data

2- The VLR/SGSN identifies the MS by its sent IMSI via HE or TMSI via old VLR/SGSN. If TMSI related to current VLR/SGSN is sent, the VLR/SGSN computes the $XMAC_{VLR,OIK}$ (as given below) and compares this with $XMAC_{MS,OIK}$ which is included in *AREQ*.

$$XMAC_{VLR,OIK}=fi_{OIK}(TMSI||RAND_{MS}||Con(SQN_{MS})) \quad (10)$$

If they are different, VLR/SGSN does not distribute the authentication data request for HE and AKA process fails. Otherwise, VLR/SGSN sends the authentication data request included IMSI, *$RAND_{MS}$*, *$Con(SQN_{MS})$* and *$MAC_{MS,K}$* to HE.

3- When the HE receives the authentication data request from the VLR/SGSN, it first retrieves sequence number of the MS as follows.

$$TAK=f5_{K}(IMSI||RAND_{MS}) \quad (11)$$

$$SQN_{MS}=fc_{TAK}^{-1}(Con(SQNMS)) \quad (12)$$

Then the HE computes the *$XMAC_{HE,K}$* (as given below) and compares it with *$MAC_{MS,K}$* which is included in the VLR/SGSN's authentication data request.

$$XMAC_{HE,K}=f1_{K}(IMSI||RAND_{MS}||SQN_{MS}) \quad (13)$$

If the *$MAC_{MS,K}$* and the *$XMAC_{HE,K}$* are different, the HE detects the MS as a fraud user and does not generate any *AV*. If they are the same, the HE verifies the *SQN* is in the correct range. If the *SQN_{MS}* is considered to be in the correct range, the HE sends an authentication data response back to the VLR/SGSN that contains an authentication vector (*AV*). The generated *AV* included IMSI, *$RAND_{HE}$*, new *CK* (*NCK*), new *IK* (*NIK*), *$Con(SQN_{HE})$*, *AMF*, and *$MAC_{HE,K}$* as shown in figure5.

$$AV=IMSI||RAND_{HE}||NCK||NIK||Con(SQN_{HE})||AMF||MAC_{HE,K} \quad (14)$$

Where

$$NCK=f3_{K}(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE}) \quad (15)$$

$$NIK=f4_{K}(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE}) \quad (16)$$

$$AK=f5_{K}(IMSI||RAND_{MS}||\ RAND_{HE}) \quad (17)$$

$$Con(SQN_{HE})=fc_{AK}(SQN_{HE}) \quad (18)$$

$$MAC_{HE,K}=f1_{K}(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE}) \quad (19)$$

Authentication Reply

4- Upon the receipt of the AV from the HE, the VLR/SGSN computes the *AREP* for the MS. The VLR/SGSN determines which UIAs and UEAs that are allowed to be used in order of preference. If no TMSI was set before, the VLR/SGSN generates the *AREP* message as follows:

$$\begin{gathered} AREP=IMSI||RAND_{HE}||fc_{NCK}(UEAs||UIAs) \\ ||Con(SQN_{HE})||MAC_{VLR,NIK}||MAC_{HE,K} \end{gathered} \quad (20)$$

Where

$$AK=f5_K(IMSI||RAND_{MS}|| RAND_{HE}) \quad (21)$$

$$Con(SQN_{HE})=fc_{AK}(SQN_{HE}) \quad (22)$$

$$MAC_{VLR,NIK}=fi_{NIK}(IMSI||RAND_{MS}||RAND_{HE} ||UEAs||UIAs||Con(SQN_{MS})||Con(SQN_{HE}))) \quad (23)$$

$$MAC_{HE,K}=f1_K(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE}) \quad (24)$$

If the VLR/SGSN allocates a TMSI encrypted by previous *CK* (*OCK*) paired with old *IK* (*OIK*) to the MS, it would generate the *AREP* message as follows:

$$AREP=TIMSI||RAND_{HE}||fc_{NCK}(UEAs||UIAs) ||Con(SQN_{HE})||MAC_{VLR,NIK}||MAC_{HE,K} \quad (25)$$

Where

$$MAC_{VLR,NIK}=fi_{NIK}(TIMSI||RAND_{MS}||RAND_{HE} ||UEAs||UIAs||Con(SQN_{MS})||Con(SQN_{HE})) \quad (26)$$

5- Upon the receipt of the *AREP* message, the MS proceeds as illustrated in figure4 and figure5. If the MS has a TMSI, allocated by the VLR/SGSN, it first retrieves the HE's sequence number as follows.

$$AK=f5_K(IMSI||RAND_{MS}|| RAND_{HE}) \quad (27)$$

$$SQN_{HE}=fc_{AK}^{-1}(Con(SQN_{HE})) \quad (28)$$

If the *SQN* is considered to be in the correct range, the MS calculates the new pair key, the new cipher key (*NCK*) and the new integrity key (*NIK*), as follows:

$$NCK=f3_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE}) \quad (29)$$

$$NIK=f4_K(IMSI+RAND_{MS}+RAND_{HE}+SQN_{MS}+SQN_{HE}) \quad (30)$$

Then the MS calculates the $XMAC_{MS,K}$ (as given below) and checks the correspondence of the $MAC_{HE,K}$ and the $XMAC_{MS,K}$.

$$XMAC_{MS,K}=f1_K(IMSI||RAND_{MS}||RAND_{HE}||SQN_{MS}||SQN_{HE}) \quad (31)$$

If they are the same, the MS decrypts the $fc_{NCK}(UEAs||UIAs)$ and gets the cryptography capabilities of the VLR/SGSN in order of preference. Then the MS computes $XMAC_{MS,NIK}$ (as given below) with regard to the allocated TMSI.

$$XMAC_{MS,NIK}=fi_{NIK}(TMSI||RAND_{MS}||RAND_{HE} ||UEAs||UIAs||Con(SQN_{MS})||Con(SQN_{HE})) \quad (32)$$

If no TMSI was allocated, the $XMAC_{MS,NIK}$ would be as follows:

$$XMAC_{MS,NIK}=fi_{NIK}(IMSI||RAND_{MS}||RAND_{HE} \\ ||UEAs||UIAs||Con(SQN_{MS})||Con(SQN_{HE})) \tag{33}$$

The MS compares $XMAC_{MS,NIK}$ with $MAC_{VLR,NIK}$ included in the received *AREP* message. If they are different, the MS does not continue authentication process and terminate the connection. Otherwise, the MS generates the security mode complete message (SMCM) as follows:

$$SMCM=TIMSI||FRESH||Con(UEA||UIA)||MAC_{MS,NIK} \tag{34}$$

Where

$$Con(UEA||UIA)=fc_{NCK}(FRESH+UEA||UIA) \tag{35}$$

$$MAC_{MS,NIK}=fi_{NIK}(TMSI||FRESH||UEA||UIA) \tag{36}$$

The MS sends the SMCM to the VLR/SGSN and the VLR/SGSN decrypts the *Con(UEA||UIA)* by the new shared cipher key (*NCK*) to get preferred UEA and UIA.
$UEA||UIA=fc_{NCK}^{-1}(Con(UEA||UIA))+FRESH$

Then the VLR/SGSN verifies the integrity of the SMCM by generating the $XMAC_{VLR,NIK}$ with regard to the chosen UEA and UIA by the MS and comparing it with the $MAC_{MS,NIK}$.

$$XMAC_{VLR,NIK}=fi_{NIK}(TMSI||FRESH||UEA||UIA) \tag{37}$$

Consequently the IAKA (Identification and AKA) process with security mode setup is complete i.e. the parties authenticate each other mutually and the preferred algorithms for ciphering and integrity checking are chosen by them.

5 Security and Performance Features of the Proposed Protocol

In this section, we analyze the security features of the new proposed protocol explained in previous section and then we evaluate the performance of the suggested protocol relatively. The EAKAP provides the main security services issued in literature like authentication, confidentiality and integrity [3]. Furthermore, the EAKAP is more robust than the UMTS AKA and the previously suggested protocols against the DoS attack so with using the EAKAP, availability service would be provided properly.

Performance Evaluation towards Previous Works

In the EAKAP, three sections of the UMTS protocol including identification, AKA and security mode set-up are combined to set-up connection. In EAKAP, security mode setup is performed after key establishment so the attacker has no information about the shared algorithms for ciphering and integrity check. The EAKAP is completed after five-way interactions. Previous works do not consider the identification and security mode set-up steps. Furthermore, if we consider the whole protocol, the number of interactions has been reduced largely. So the load of the network signaling is reduced and performance of the IAKA procedure grows efficiently.

Confidentiality, Integrity, and Authentication

As described in previous section, the only shared secret between the MS and the HE is a private key (K). Sometimes the old session pair keys (*OCK*, *OIK*) are shared between MS and VLR/SGSN if the MS has visited the area of the VLR/SGSN already. After the IAKA process, the MS and the VLR/SGSN share the new session pair key (*NCK*, *NIK*). These keys are hidden from the sight of adversary and we show that they could not be guessed or eavesdropped easily.

The TMSI/IMSI fields are sent public without any encryption because anonymity service is out of the scope of the article. Also RAND fields are sent as plain text because they are needed for the other side to generate necessary fields. The sequence numbers, UEA, and UIA fields are encrypted by the shared keys (K or *NCK*). So the confidentiality service is provided properly. Also *MAC* fields are protected by the shared keys (K, *OIK* or *NIK*) and hence the integrity service is afforded appropriately.

The HE could authenticate the identity (IMSI) of the MS by verifying the integrity of the $MAC_{MS,K}$ generated by the MS. The MS could confirm the identity of its home network by checking the $MAC_{HE,K}$ calculated by the HE. So two parties authenticate each other suitably referred to mutual authentication. Besides, the MS and the VLR/SGSN could authenticate each other by verifying the $MAC_{MS,OIK}$, $MAC_{MS,NIK}$ and $MAC_{VLR,NIK}$, as the session keys (*OIK, NIK*) could be computed or available in both sides (MS side and VLR/HE side). So the mutual authentication with security mode set-up is done by the EAKAP.

Sequence Number Protection vs. Private Key Derivation

MS increments the SQN_{MS} for every IAKA procedure and HE runs the SQN_{HE} for every generated *AV*. MS and HE use sequence number and random number generator to achieve strong freshness of their messages. However pseudo-random numbers could provide the freshness of the ways of the EAKAP, but both MS and HE need a large directory to save used random numbers in it. Also to avoid the usage of the repeated random numbers they have to search in their directory or sort and update the directory for every procedure that it consumes a large amount of energy and memory.

In the EAKAP, anonymity protection of the sequence number is higher than the UMTS AKA because the sequence number of the UMTS AKA is XOR-ed by anonymity key derived from nonce and the private key [2] like stream ciphers. As sequence number value is guessable because it starts from zero and changes incrementally slowly, the known plaintext attack is imaginable on the *f5* algorithm to get private key (K). However the core of the *f5* algorithm is based on Advanced Encryption Standard (AES) [4] that the recent proposed attacks are not practical on it [5]. In the EAKAP, the sequence number is encrypted by a block cipher algorithm called *fc* with derived key from *f5* algorithm and so the privacy of the sequence number is strongly established.

Availability (Robustness against DoS Attack)

If the MS enters area of a new VLR/SGSN, it won't have a TMSI allocated by VLR/SGSN. So the MS must use its IMSI to initiate the procedure. In this situation, the MS actually aims its HE not VLR/SGSN. In fact SRNC and VLR/SGSN role as relays to forward the MS messages to the HE. Here we don't struggle with the privacy establishment of the MS.

First, we consider the condition that no TMSI was set before. The MS and the VLR/SGSN have no shared key before and so, as mentioned in step1 of the previous section, the *AREQ* message has been sent to the HE via the intermediate SRNC and VLR/SGSN. As mentioned in step3, the HE could check the integrity of the $MAC_{MS,K}$, so the spam messages generated by spurious users are discarded by the HE. Furthermore, the DoS attack organized by unauthentic MS at the first step, could be detected at the HE side and no more traffic would be procreated in the network.

Second, we consider the situation where the VLR/SGSN has allocated the MS a TMSI so the MS and the VLR/SGSN have shared a pair key (*OCK* and *OIK*) with each other. Consequently, as explained in step2, the VLR/SGSN checks the integrity of the $MAC_{MS,OIK}$ and rejects the forged messages. Furthermore, with the usage of the pre-shared information of the MS, the network could prevent DoS attack of first step at the VLR/SGSN side.

In the other steps of the EAKAP, by using MAC mechanism, the vulnerability towards DoS attack is reduced as mentioned above. Although in some cases, the *SQN* should be computed before the MAC so the computation load would be increased. But no further signaling load would be injected to the network and the elements of the network could decide about the genuineness of the message by verifying the MAC integrity.

Mutual Freshness and Unguessable Keys

The SQN_{MS} is encrypted by temporary anonymity key (*TAK*) derived from the output of the *f5* algorithm by a high entropy seed which is $IMSI||RAND_{MS}$. The SQN_{HE} is protected by anonymity key (*AK*) derived from the *f5* algorithm by concatenation of the IMSI, $RAND_{MS}$ and $RAND_{HE}$ that is an entropic seed to generate a fresh key. The seed entropy of the *f5* algorithm depends on two random numbers $RAND_{MS}$ and $RAND_{HE}$ and hence is improbable to be guessed by adversary. So the sequence numbers could not be revealed. Also if an anonymity key is compromised at a session, no next-generated keys will be concealed i.e. our scheme supports forward security towards session key compromising. The MAC fields like the *SQN* fields are fresh. The entropic seed of the MAC fields is derived from the pseudo random numbers as well as IMSI and sequence number.

6 Conclusion

Most of the proposed protocols do not consider the previous and next stages of the UMTS AKA protocol and they try to improve the UMTS AKA protocol solely. Our proposed protocol, so called EAKAP, merges all of the stages in five ways so it improves performance by reducing the load of the network signaling. Also EAKAP support mutual freshness and is more robust against DoS attack by applying MAC mechanism between the MS and the network.

References

1. 3GPP TS 33.102 V8.0.0 (2008-06), 3GPP Technical Specification Group Services and System Aspects, 3G Security, Security Architecture (Release 8)
2. 3GPP TS 35.206 V7.0.0 (2007-06), Technical Specification Group Services and System Aspects, 3G Security, Algorithm Specification (Release 7)
3. Stallings, W.: Cryptography and Network Security: Principles and Practice. Prentice-Hall, Englewood Cliffs (1998)
4. National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), Federal Information Processing Standards Publications (FIPS PUBS) 197 (2001)
5. Dobbertin, H., Knudsen, L., Robshaw, M.: The cryptanalysis of the AES - A brief survey. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 1–10. Springer, Heidelberg (2005)
6. Shannon, C.E.: A Mathematical Theory of Communication. The Bell System Technical Journal 27, 379–423, 623–656 (1948)
7. Huang, C.M., Li, J.W.: Authentication and Key Agreement Protocol for UMTS with Low Bandwidth Consumption. In: Proceedings of the 19th International Conference on Advanced Information Networking and Applications (2005)
8. Juang, W.S., Wu, J.L.: Efficient 3GPP Authentication and Key Agreement with Robust User Privacy Protection. In: IEEE Communications Society. Proceedings of the WCNC (2007)
9. Al-Saraireh, J., Yousef, S.: A New Authentication Protocol for UMTS Mobile Networks. EURASIP Journal on Wireless Communications and Networking, Article ID 98107, 1–10 (2006)
10. Meyer, U., Wetzel, S.: A Man-in-the-Middle Attack on UMTS. In: Proceedings of the 3rd ACM workshop on Wireless security, pp. 90–97 (2004) ISBN:1-58113-925-X
11. Harn, L., Hsin, W.J.: On the Security of Wireless Network Access with Enhancements. In: Proceedings of the 2nd ACM workshop on Wireless security, pp. 88–95 (2003) ISBN:1-58113-769-9
12. AL-Saraireh, J., Yousef, S.: Extension of Authentication and Key Agreement Protocol (AKA) for Universal Mobile Telecommunication System (UMTS). International Journal of Theoretical and Applied Computer Sciences 1(1), 109–118 (2006)
13. Zhang, M., Fang, Y.: Security Analysis and Enhancements of 3GPP: Authentication and Key Agreement Protocol. IEEE Transactions on Wireless Communications 4(2) (March 2005)

APPENDIX (Abbreviations)

3GPP	Third-Generation Partnership Project
AK	Anonymity Key
AKA	Authentication and Key Agreement
AMF	Authentication Management Field
AREP	Authentication Reply
AREQ	Authentication Request
AuC	Authentication Center
AUTN	Authentication Token
AV	Authentication Vector

BSC	Base Station Controller
BTS	Base Transceiver Station
CK	Cipher Key
CM	Connection Management
CS	Circuit Switched
DoS	Denial of Service
EAKAP	Extended AKA Protocol
FRESH	Pseudo-Random Number Generated by MS
GSM	Global System for Mobile
HE	Home Environment included HLR and AuC
HLR	Home Location Register
HN	Home Network
IAKA	Identification and AKA
IK	Integrity Key
IMSI	International Mobile Subscriber Identity
KSI	Key Set Identifier
MAC	Message Authentication Code
MAC-I	Message Authentication Code for Integrity
MS	Mobile Station
MSC	Mobile Switching Center
PS	Packet Switched
$RAND_X$	Pseudo-Random Number Generated by X
RES	Response
SGSN	Serving GPRS Support Node
SMCM	Security Mode Complete Message
SQN	Sequence Number
SRNC	Serving Radio Network Controller
TAK	Temporary Anonymity Key
TIMSI	TMSI or IMSI
TMSI	Temporary Mobile Subscriber Identity
UEA	UMTS Encryption Algorithm
UIA	UMTS Integrity Algorithm
UMTS	Universal Mobile Telecommunications System
USIM	Universal Subscriber Identity Module
VLR	Visitor Location Register

Hash-Based Key Management Schemes for MPEG4-FGS

Mohamed Karroumi and Ayoub Massoudi

Thomson R&D France
Technology Group, Corporate Research, Security Laboratories
1 avenue de Belle Fontaine, 35576 Cesson-Sévigné Cedex, France
{mohamed.karroumi,ayoub.massoudi}@thomson.net

Abstract. We propose two symmetric-key management schemes for the encryption of scalable compressed video content. The schemes are applicable to MPEG-4 Fine Grain Scalability video coding. Our constructions make only use of hash functions and achieve the optimal bound regarding the minimum number of keys and the time complexity in computing all the decryption keys. We also formalize new security notions about collusion-resistance. Unlike prior key management schemes, our second scheme resists to certain collusion attacks. The collusion-resistance achieved is practical and hence sufficient for encryption of scalable video streams.

Keywords: symmetric-key encryption, key management, MPEG-4 Fine Grain Scalability.

1 Introduction

Scalability has become a fundamental feature in video coding. It allows one to encode a video once and to send it to many users with different usage profiles and capabilities (high-end PCs, high-definition displays, cell phones,...). Each user has access to a particular portion or quality of the content. Efficient access control methods are required in scalable video transmission in order to grant access only to authorized users for the video quality they purchased for.

In scalable video coding, the compressed bitstream is composed of two layers: a non scalable part called base layer and a scalable part containing many enhancement layers. If only the base layer is decoded, this will result in low quality version of the original content. If the enhancement layers are decoded, combined with the base layer, this produces an improved quality of the video, proportional to the Enhancement portion decoded. MPEG-4 Fine Grain Scalability (MPEG4-FGS) [4] provides flexibility in supporting:

– PSNR scalability by defining different quality bitplanes;
– Bitrate scalability by arbitrary bitplanes truncation.

Scalable compressed video is particularly adapted to flexible services provided by DRM (Digital Rights Management) technologies [1,5]. Scalable encryption techniques if combined with a suitable key management scheme enables to improve access control processes as proposed for MPEG4-FGS [8].

To enforce a scalable access control, a typical approach is to encrypt the different portions in the enhancement layers with a different encryption key. The keys will be distributed selectively to legitimate users who should access only the keys that render the authorized video quality. In other words, the scalable access control problem for the video content is transformed into scalable access control on the encryption key.

In this paper, we focus on a layered access control scheme called Scalable Multi-Layer FGS Encryption (SMLFE) [8,10] that supports both PSNR and bit-rate scalability for the MPEG4-FGS coding. In SMLFE, the enhancement layers are encrypted into a single stream with multiple quality layers divided according to PSNR values and bit-rates. Enhancement layers for each frame are partitioned into n bit-rates and t PSNR layers independently.

Background and Related Works. In the trivial scheme [8], each segment $\mathsf{S}_{i,j}(1 \leq i \leq n,\ 1 \leq j \leq t)$, is at the intersection of one bit-rate and one PSNR layer and is encrypted with a different *segment encryption key*. Therefore, $n \cdot t$ segment keys are randomly generated to encrypt all the segments. The protected content is next transferred to an appropriate server (distribution server) for the distribution while the $n \cdot t$ segment keys are stored in a license server. When the user receives the protected MPEG4-FGS content he requests for a license. The license server generates it according to the quality and/or resolution requested. For instance, if a user acquires the rights for the full quality of the video, license server sends all the $n \cdot t$ segment decryption keys. In this trivial scheme, the license server has to securely store all the $n \cdot t$ segment keys.

Another method based on the Diffie-Hellman key agreement protocol is described in [9]. In this scheme, only two secret keys are maintained by the license server (instead of $n \cdot t$). for the full content quality, n public keys sent along

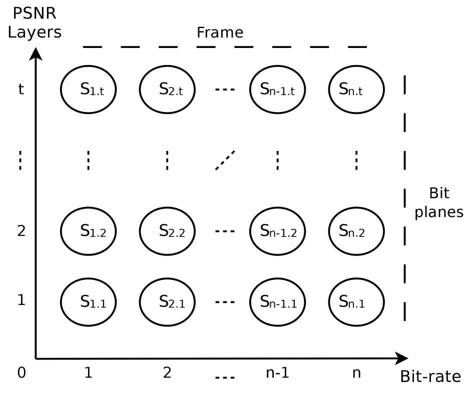


Fig. 1. Segments in the Enhancement Layers

Table 1. Comparing Key Management Schemes Efficiency

Schemes	Number of Keys			Time-complexity	
Symmetric Key	server	user	length	hashes	exponentiations
Trivial	$n \cdot t$	$n \cdot t$	128 bits	0	0
Basic Scheme	1	1	128 bits	$n + t - 2$	0
Enhanced Scheme	2	2	128 bits	$n + t + 2^{n-1} + 2^{t-1} - 4$	0
Asymmetric Key	server	user	length	hashes	exponentiations
ZFL	2	$n + 1$	256 bits	$n + t - 2$	$n \cdot t$

with the protected content plus one secret key, sent in the license, are needed to compute all the segment keys. This system makes use of the hybrid encryption method where the asymmetric algorithm helps in computing a symmetric key used for segments encryption.

Notation. A $2l$-bit integer x is an integer such that $2^{2l-1} \leq x < 2^{2l}$. By $\lfloor x \rfloor$ we mean the highest integer less than or equal to x, i.e. $\lfloor x \rfloor = \max \{n \in \mathbb{N} \mid n \leq x\}$. The l most significant bits of x are noted $\{x\}_L$ and l least significant bits $\{x\}_R$, i.e $x = \{x\}_L \| \{x\}_R$ and could be written $\{x\}_L 2^l + \{x\}_R$. To ease the presentation, we do not distinguish between an integer and its representation.
For a set $\mathcal{S}$, we denote by $|\mathcal{S}|$ the number of elements in $\mathcal{S}$.

Our contribution. In this work, we describe two symmetric-key management schemes for scalable multilayer fine grain scalability encryption. In the proposed schemes, we consider that a user having access to a given high resolution should access all lowest resolutions, and similarly if he has access to a given quality he should also have access to lowest qualities.

The "basic" scheme is the simplest construction that satisfies the SMLFE architecture. It is the most efficient in terms of number of keys sent to user and time complexity in recovering the segment decryption keys. The basic scheme is however vulnerable to some collusion attacks. We therefore propose the "enhanced" scheme to address the collusion problem.

Table 1 compares our schemes with the previous proposal in terms of number of keys maintained by the license server, number of keys and time complexity needed at user side to recover all the segment keys. We do not distinguish between secret keys and public keys sent along with the content. We consider the symmetric-key trivial scheme proposed by Yuan et al. [8] as well as the public-key scheme ZFL proposed by Zhu et al. [9]. Regarding the key lengths, asymmetric algorithm keys must be longer for equivalent resistance to attack than symmetric keys. To compare systems with roughly the same level of security, 128 bits for the symmetric-key schemes are compared with 256 bits for ZFL public key scheme using elliptic curves [6].

2 Basic Scheme

In this section, we propose a basic key management scheme. The scheme design is based on an iterative application of a one-way function. The one-way chain idea has been earlier proposed by Lamport in [3] to construct a one-time password scheme. [2] generalizes the idea behind Lamport's scheme through the use of several one-way functions.

Definition 1. *A key scheme respecting the SMLFE architecture is said* simply secure *if the knowledge of any secret key $k_{i,j}$ associated to $\mathsf{S}_{i,j}$ does not permit to get the secret key of any higher segment $\mathsf{S}_{k,l}$ such that $k > i$ or $l > j$.*

The defintion here tells that the access to a given low quality of the video content should not lead to the access of a higher quality of this video.

Let us consider that there are $n \cdot t$ different segments in the enhancement layers. Then each different segment will be encrypted with a different key. More precisely, the content provider generates at random a $2l$-bit master encryption key K. This key encrypts the segment $\mathsf{S}_{n,t}$ that is in the higher quality level for both parameters PSNR and bit-rate. Segment keys $k_{i,j}$, which encrypt lower quality are computed recursively as follow:

$$k_{i,j} = \mathsf{h}^{(n-i)}(\{K\}_L) \parallel \mathsf{h}^{(t-j)}(\{K\}_R) , \tag{1}$$

where $\mathsf{h} : \{0,1\}^l \rightarrow \{0,1\}^l$ is a one-way (hash) function. $\mathsf{h}^{(n)}(x)$ denotes the result after applying n times the one-way function h to x.
The master key K is sent to the license server and encrypted content to distribution server.

At the other end of the process, a user acquires a given quality of the video. Let us consider that the requested quality is obtained by accessing segments below PSNR level s and below bit-rate r. Then, the license server computes using formula (1)

$$K_{r,s} = \mathsf{h}^{(n-r)}(\{K\}_L) \parallel \mathsf{h}^{(t-s)}(\{K\}_R) ,$$

and sends it to user in a license. From this key, the user is able to compute the $r \cdot s$ keys of lower quality segments.

Proposition 1. *The basic scheme is* simply secure*.*

Proof. This immediately follows from the one-wayness of h. If an attacker knows a key $k_{i,j}$, finding $k_{x,y}$ for some $x > i$ or some $y > j$ requires the inversion of the one-way function h which is assumed to be computationally infeasible. $\square$

We now discuss about the complexity of the scheme. From equation (1) we can see that it requires for a user at most $n + t - 2$ calls to a one-way function to compute all the segment keys. It also requires for the license server to maintain one key and whatever the quality of the scalable content to transmit only one key to user.

Another advantage of the basic scheme is the possibility to derive from one secret value two orthogonal segment decryption keys. The license server have then the possibility to use both parameters PSNR and bit-rate to answer a requested quality. In the ZFL scheme, public keys are distributed with the encrypted content, then the license server can later only use one parameter (i.e. PSNR) for varying the quality of the content.

A security concern in SMLFE is the *collusion-resistance*. A user could purchase different low quality versions of a video and combine the encryption keys in order to compromise the full quality of the video. Encryption keys should then be constructed to protect high quality keys from collusions of low quality encryption keys. The basic scheme is not resistant to such collusion attacks.

3 Collusion Resistance

We now formalize the model of a collusive attack against a key management scheme for SMLFE.

Definition 2 (Basis). *A* basis *of a set $\mathcal{U}$, containing different segment keys of the enhancement layers, is the minimal set of segment keys that permits to compute all other segment keys in $\mathcal{U}$.*

Remark 1. A basis may contain one single key. Any set $\mathcal{U}$ can be then represented by its basis $(k_1, k_2, \ldots, k_m)$.

Definition 3 (Collusion-Resistance). *A key management scheme is called* collusion-resistant *(CR) to any set $\mathcal{C}$ containing low quality segments (PSNR or Bit-rate), if for every high quality segment $\mathsf{S}_{i,j} \notin \mathcal{C}$, the knowledge of all secrets associated to segments in $\mathcal{C}$ does not permit to obtain the secret associated to $\mathsf{S}_{i,j}$.*

The definition states that if a coalition of users has the keys associated to segments in $\mathcal{C} = \{\mathsf{S}_1, \mathsf{S}_2, \ldots, \mathsf{S}_k\}$ then the sharing of secrets by a coalition should not permit to get an advantage of knowing segment keys that are not in $\mathcal{C}$.

Definition 4 (Orthogonality). *Two different segment $\mathsf{S}_{i,j}$ and $\mathsf{S}_{x,y}$ are* orthogonal *when the right to access one of the two segments does not necessarily give the right access the other and vice-versa. A sequence of k different segments is said orthogonal if any two different segments in the k-sequence are orthogonal.*

Definition 5 (Orthogonal Collusion-Resistance). *The collusion-resistance is* orthogonal *(OCR) if the colluders set keys is generated by at least 2 keys and the knowledge of any couple $(k_{i,j}, k_{u,v})$ in the basis does not permit to obtain the secrets associated to any segment in $\mathcal{S}$, with*

$$\mathcal{S} = \big\{\{\mathsf{S}_{x,y} \mid i < x \leq n \text{ or } j < y \leq t)\} \cap \{\mathsf{S}_{x,y} \mid u < x \leq n \text{ or } v < y \leq t\}\big\} .$$

A scheme is called k-OCR if it is collusion-resistant to any set $\mathcal{C}$ generated by at most k orthogonal segments. Out of these general definitions about collusion-resistance we will only consider in this paper collusion with 2 orthogonal segment decryption keys.

The scheme described in previous section is not 2-OCR since knowing any two orthogonal keys permits to compute a higher quality segment key. For instance, if an adversary query two orthogonal decryption keys of the form

$$k_{n,j} = \{K\}_L \mathbin{\|} \mathsf{h}^{(t-j)}(\{K\}_R), \qquad k_{i,t} = \mathsf{h}^{(n-i)}(\{K\}_L) \mathbin{\|} \{K\}_R$$

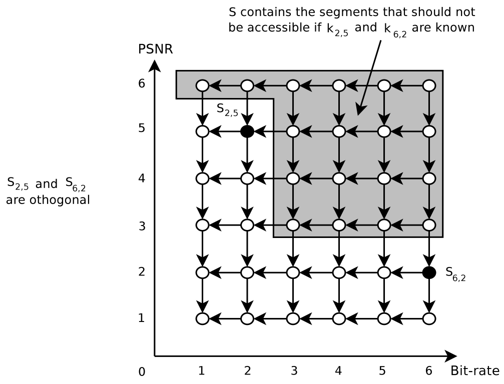

S contains the segments that should not be accessible if k2,5 and k6,2 are known
PSNR
S2,5
S6,2
S2,5 and S6,2 are othogonal
Bit-rate
0 1 2 3 4 5 6
1 2 3 4 5 6

Fig. 2. Orthogonality and Orthogonal Collusion-Resistance

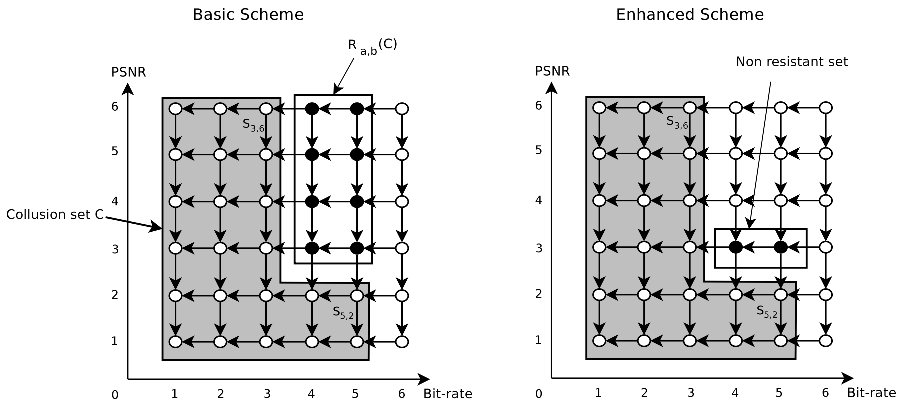


Fig. 3. Non 2-OCR Segments in Basic and Enhanced Scheme

he is able to reconstruct the key $K = \{K\}_L \parallel \{K\}_R$ and to access all segments in the enhancement layers.

The segments that do not resist to a collusion set generated by $(k_{i,j}, k_{u,v})$ is the rectangle $\mathcal{R}_{a,b}(\mathcal{C})$ containing the segments $\{\mathsf{S}_{a,b}\}$ such that

$$\min(i,u) < a \leq \max(i,u); \quad \min(j,v) < b \leq \max(j,v).$$

There are then $\big(\max(i,u) - \min(i,u)\big) \cdot \big(\max(j,v) - \min(j,v)\big)$ segments that are vulnerable to the collusion. All other segments not in $\mathcal{C}$ and outside the rectangle $\mathcal{R}_{a,b}(\mathcal{C})$ remain resistant to the collusion. Figure 4 illustrates an example where $\mathcal{C} = (k_{3,6}, k_{5,2})$.

The maximum number of the non collusion-resistant segments is reached for $\mathcal{C} = (k_{n,1}, k_{1,t})$ where there are $(n-1) \cdot (t-1)$ non collusion-resistant segments keys. In the next section we proposes a scheme that better resists to a 2-orthogonal collusion by reducing the number of non collusion-resistant segments.

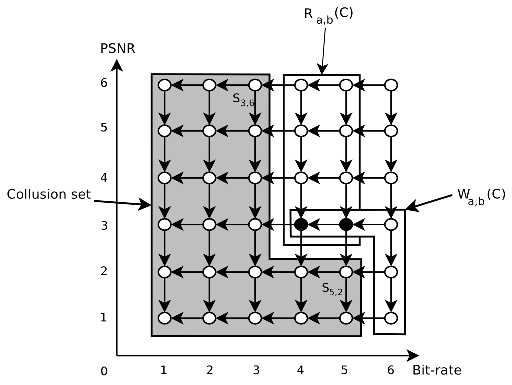


Fig. 4. Non Collusion-Resistant Set

4 Enhanced Scheme

In this section, we propose an enhanced key management scheme. We brought some modifications to the basic scheme to get better collusion-resistance properties.

The content provider generates randomly two $2l$-bits master encryption keys K_1 and K_2. $K_1 \oplus K_2$ is the key that encrypts the segment $\mathsf{S}_{n,t}$ and the values of n and t are made public. Segment keys $k_{i,j}$ are computed by the license server as follows:

- compute $k^1_{i,j}$ from K_1 using equation (1),
- compute next $z(i,j)$:

$$z(i,j) = \begin{cases} 0 & \text{if } \; n-i=0 \text{ and } \; t-j=0 \; , \\ 2^{n-i}-1 & \text{if } \; t-j=0 \text{ and } \; n-i \neq 0 \; , \\ 2^{t-j}-1 & \text{if } \; n-i=0 \text{ and } \; t-j \neq 0 \; , \\ 2^{n-i}+2^{t-j}-2 & \text{for } 1 \leq i < n \text{ and } \; 1 \leq j < t \; . \end{cases} \tag{2}$$

- compute $k^2_{i,j}$ from K_2:

$$k^2_{i,j} = \mathsf{h}_2^{(z(i,j))}(K_2) \text{ where } \mathsf{h}_2 : \{0,1\}^{2l} \rightarrow \{0,1\}^{2l} \; ,$$

- segment key $k_{i,j}$ is calculated by generating $k_{i,j} = k^1_{i,j} \oplus k^2_{i,j}$.

For the quality corresponding to PSNR level s and bit-rate level r, license server delivers to user two keys $K^1_{r,s}$ and $K^2_{r,s}$ such that:

$$K^1_{r,s} = \mathsf{h}_1^{(n-r)}(\{K_1\}_L) \,\|\, \mathsf{h}_1^{(t-s)}(\{K_1\}_R) \; , \qquad K^2_{r,s} = \mathsf{h}_2^{(z(r,s))}(K_2) \; ,$$

where $z(r,s)$ is calculated using formula (2).

From these two keys, the user is able to compute all necessary $r \cdot s$ segment decryption keys. Let us suppose that the user want to compute a lower-segment key $k_{\alpha,\beta}$. From $k^1_{\alpha,\beta}$, it is easy for the user to guess the number of times that the one-way function h_1 should be applied to the $K^1_{r,s}$ knowing α, β, r and s. Indeed,

$$k^1_{\alpha,\beta} = \mathsf{h}_1^{(r-\alpha)}(\{K^1_{r,s}\}_L) \parallel \mathsf{h}_1^{(s-\beta)}(\{K^1_{r,s}\}_R) \ .$$

However, it is not as easy for the second part of the key $k^2_{\alpha,\beta}$. The formula (2) gives the method to compute the value of $z(\alpha,\beta)$ in case the derivation is done using the master key K_2. The derivation process can also be performed in a relative way, i.e from a key $K^2_{r,s}$ where $K^2_{r,s}$ is not necessarily the second master key. Let $\Delta_{(r,s)}z(\alpha,\beta)$ an integer that depends on the distance between $\mathsf{S}_{r,s}$ and $\mathsf{S}_{\alpha,\beta}$, the key $k^2_{\alpha,\beta}$ is calculated as follow:

– compute $z(r,s)$ for $K^2_{r,s}$ using formula (2), e.g. $z(r,s) = 2^{n-r} + 2^{t-s} - 2$,
– compute $z(\alpha,\beta)$ for $k^2_{\alpha,\beta}$ using formula (2), e.g. $z(\alpha,\beta) = 2^{n-\alpha} - 1$,
– compute $\Delta_{(r,s)}z(\alpha,\beta) = z(\alpha,\beta) - z(r,s)$,
– generate $k^2_{\alpha,\beta} = \mathsf{h}_2^{(\Delta_{(r,s)}z(\alpha,\beta))}(K^2_{r,s})$.

The user needs to know n and t (public values) to compute each lower segment key $k^2_{\alpha,\beta}$. Finally, $k_{\alpha,\beta}$ is calculated by generating $k_{\alpha,\beta} = k^1_{\alpha,\beta} \oplus k^2_{\alpha,\beta}$.

In this scheme, the user needs at most $n+t+2^{n-r}+2^{t-s}-4$ calls to one-way functions for the computation of all the segment keys. In addition, all segment keys are derived recursively from two common master keys, thus the license server maintains two keys and whatever the quality requested by user only two keys are transmitted. A use case example is given in the Appendix A.

We figure out that there are particular cases where the enhanced scheme is 2-OCR. These are cases where all the segment keys that should not be accessible remain resistant to a 2-orthogonal collusion attack. This is formalized by proposition 2.

Proposition 2. *Enhanced scheme is* 2*-OCR for any set* $\mathcal{C} = (k_{i,j}, k_{u,v})$ *such that* $n - i \neq t - j$ *and either* $n - i = t - v$ *or* $t - j = n - u$*.*

Proof. See Appendix A. □

Proposition 2 states that the enhanced scheme is 2-orthogonal collusion-resistant in the case $n - i = t - v$ or $t - j = n - u$. There are no other cases where this scheme is 2-orthogonal collusion-resistance. To compare it with the basic scheme, we studied the number of segments keys that are not 2-OCR in the general case, i.e when $n-i \neq t-v$ and $t-j \neq n-u$. Proposition 3 gives the maximum number of the non 2-OCR segments keys and figure 3 illustrates an example where non 2-OCR segments are filled with black.

Proposition 3. *For all* $(n > 1, t > 1)$*, the number of segments keys that are not resistant to any* 2*-orthogonal collusion is bounded by* $\lfloor \frac{t}{2} \rfloor^2$ *if* $n \leq t$ *or* $\lfloor \frac{n}{2} \rfloor^2$ *if* $n > t$ *.*

Proof. See Appendix A. □

In proposition 3 the number of non resistant segments keys is bounded by a smaller value. Enhanced scheme has therefore better collusion-resistance properties than basic scheme.

5 Conclusion

We have proposed two symmetric-key management schemes for protecting a multilayer scalable part of an MPEG4-FGS video content. Compared to previous work, our schemes reduces the number of keys managed by a license server and the number of keys needed by the user to access the protected scalable video. On the top of requiring less computational resources, symmetric algorithms permit to handle short keys as well, which makes the proposed schemes attractive and practical.

Besides, we focused on collusion-resistance and enhanced scheme has interesting collusion-resistance property. An open problem is the design of a symmetric-key system that is at least as efficient as the enhanced scheme and which remains fully secure no matter how many orthogonal segment keys are at the disposal of the attacker.

Acknowledgments. We thank Alain Durand, Marc Joye and Davide Alessio for valuable discussions and suggestions. We also thank Eric Diehl for helpful comments.

References

1. Hanaoka, G., Ogawa, K., Murota, I., Ohtake, G., Majima, K., Oyamada, K., Gohshi, S., Namba, S., Imai, H.: Separating encryption and key issuance in digital rights management systems. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 365–376. Springer, Heidelberg (2003)
2. Joye, M., Yen, S.-M.: One-way cross-trees and their applications. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 346–356. Springer, Heidelberg (2002)
3. Lamport, L.: Password authentication with insecure communication. Commun. ACM 24(11), 770–772 (1981)
4. Li, W.: Overview of fine granularity scalability in MPEG-4 video standard. IEEE Trans. Circuits Syst. Video Techn. 11(3), 301–317 (2001)
5. Liu, Q., Safavi-Naini, R., Sheppard, N.P.: Digital rights management for content distribution. In: Johnson, C., Montague, P., Steketee, C. (eds.) ACSW Frontiers. CRPIT, vol. 21, pp. 49–58. Australian Computer Society (2003)
6. ECRYPT Yearly Report on Algorithms and Keysizes. D.SPA.10 Rev. 1.1. IST-2002-507932 ECRYPT (2006) (January 2007), `http://www.ecrypt.eu.org/documents/D.SPA.21-1.1.pdf`
7. Rivest, R.: The MD5 Message - digert Algorithm. Internet activities board, internet privacy task force, request for comments, RFC 1321 (1992)
8. Yuan, C., Zhu, B.B., Su, M., Wang, X., Li, S., Zhong, Y.: Layered access control for MPEG-4 fgs video. In: ICIP (1), pp. 517–520 (2003)
9. Zhu, B.B., Feng, M., Li, S.: An efficient key scheme for layered access control of mpeg-4 fgs video. In: ICME, pp. 443–446. IEEE, Los Alamitos (2004)
10. Zhu, B.B., Yuan, C., Wang, Y., Li, S.: Scalable protection for MPEG-4 fine granularity scalability. IEEE Transactions on Multimedia 7(2), 222–233 (2005)

A Appendix

A.1 Enhanced Scheme Example

In this section, we give an example for the enhanced scheme where $n = 5$ and $t = 4$. The content provider generates two random 128-bit keys K_1 and K_2, which are delivered to license server. $K_1 \oplus K_2$ is the key that encrypts the segment $\mathsf{S}_{5,4}$. Segment $\mathsf{S}_{i,j}$, where $1 \leq i \leq 5$ and $1 \leq j \leq 4$, is encrypted with a key that results from a XOR between $K^1_{i,j}$ and $K^2_{i,j}$. Table 2 illustrates how these values are computed.

Table 2. Segment Decryption Keys

$$\boxed{\text{Computation of } K^1_{i,j}}$$

$$\begin{tikzcd}
\mathsf{h}_1^4(L)\|R \arrow[d] & \mathsf{h}_1^3(L)\|R \arrow[l] \arrow[d] & \mathsf{h}_1^2(L)\|R \arrow[l] \arrow[d] & \mathsf{h}_1(L)\|R \arrow[l] \arrow[d] & K_1 = L\|R \arrow[l] \arrow[d] \\
\mathsf{h}_1^4(L)\|\mathsf{h}_1(R) \arrow[d] & \mathsf{h}_1^3(L)\|\mathsf{h}_1(R) \arrow[l] \arrow[d] & \mathsf{h}_1^2(L)\|\mathsf{h}_1(R) \arrow[l] \arrow[d] & \mathsf{h}_1(L)\|\mathsf{h}_1(R) \arrow[l] \arrow[d] & L\|\mathsf{h}_1(R) \arrow[l] \arrow[d] \\
\mathsf{h}_1^4(L)\|\mathsf{h}_1^2(R) \arrow[d] & \mathsf{h}_1^3(L)\|\mathsf{h}_1^2(R) \arrow[l] \arrow[d] & \mathsf{h}_1^2(L)\|\mathsf{h}_1^2(R) \arrow[l] \arrow[d] & \mathsf{h}_1(L)\|\mathsf{h}_1^2(R) \arrow[l] \arrow[d] & L\|\mathsf{h}_1^2(R) \arrow[l] \arrow[d] \\
\mathsf{h}_1^4(L)\|\mathsf{h}_1^3(R) & \mathsf{h}_1^3(L)\|\mathsf{h}_1^3(R) \arrow[l] & \mathsf{h}_1^2(L)\|\mathsf{h}_1^3(R) \arrow[l] & \mathsf{h}_1(L)\|\mathsf{h}_1^3(R) \arrow[l] & L\|\mathsf{h}_1^3(R) \arrow[l]
\end{tikzcd}$$

$$\boxed{\text{Computation of } K^2_{i,j}}$$

$$\begin{tikzcd}
\mathsf{h}_2^{15}(K_2) \arrow[d] & \mathsf{h}_2^7(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2^3(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2(K_2) \arrow[l] \arrow[d] & K_2 \arrow[l] \arrow[d] \\
\mathsf{h}_2^{16}(K_2) \arrow[d] & \mathsf{h}_2^8(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2^4(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2^2(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2(K_2) \arrow[l] \arrow[d] \\
\mathsf{h}_2^{18}(K_2) \arrow[d] & \mathsf{h}_2^{10}(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2^6(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2^4(K_2) \arrow[l] \arrow[d] & \mathsf{h}_2^3(K_2) \arrow[l] \arrow[d] \\
\mathsf{h}_2^{22}(K_2) & \mathsf{h}_2^{14}(K_2) \arrow[l] & \mathsf{h}_2^{10}(K_2) \arrow[l] & \mathsf{h}_2^8(K_2) \arrow[l] & \mathsf{h}_2^7(K_2) \arrow[l]
\end{tikzcd}$$

One way functions h_1 and h_2 can use MD5 [7], which provides 128-bit ouput size[1]. Computing all the segment encryption keys will then require 41 MD5 hashes and 20 XOR operations.

For the quality associated to PSNR level 3 and bit-rate level 2, license server delivers to user two keys: $K^1_{2,3} = \mathsf{h}_1^3(L) \parallel \mathsf{h}_1(R)$ and $K^2_{2,3} = \mathsf{h}_2^8(K_2)$. The user is next able to compute $K^1_{2,2}$, $K^1_{2,1}$, $K^1_{1,3}$, $K^1_{1,2}$ and $K^1_{1,1}$ by applying h_1 to $K^1_{2,3}$ as illustrated in the table 2. Knowing the value of n and t and the coordinates of $\mathsf{S}_{2,3}$ i.e. $(2, 3)$, the user finds the number of times the one-way function h_2 is applied to $K^2_{2,3}$ and obtains the other keys $K^2_{2,2}$, $K^2_{2,1}$, $K^2_{1,3}$, $K^2_{1,2}$ and $K^2_{1,1}$. Finally, segment decryption keys are calculated by generating $K_{i,j} = K^1_{i,j} \oplus K^2_{i,j}$ with $1 \leq i \leq 2$ and $1 \leq j \leq 3$. In this example, the user performs 19 MD5 hashes and 6 XOR operations.

[1] Given that a smaller input/output size is needed for h_1, we can apply the hash function to 64 bits part of the key to be hashed and the resulting output is truncated by selecting leftmost 64 bits and discarding the rightmost 64 bits output.

A.2 Proof of Proposition 2

Since $k_{i,j}$ and $k_{u,v}$ are orthogonal either $i < n$ or $j < t$. Let suppose that $i < n$. If $n - i = t - v$ and $t - j = n - u$ then from (2) we have

$$k_{i,j}^2 = k_{u,v}^2 = \begin{cases} \mathsf{h}_2^{(2^{n-i}-1)}(K_2), \text{ if } t - j = 0 \\ \mathsf{h}_2^{(2^{n-i}+2^{t-j}-2)}(K_2), \text{ otherwise} . \end{cases} \tag{3}$$

If $n - i \neq t - v$ or $t - j \neq n - u$ then $\exists (m_1, m_2) \in \{1, \ldots, 2^{n-1} + 2^{t-1} - 2\}^2$ such that $k_{i,j}^2 = \mathsf{h}_2^{(m_1)}(K_2)$ and $k_{u,v}^2 = \mathsf{h}_2^{(m_2)}(K_2)$ with either $m_1 < m_2$ or $m_1 > m_2$. The knowledge of one key enables to compute the other one. It is then sufficient to consider $k_{i,j}^2$ by supposing $m_1 < m_2$.

$k_{i,j}^1$ and $k_{u,v}^1$ do not reveal the keys that are outside $\mathcal{R}_{a,b}$. It suffices then to find the keys in $\mathcal{R}_{a,b}$ that can be computed when $k_{i,j}^2$ is known. In the case $n - i = t - v$ or $t - j = n - u$, $\mathcal{R}_{a,b}$ contains the segment $\mathsf{S}_{a,b}$ such that

$$\min(i, j) < a, b \leq \max(i, j)$$

$$\Updownarrow$$

$$\min(n - i, t - j) \leq n - a, t - b < \max(n - i, t - j) .$$

Consider that $n - i = \max(n - i, t - j)$. The segment key $\{k_{a',b'}\}_2 \in \mathcal{R}_{a,b}$ that has the greatest hash exponent verifies $(n - a', t - b') = (n - i - 1, n - i - 1)$

$$\{k_{a',b'}\}_2 = \begin{cases} K_2, \text{ if } n - i - 1 = 0 \\ \mathsf{h}_2^{(2^{n-i}-2)}(K_2), \text{ otherwise} . \end{cases} \tag{4}$$

From (3) and (4) we have then $\{k_{i,j}\}_2 = \mathsf{h}_2^{(x)}(K_2)$ and $\{k_{a',b'}\}_2 = \mathsf{h}_2^{(y)}(K_2)$ with $y < x$. Since h_2 is non-invertible $\Rightarrow NCRS_{\mathcal{C}} = \emptyset$. □

A.3 Proof of Proposition 3

Collusion set is $\mathcal{C} = (k_{i,j}, k_{u,v})$. $k_{i,j}^1$ and $k_{u,v}^1$ reveal first part of the keys in $\mathcal{R}_{a,b}(\mathcal{C})$. Let $k_{a,b}^2$ s.t. $(a, b) \in \{1, \ldots, n\} \times \{1, \ldots, t\}$ that can be computed from $k_{i,j}^2 = \mathsf{h}_2^{(2^{n-i}+2^{t-j}-2)}(K_2)$. We have then

$$2^{n-i} + 2^{t-j} \leq 2^{n-a} + 2^{t-b}. \tag{5}$$

Let suppose that $n - i \leq t - j$, therefore three cases are possible:

- if $\max(n - a, t - b) > t - j$, (5) has always a solution (because $\sum_{i=0}^{k} 2^i < 2^{k+1}$),
- if $\max(n - a, t - b) = t - j$, (5) has a solution only if $\min(n - a, t - b) \geq n - i$,
- if $\max(n - a, t - b) < t - j$, (5) has no solution.

The solution set verifies then

$$\max(n-a,t-b) > \max(n-i,t-j)$$
$$or$$
$$\big(\max(n-a,t-b) = \max(n-i,t-j) \ \mathit{and}\ \min(n-a,t-b) \geq \min(n-i,t-j)\big)\ .$$

The keys that can be computed from k_{ij}^2 and k_{uv}^2 are associated to the segments in $\mathcal{W}_{a,b}$ where

$$\mathcal{W}_{a,b} = \Big\{\mathsf{S}_{a,b} \notin \mathcal{C}\Big|\big(\max(n-a,t-b) > \max(n-i,t-j)\big)\ or$$
$$\big(\max(n-a,t-b) = \max(n-i,t-j)\ \mathit{and}\ \min(n-a,t-b) \geq \min(n-i,t-j)\big)\Big\}\ ,$$

and non collusion-resistant set $NCRS_{\mathcal{C}} = \mathcal{R}_{a,b} \cap \mathcal{W}_{a,b}$. We now give the bound $NCRS_{\mathcal{C}}$ in the worst case.

- if $n \geq t$, $\exists\ j \in \{1, t-1\}$ s.t. $NCRS_{\mathcal{C}}$ has maximum size for $\mathcal{C}' = (\mathsf{S}_{n,j}, \mathsf{S}_{1,t})$,
- if $n < t$, $\exists\ i \in \{1, n-1\}$ s.t. $NCRS_{\mathcal{C}}$ has maximum size for $\mathcal{C}' = (\mathsf{S}_{n,1}, \mathsf{S}_{i,t})$,

$$\Rightarrow |NCRS_{\mathcal{C}'}| \leq \begin{cases} \max\limits_{j=1}^{t-1}\big((n-t+j-1)\cdot(t-j)\big), & \text{if } t = \min(n,t), \\ \max\limits_{i=1}^{n-1}\big((t-n+i-1)\cdot(n-i)\big), & \text{if } n = \min(n,t). \end{cases}$$

Let $n = \max(n,t)$, then $\max\limits_{j=1}^{t-1}\big((n-t+j-1)\cdot(t-j)\big) = k\cdot(k-1) \leq k^2$ if n is even (i.e. $n = 2k$); or $\max\limits_{j=1}^{t-1}\big((n-t+j-1)\cdot(t-j)\big) = k^2$ if n is odd (i.e. $n = 2k+1$). This implies $\max\limits_{j=1}^{t-1}\big((n-t+j-1)\cdot(t-j)\big) \leq k^2 = \lfloor\frac{n}{2}\rfloor^2$. The same result holds for t when $t = \max(n,t) \Rightarrow \forall\mathcal{C}, |NCRS_{\mathcal{C}}| \leq \max(\lfloor\frac{n}{2}\rfloor^2, \lfloor\frac{t}{2}\rfloor^2)$. □

TWISTER – A Framework for Secure and Fast Hash Functions

Ewan Fleischmann[1], Christian Forler[2], Michael Gorski[1], and Stefan Lucks[1]

[1] Bauhaus-University Weimar, Germany
{Ewan.Fleischmann,Michael.Gorski,Stefan.Lucks}@uni-weimar.de
[2] Sirrix AG Security Technologies
c.forler@sirrix.com

Abstract. In this paper we present TWISTER, a new framework for hash functions. TWISTER incorporates the ideas of wide pipe and sponge functions. The core of this framework is a – very easy to analyze – `Mini-Round` providing both extremely fast diffusion as well as collision-freeness for one `Mini-Round`. The total security level is claimed to be not below $2^{n/2}$ for collision attacks and 2^n for 2nd pre-image attacks. TWISTER instantiations are secure against all known generic attacks. We also propose three instances TWISTER-n for hash output sizes $n = 224, 256, 384, 512$. These instantiations are highly optimized for 64-bit architectures and run very fast in hardware and software, e.g TWISTER-256 is faster than SHA2-256 on 64-bit platforms and TWISTER-512 is faster than SHA2-512 on 32-bit platforms. Furthermore, TWISTER scales very well on low-end platforms.

Keywords: Hash Function, Sponge Function, AES, HAIFA, Wide pipe, Randomized Hashing.

1 Introduction

One of the most used primitives in modern cryptography are hash functions. A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ takes an input of arbitrary size and computes an n-bit fingerprint out of it. Established security requirements for cryptographic hash functions are collision-, pre-image and 2nd pre-image resistance – but ideally, cryptographers expect a good hash function to *behave like a random function*. Nearly all iterative hash functions are designed using the MERKLE-DAMGÅRD construction [16, 31]. A MERKLE-DAMGÅRD hash function is an iterated hash function that uses a fixed length compression function $C : \{0,1\}^{n_c} \times \{0,1\}^m \rightarrow \{0,1\}^{n_c}$ where n_c is the size of the chaining value and m the size of a message block. We have $n = n_c$ for hash functions using the MERKLE-DAMGÅRD construction. By assuming a padded message $M = (M_1, \ldots, M_l)$, $|M_i| = m$, $1 \leq i \leq l$ and an internal chaining value $h_i \in \{0,1\}^{n_c}$ (h_0 is called the initial value) the computation of the hash value h_l for such a message M is as follows: $h_i = C(h_{i-1}, M_i)$, $i = 1, \ldots, l$.

The main benefit of the Merkle-Damgård transformation is that it preserves collision resistance: if the compression function C is collision resistant, then so is the hash function. Unfortunately, this result does not extend to pre- and 2nd preimage resistance. Recent results highlight some intrinsic limitations of the Merkle-Damgård approach. This includes being vulnerable to multi-collision attacks [22], long 2nd preimage attacks [24], and herding [23]. Even though the practical relevance of these attacks is unclear, they highlight some security issues which designers are well advised to avoid or take care of.

Related Work. Most popular hash functions such as MD5 [38], SHA-0 [34] or SHA-1 [33] have weaknesses in their design, leading to a huge amount of attacks [6, 7, 13, 18, 37, 40, 41, 42]. But also most new hash functions [2, 21, 25, 26] which try to take care for weaknesses in the Merkle-Damgård -construction itself were broken soon after their publications [29, 36, 35, 30, 20].

The concept of sponge functions [4] uses for example a big internal state that absorbs a message of arbitrary length and that later squeezes out a hash value of variable size. RadioGatún [3] with XOR sponges and Grindahl [26] with truncate-overwrite sponges are the first hash function that use this framework. Grindahl was shown to be vulnerable to several attacks [35, 20].

Our contribution. The design of secure and practical hash functions is of great interest since most hash functions have been broken. Due to the SHA-3 [32] competition many new proposals for hash function primitives will be published in the next months. In this paper we present a new hash function framework called Twister. Our proposal is based on a sponge construction [5] as well as on an wide pipe approach [27]. It also includes randomized hashing as proposed in the HAIFA framework [8]. The main goal of our approach is to present a fast, secure hash function which is flexible and simple to analyze. We can show that one cannot find a collision after one so called `Mini-Round` and that we obtain full diffusion after two application of a `Mini-Round`.

More precise, it uses XOR-sponges with a big internal state as proposed in [27]. The randomized hashing is built in via a salt value - a method proposed in the HAIFA framework [8]. In some sense we learn from the Grindahl design [26], but our approach is different in many ways. We take advantage of the well studied basic operations of AES [15] and adopt several of them, including some optimization techniques.

Due to recent breakdowns of many proposed hash functions we analyze the resistance of Twister against all known generic attacks on hash functions. We find, that the Twister framework resists all of them if the size of the internal chaining value is at least double the size of the hash output.

Outline. In Section 2 we present an informal description of the Twister hash function family. Then, in Section 3, we give an detailed description of the general Twister framework and we propose three instances: Twister-256, 384 and 512 for 256, 384 and 512 bit hash sizes. In Section 4 we discuss some security issues of TWISTER. Finally, we conclude our paper in Section 5. Some performance aspects of TWISTER are presented in Appendix A.

2 An Informal Description of the TWISTER Framework

In this section, we give a general description of the TWISTER hash function family and its building blocks. For a complete description of TWISTER, see the formal specifications in Section 3. TWISTER follows a very simple and clear design goal. It consists of an iterated compression function and of an output function.

2.1 The Compression Function

The compression function of TWISTER consists of building blocks called `Mini-Rounds` which are grouped into `Maxi-Rounds`. Each `Mini-Round` is a combination of well studied primitives, which are easy to analyze and fast to implement in software and hardware. The instances of the TWISTER hash function family differ only in their construction of a `Maxi-Round`. We will give an informal description of these principles below.

A `Mini-Round` consists of the following primitives:

– MessageInjection inserts a 64-bit message block into the last row of the state matrix,
– SubBytes applies a non-linear S-box table look up on each byte of the state matrix in parallel,
– AddTwistCounter XOR's a round dependent counter into the second column of the state matrix,
– ShiftRows rotates row i by $i-1$ positions to the left,
– MixColumns applies a linear diffusion on each column of the state matrix in parallel

A visualization of a mini round is shown in Appendix B.

A `Maxi-Round` contains between three and four `Mini- Rounds` and zero or one `blank rounds`. A `blank round` is a `Mini-Round` with no message input, which is equivalent to the all zero message block. Each `Maxi-Round` uses also a feed forward operation, i.e., the state before a `Maxi-Round` is feed forwarded with the state after a `Maxi-Round`. Figure 1 gives a high level description of a `Maxi-Round`.

2.2 The Output Function

The `Output-Round` of TWISTER contains a global feed forwards as well as some `Mini-Rounds` depending on the size of the hash output. First a `Mini-Round` is applied on the state H_{i-1}, then the resulting state is XOR'ed with H_{i-1} another `Mini-Rounds` is applied which gives the state H_i. Let H_f be the final state after the last compression function call. A 64-bit output stream out_i is then obtained

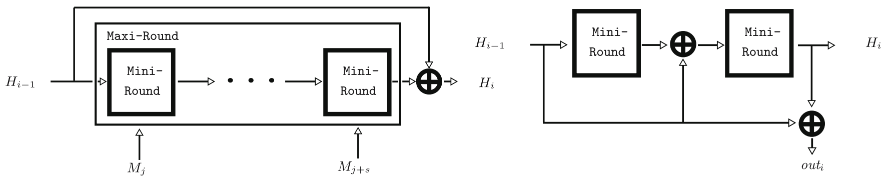


Fig. 1. Left: A `Maxi-Round` Right: An `Output-Round`

by XORing the first column of H_i with the first column of H_f. This procedure takes place until the needed amount of output bits are obtained. The last output stream can be varied between 32 bits and 64 bits by taking only the first half of out_i. This allows to vary the output size for a huge amount of applications. Figure 1 gives a high level description of a `Output-Round`.

3 Specification: The Twister Hash Function Family

In this section we present our hash function. We start off with a description of the general design strategy. The design is based on a block cipher that is iterated using the Davies-Meyer (DM) mode of operation [10]. Twister is a byte-oriented framework that operates on a square state matrix. The building block of the block cipher is called `Mini-Round`. It takes a sub portion of the message and processes it into the state $\mathcal{S}$ whereas $\mathcal{S}$ is a $N \times N$ byte-matrix, $N \in \mathbb{N}$. N should be at least of size 4 to obtain a valuable structure at all. After two `Mini-Rounds`, the state is guaranteed to have full diffusion. Also, two subsequent iterations of the `Mini-Round` is can be proved to be collision free. See Section 4 for a detailed discussion on our security issues.

After processing the padded message (i.e. the message is completely absorbed by the state $\mathcal{S}$), the output follows. This technique follows the design ideas of the sponge function [4] by not presenting the complete internal state to the attacker at once but slice by slice.

The following notations are used in the following:

$\mathcal{S} = (S_{i,j})_{1 \le i,j \le N}$	internal state matrix
$\mathcal{C} = (C_{i,j})_{1 \le i,j \le N}$	internal checksum matrix
N	number of rows and columns of the internal state matrix
msg_{size}	size of unpadded message (in bits)
R_{total}	number of total `Mini-Rounds` per compression function
R_{msg}	number of N-byte blocks processed per compression function
m	size of the padded message, measured in $N \cdot R_{msg}$ -byte blocks, i.e. the number of compression

	function calls needed to absorb the message into the state $\mathcal{S}$
$M = (M_1, \ldots, M_m)$	padded message to be handled by the TWISTER hash function
M_{unpad}	unpadded message
out	size of the hash value measured in N-byte blocks, i.e. $out = n/(8 \cdot N)$
n	size of the hash value in bits, i.e. $n = out \cdot 8 \cdot N$, where $n \leq (8 \cdot N)^2$
$H = (H_1, \ldots, H_{out})$	number of N-byte blocks of the hash output
ϕ	TwistCounter

3.1 TWISTER Components

This section describes in detail the TWISTER components.

The State *S*. TWISTER operates on a square state matrix $\mathcal{S} = (S_{i,j})$, $1 \leq i, j \leq N$, consisting out of N rows and columns, where each cell $S_{i,j}$ represents one byte.

$S_{1,1}$	$S_{1,2}$	...	$S_{1,N}$
$S_{2,1}$	$S_{2,2}$	...	$S_{2,N}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$S_{N,1}$	$S_{N,2}$	...	$S_{N,N}$

Notation: $S_{(i\rightarrow)} := (S_{i,1}, \ldots, S_{i,N})$ denotes the i-th row vector and $S_{(j\downarrow)} := (S_{1,j}, \ldots, S_{N,j})$ the j-th column vector.

Checksum *C*. The checksum enlarges the state of TWISTER-384 and TWISTER-512 to stick to our wide pipe design [27] decision. In other words: using the checksum we can double the internals state.

The checksum is as the state S a square checksum matrix $\mathcal{C} = (C_{i,j})$, $1 \leq i, j \leq N$, consisting out of N rows and columns, where each cell $C_{i,j}$ represents one byte.

$C_{1,1}$	$C_{1,2}$	...	$C_{1,N}$
$C_{2,1}$	$C_{2,2}$	...	$C_{2,N}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$C_{N,1}$	$C_{N,2}$	...	$C_{N,N}$

TwistCounter *ϕ*. The TwistCounter ϕ is a unsigned 64 bit integer that is added and decreased within a `Mini-Round` to prevent slide attacks.

3.2 The Compression Function

The Twister-hash function calls the compression function using the DM mode of operation. After the message is absorbed in the internal state the output function is called for every N bytes of output.

Recall that the compression function of Twister consists of building blocks called `Mini-Rounds` which are grouped into `Maxi-Rounds`. The instances of the Twister hash function family differ only in their construction of a `Maxi-Round`.

The compression function takes a 512-bit block and processes them into the internal state matrix $\mathcal{S}$. As a `Maxi-Round` only indicates the position of the local feed forward XOR-operation we will normally only discuss a compression function as a set of `Mini-Rounds`. The local feed-forward operation is an optional security feature and is discussed in Section 4. More general, the compression function works as follows. Let R be the number of `Mini-Rounds` in a compression function. (Note: In Figure 2 we have $R = 9$.)

Twister-*224 and* Twister-*256.* The Twister-224 and Twister-256 compression function consists of three `Maxi-Rounds`. Each `Maxi-Rounds` is followed by a feed-forward XOR-operation. The first and second `Maxi-Round` consist of three `Mini-Rounds`. The last `Maxi-Round` consists of two `Mini-Rounds` and one `blank round`. Figure 2 illustrates the compression function.

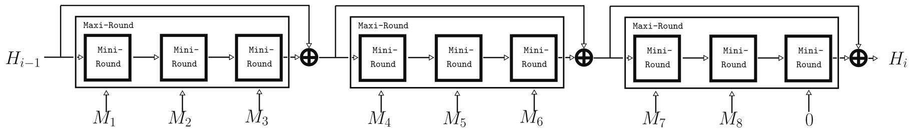


Fig. 2. The compression function of Twister-224 and Twister-256

Twister-*384 and* Twister-*512.* The Twister-384 and Twister-512 compression function consists of three `Maxi-Rounds`, too. The first `Maxi-Rounds` consists of three `Mini-Rounds`. The second `Maxi-Rounds` consists of a `Mini-Round` followed by a `blank round` and an other `Mini-Round`. The last `Maxi-Rounds` consists of three `Mini-Rounds` followed by a `blank round`. Figure 3 illustrates the compression function.

`Mini-Round`**.** The `Mini-Round` is the underlying building-block of any Twister hash function. It's main purpose is to inject the message (Message-Injection) and to take care for the diffusion of the state matrix $\mathcal{S}$. It is visualized in Figure 5. Twister can handle at most 2^{64} `Mini-Rounds`. This limitation causes by the `Add-TwistCounter` operation where a 64-bit counter is added. Each `Mini-Round` can process 64 bit of message data. Therefore, with a native usage of a `Mini-Round` it is possible to process up to $2^{64} \cdot 64$ message bits. If this limitation became in the future a real world issue it is possible to increase the size of the `TwistCounter` to 128 bit with almost no performance loss.

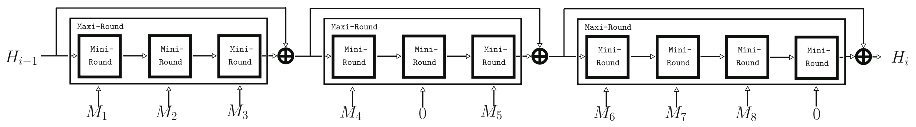


Fig. 3. The compression function of Twister-384 and Twister-512

Message injection. A 64 bit message block m is inserted (via XOR $\oplus$) into the last row. By using the notation $m = (m[1], \ldots, m[N])$ whereas the length of $m[N]$ is one byte, and

$$S_{(\to j)} \oplus m := (S_{1,j} \oplus m[1], \ldots, S_{N,j} \oplus m[N])$$

we define the message injection process by

$$S_{(\to 1)} = S_{(\to N)} \oplus m.$$

AddTwistCounter. The TwistCounter ϕ is a unsigned 64 bit integer. The initial state is the maximum value (0xFFFFFFFFFFFFFFFF). By using the notation $\phi = (\phi[1], \ldots, \phi[N])$ whereas the length of $\phi[N]$ is one byte, and $\phi[1]$ is the most significant byte of ϕ. The counter is added byte by byte - via the XOR operation - into the second column of the state $\mathcal{S}$.

$$S_{2,\downarrow} \oplus \phi := S_{2,1} \oplus \phi[1], \ldots, S_{2,N} \oplus \phi[N])$$

We define the TwistCounter addition by

$$S = S_{2,\downarrow} \oplus \phi$$

After the addition ϕ is decreased by one.

SubBytes. The function is defined as an bijection

$$SubBytes : \{0,1\}^8 \longrightarrow \{0,1\}^8$$

and is used as an S-box for each byte. It should, among other properties, be highly non-linear. A discussion on how to obtain such cryptographically strong S-boxes (for 8x8 S-boxes) can be found in [43]. Twister uses the well known and studied AES S-box. It can be found in [15].

We define the SubBytes operation by

$$S_{i,j} = SB(S_{i,j}) \quad \forall i, j.$$

ShiftRows. ShiftRows is a cyclic left shift similar to the ShiftRows operation of AES. It rotates row j by $(j-1) \bmod N$ bytes to the left.

We define the ShiftRows operation by

$$S_{(i,j-1)} := S_{(i,j)} \quad \forall i, j.$$

MixColumns. The MixColumn step is a permutation operation on the state. It applies a $N \times N$-MDS A (a maximum distance separable matrix as defined below) to each column, i.e. performs the operation $A \cdot S_{(j\downarrow)}$.

Definition 1. *An [n,k,d] code with generator matrix*

$$G = [I_{k\times k}\ A_{k\times(n-k)}]$$

is an MDS code if every square submatrix of A is non singular. The matrix A is called a MDS-matrix.

Our chosen MDS matrix is cyclic, i.e., its i-th row can be obtained by a cyclic right rotation of (02 01 01 05 07 08 06 01) by i entries. It has a branch number of 9 meaning that if two 8 byte input vectors differ in $1 \le k \le 8$ bytes, the output of MixColumns differs in at least $9 - k$ bytes. The 8×8-MDS matrix used for all proposed instances of TWISTER is:

$$MDS = \begin{pmatrix} 02\ 01\ 01\ 05\ 07\ 08\ 06\ 01 \\ 01\ 02\ 01\ 01\ 05\ 07\ 08\ 06 \\ 06\ 01\ 02\ 01\ 01\ 05\ 07\ 08 \\ 08\ 06\ 01\ 02\ 01\ 01\ 05\ 07 \\ 07\ 08\ 06\ 01\ 02\ 01\ 01\ 05 \\ 05\ 07\ 08\ 06\ 01\ 02\ 01\ 01 \\ 01\ 05\ 07\ 08\ 06\ 01\ 02\ 01 \\ 01\ 01\ 05\ 07\ 08\ 06\ 01\ 02 \end{pmatrix}$$

All of the byte entries are considered to be elements of $\mathbb{F}_{2^8}$. An element $\sum_{i=0}^{7} a_i x^i \in \mathbb{F}_{2^8}$ is represented by $\sum_{i=0}^{7} a_i 2^i$. The reduction polynomial $m(x)$ of $\mathbb{F}_{2^8}$ is defined as

$$m(x) = x^8 + x^6 + x^3 + x^2 + 1. \tag{1}$$

Properties of MDS matrices/codes can be found e.g. in [28]. A discussion on how to obtain suitable MDS matrices can be found in the full version of the paper.

Maxi-Round. A `Maxi-Round` contains of several `Mini-Rounds`, `blank round` and a optional checksum updates. We defined a checksum update operation as

$$C_{(i,\downarrow)} = C_{(i,\downarrow)} \oplus C_{(i+1,\downarrow)} \boxplus S_{(i,\downarrow)}$$

`Maxi-Rounds` use also a feed forward operation where the state before a `Maxi-Round` S^k is feed forwarded with the state after a `Maxi-Round` S^{k+1}. We defined the feed forward operation as

$$S_{(i,j)} := S^k_{(i,j)} \oplus S^{k+1}_{(i,j)} \quad \forall i, j.$$

Figure 1 illustrates a `Maxi-Round`.

3.3 Postprocessing

This section describes the TWISTER finalization process.

Postprocessing shall take place after the message processed by the compression function. This post processing consists of two steps:

1. Padding the message M
2. Message digest computation

Padding. The message, M, shall be padded before hash computation begins. For the padding, the well known 10-padding rule is applied (see e.g. [33])

Message digest computation. The `Output-Round` computes the message digest. It contains a global feed forwards as well as some `Mini-Rounds` depending on the size of the hash output. For every 64 message digest bits `Mini-Round` is applied on the state S, then the resulting state is XOR'ed with S^{k-1} another `Mini-Rounds` is applied which gives the state S^k. Let S^f be the final state after the last compression function call. A 64-bit output stream out_i is then obtained by XORing the first column of S^k with the first column of S^{k-1}. This procedure takes place until the needed amount of message digest bits are obtained. The last output stream can be varied between 32 bits and 64 bits by taking only the first half of out_i. This allows to vary the output size for a huge amount of applications. Figure 1 illustrates the `Output-Round`.

4 Security

In this section we analyze the security of TWISTER and show that it is resistant to all known generic attacks.

4.1 Generic Attacks

Length-Extension Attacks. Given an MERKLE-DAMGÅRD based hash function H. If one can find a collision for two messages M, M' with $M \neq M'$, such that $H(M)$ and $H(M')$ collide, then one can apply a length extension attack. For any message N one can easily produce a collision for $M||N$ and $M'||N$ as $H(M||N) = H(M'||N)$. Our padding rule avoids such type of attacks since we concatenate the length of the message to the message itself. Another attack can be as follows. For a known hash value $H(M)$ one can compute the hash value $H(M||X||N)$ for any suffix N, if the length of an unknown message M is known as well as the padding X of M. We prevent TWISTER to this kind of attack by two countermeasures: (i) By knowing only the hash value an attacker can not easily determine the state $\mathcal{S}$ after the last compression function call as he has only access to the result of the `final()` function. This function squeezes out some bits of the state applying output transformation and squeezes out some bits again. The bits of a squeezing process do not leave enough information to recover the internal state. (ii) The multiple feed-forward does also prevent any attacker to successfully gain any knowledge of prior state information. In each squeezing process the one feed forward takes place.

Multi-Collision Attacks. Joux [22] found that when iterative hash functions are used, finding a set of 2^k messages all colliding on the same hash value (a 2^k-multi-collision) is as easy as finding k single collision for the hash function. Finding a collision in the compression function, i.e., a single block collision one can find k of such collisions each starting from the chaining value produced by the previous one-block collision. In other words, one have to find two messages blocks M_i and $M_i' \neq M_i$ with $C(h_{i-1}, M_i) = C(h_{i-1}, M_i')$, where $C(\cdot)$ represents the compression function and h_i the chaining value. Then it is possible to construct 2^k messages with the same hash value by choosing for block i either the message block M_i or M_i'. Joux also showed that the concatenation of two different hash functions is not more secure against collision attacks than the strongest one. This attack can find 2^k-way internal multi-collisions with a complexity of $k \cdot 2^{n_c/2}$. An instance of Twister fully resists the multi-collision attack if $8 \cdot N^2 \geq 2n_c$, since the complexity is determined by $k \cdot 2^{(8 \cdot N^2)/2}$. All instances of Twister have this feature, although the state of Twister-384 and Twister-512 is not big enough to prevent this attack alone, including the check sum can be viewed as an enlarging of the state, which then guarantees to prevent for this attack.

Herding Attacks. The herding attack [23] works as follows. An attacker takes 2^k chaining values which are fixed or randomly chosen. Then he chooses $O(2^{n_c/2-k/2})$ message blocks. He computes the output of the compression function for each chaining value and each block. It is expected that for each chaining value there exists another chaining value, such that both collide to the same value. The attacker stores the message block that leads to such a collision in a table and repeats this process again with the newly found chaining values. Once the attacker has only one chaining value, it is used to compute the hash value to be published. To find a message whose chaining value is among the 2^k original values, the attacker has to perform $O(2^{n_c-k})$ operations. For such a message the attacker can retrieve from the stored messages the message blocks that would lead to the desired hash value. The time complexity of this attack is about $O(2^{n_c/2+k/2})$ operations for the first step and $O(2^{n_c-k})$ operations for the second step. The whole attack on an n-bit hash function requires approximately $2^{(2n_c-5)/3}$ work. For Twister we have $n_c = 8 \cdot N^2$ and with $8 \cdot N^2 \geq (3n_c+5)/2$ the attack has the same complexity as for for a (second) pre-image attack on a random oracle. The complexity of this attack decreases with increased size of the message. If the message is of size about 2^t, then the complexity of the attack is $2^{(2n_c-5)/3-t}$. One has to choose N such that the hash function is protected against this kind of attack for a given upper bound. All of our proposed instances of Twister resist this kind of attack.

Long 2nd pre-image Attacks. Dean [17] found that fix-points in the compression function can be used for a second-pre-image attack against long messages in time $O(n_c \cdot 2^{n_c/2})$ and memory $O(n_c \cdot 2^{n_c/2})$, where $n_c = 8 \cdot N^2$ is the size of the internal state (which is equal to the size of the hash output for a plain MERKLE-DAMGÅRD constructions). Kelsey and Schneier [24] extend this result and provide an attack to find a 2nd pre-image on a MERKLE-DAMGÅRD construction with MERKLE-DAMGÅRD strengthening much faster than the expected workload of 2^{n_c}. The complexity of the attack is determined by the complexity of finding expandable messages. These are messages of varying sizes such that all these messages collide internally for a given IV. Expandable message can either be found using internal collisions or fixed points between a one-block message and an α-block message for varying values of α. The complexity of the generic attack to find a 2nd pre-image for a 2^k-message block is about $k \cdot 2^{n_c/2+1} + 2^{n_c-k+1}$ compression function calls.

Long 2nd preimage attacks in this form cannot be applied to the TWISTER framework for three reasons. First, we include the twist counter which does not allow to find expandable messages. Second, we make use of multiple feed-forwards and, third, the internal chaining value is in general much larger than n. This make it harder to find collisions and fix points since we essentially have a constructions similar to the wide pipe design [27].

Andreeva et al. [1] have shown that a combination of the attacks from [17, 24, 23] can be mounted to dithered hash functions, which gives the attacker more control on the second-pre-image, since he can choose about the first half of the message in an arbitrary way. This attack can be done in time $2^{n_c/2+k/2+2} + 2^{n_c-k}$. Although it is more expensive than the attack of Kelsey and Schneier [24], it works even when an additional input to the compression function (dithering) is given. One have to make the dithering as huge as possible, such that there are no small cycles. TWISTER includes the twist counter ϕ which is very large, i.e., the twist counter is of size of the maximal message length. The larger this counter is the longer cycles we have, which increases the protection against this type of attack.

Slide Attacks. Slide attacks are common in block cipher cryptanalysis, but they also applicable to hash functions. Given a hash function H and two messages M and M' where M is a prefix of M', one can find a slid pair of messages (M, M') such that the the last message input block of the longer message M' performs only an additional blank round, e.g. for sponge constructions. These two messages are then slid by one blank round. This attack allows to recover the internal state of a slid pair of messages an even backward computation as shown in [20]. The twist counter ϕ avoids the possibility of slide attacks, since XOR-ing a different value in each `Mini-Round` into the state matrix does not allow to find slid pairs of messages. Furthermore, the last inserted message block cannot be the all zero block due to the padding rules. Thus slide attacks are unavailable for TWISTER.

Differential Attacks. The essential idea of differential attack on hash functions [12], as used to break MD5 and SHA-0/1, is to exploit a high-probability input/output differential over some component of the hash function, e.g. under the form of a "perturb-and-correct" strategy for the latter functions, exploiting high probability linear/non-linear characteristics. In the design of the Twister framework, we applied the following countermeasures against differential attacks:

– *High-Speed-Diffusion.* The strong diffusion capabilities of the `Mini-Round` in combination with the non-linear S-box make the exploit of linear approximations highly implausible. As it is impossible to find a collision after one `Mini-Round`, any attacker has to trail presumably very long paths to be able to find a collision.
– *Nested feed-forward.* The internal feed-forward operations aim at strengthening the function against differential paths.
– *Optional internal wide pipe.* this makes internal collision unlikely, and the output-rounds make the differences much harder to predict in the hash value.
– Using different operators (e.g. $\boxplus$ and $\oplus$) highly complicates the computation of good differential paths.

4.2 Security Proofs for Twister

Twister was designed such that a single `Mini-Round` is proven to be collision free. This is expressed by the following lemma.

Lemma 1. *From any state h_{i-1} one cannot find input message blocks M, $M' \neq M$ such that*

$$\texttt{Mini-Round}(h_{i-1}, M) = \texttt{Mini-Round}(h_{i-1}, M')$$

for all M, $M' \neq M$.

Proof. Assume that h_i^M is the state after inserting the message block M and $h_i^{M'}$ is the state after inserting M'. Then, if M and M' are different in byte j the states h_i^M and $h_i^{M'}$ are different in column j in at least $9 - k$ bytes. This is due to the MDS property of our diffusion layer, which has a branch number of 9. ■

We can also show that Twister offers full diffusion after two input blocks.

Lemma 2. *Given an internal state h_{i-1} and two input blocks M_1 and M_1' where $M_1 \neq M_1'$. Then, we have full diffusion of the state after two `Mini-Rounds`.*

Proof. The message distribution process is visualized in Figure 4. Two messages M_1 and $M_1' \neq M_1$ are different in at least one byte. Due to the diffusion of MixColumns at least one state column differs in 8 bytes. The ShiftRows of the following `Mini-Round` with no message input will distribute the all difference column into a one byte difference in each state column. After that MixColumns generates a difference in all state bytes. Which leads to full diffusion. ■

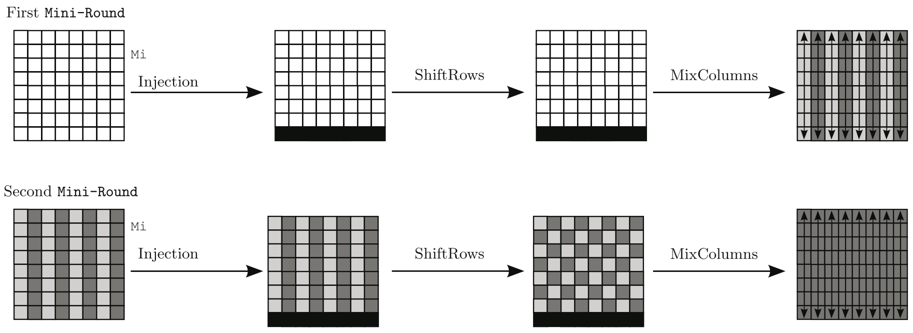


Fig. 4. Visualization of the diffusion of a message after two `Mini-Rounds`

5 Conclusion

In this paper we proposed a family of hash functions which overcome several identified weaknesses of the commonly used MDx family of hash functions (MD4, MD5, SHA-X). The weaknesses are addressed by several methods. By using some of the well analyzed building blocks and ideas of Rijndael we obtain a design for which we claim that no efficient differential collision structures exist. In addition, we limit access to the internal structure and take care that any possible difference quickly diffuses into the internal state. Furthermore it is highly scalable as there are – as proposed in e.g. in TWISTER-256 and TWISTER-512 – many possible ways to adopt our main building block, the `Mini-Round`.

We proposed three instantiations of the TWISTER framework, TWISTER-n, where $n = 256, 384$, 512. The claimed security level for TWISTER-256 with respect to collision resistance is 2^{128} and with respect to (2nd) pre-image resistance 2^{256}. For TWISTER-384, we claim a collision resistance of 2^{192} and a (2nd) pre-image resistance of 2^{384}. For TWISTER-512, the claimed security level for collision resistance is 2^{256} and for (2nd) pre-image resistance 2^{512}. The TWISTER family of hash functions exploits mathematical structures (i.e. MDS matrices) and, at the same time, having comparable speed as the SHA-2 family. Thus, instances of the TWISTER framework are suitable for a huge range of applications from low-end 8-bit platforms up to high-end 64-bit architectures.

TWISTER is submitted to the NIST SHA-3 competition. A new slightly modified version of TWISTER as well as the full submission document can be found on www.twister-hash.com.

References

[1] Andreeva, E., Bouillaguet, C., Fouque, P.-A., Hoch, J.J., Kelsey, J., Shamir, A., Zimmer, S.: Second Preimage Attacks on Dithered Hash Functions. In: Smart [39], pp. 270–288 (2008)

[2] Aumasson, J.-P., Meier, W., Phan, R.C.-W.: The Hash Function Family LAKE. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 36–53. Springer, Heidelberg (2008)
[3] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Radiogatun, a belt-and-mill hash function. Presented at Second Cryptographic Hash Workshop, Santa Barbara (August 24-25, 2006) (2006), http://radiogatun.noekeon.org/
[4] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge Functions. Ecrypt Hash Workshop (2007), http://gva.noekeon.org/papers/bdpv07.html
[5] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart [39], pp. 181–197 (2008)
[6] Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin [19], pp. 290–305 (2004)
[7] Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and Reduced SHA-1. In: Cramer [14], pp. 36–57 (2005)
[8] Biham, E., Dunkelman, O.: A Framework for Iterative Hash Functions - HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007)
[9] Biryukov, A. (ed.): FSE 2007. LNCS, vol. 4593. Springer, Heidelberg (2007)
[10] Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
[11] Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
[12] De Cannière, C., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
[13] Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
[14] Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
[15] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
[16] Damgård, I.: A Design Principle for Hash Functions. In: Brassard [11], pp. 416–427 (1989)
[17] Deam, R.D.: Formal Aspects of Mobile Code Security. Ph.D. dissertation, Princeton University (1999)
[18] Dobbertin, H.: Cryptanalysis of MD4. J. Cryptology 11(4), 253–271 (1998)
[19] Franklin, M. K. (ed.): CRYPTO 2004. LNCS, vol. 3152. Springer, Heidelberg (2004)
[20] Gorski, M., Lucks, S., Peyrin, T.: Slide Attacks on Hash Functions. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 143–160. Springer, Heidelberg (2008)
[21] Hong, D., Chang, D., Sung, J., Lee, S.-J., Hong, S.H., Lee, J.S., Moon, D., Chee, S.: A New Dedicated 256-Bit Hash Function: FORK-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 195–209. Springer, Heidelberg (2006)
[22] Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin [19], pp. 306–316 (2004)
[23] Kelsey, J., Kohno, T.: Herding Hash Functions and the Nostradamus Attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
[24] Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer [14], pp. 474–490 (2005)

[25] Knudsen, L.R.: SMASH - A Cryptographic Hash Function. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 228–242. Springer, Heidelberg (2005)

[26] Knudsen, L.R., Rechberger, C., Thomsen, S.S.: The Grindahl Hash Functions. In: Biryukov [9], pp. 39–57 (2007)

[27] 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)

[28] MacWilliams, F.I., Sloane, N.J.A.: The Theory of Error-Correcting Codes (1977)

[29] Matusiewicz, K., Peyrin, T., Billet, O., Contini, S., Pieprzyk, J.: Cryptanalysis of FORK-256. In: Biryukov [9], pp. 19–38 (2007)

[30] Mendel, F., Schläffer, M.: Collisions for Round-Reduced LAKE. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 267–281. Springer, Heidelberg (2008)

[31] Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [11], pp. 428–446 (1989)

[32] National Institute of Standards and Technology. Cryptographic Hash Project, http://csrc.nist.gov/groups/ST/hash/index.html

[33] National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard (April 1995), http://csrc.nist.gov

[34] National Institute of Standards and Technology. FIPS 180: Secure Hash Standard (1993), http://csrc.nist.gov

[35] Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 551–567. Springer, Heidelberg (2007)

[36] Pramstaller, N., Rechberger, C., Rijmen, V.: Breaking a New Hash Function Design Strategy Called SMASH. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 233–244. Springer, Heidelberg (2006)

[37] Rijmen, V., Oswald, E.: Update on SHA-1. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 58–71. Springer, Heidelberg (2005)

[38] Rivest, R.: The MD5 Message-Digest Algorithm (1992)

[39] Smart, N.P. (ed.): EUROCRYPT 2008. LNCS, vol. 4965. Springer, Heidelberg (2008)

[40] Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer [14], pp. 1–18 (2005)

[41] Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

[42] Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer [14], pp. 19–35 (2005)

[43] Yi, X., Cheng, S.X., You, X.H., Lam, K.Y.: A Method for Obtaining Cryptographically Strong 8x8 S-boxes. In: IEEE Global Telecommunications Conference, GLOBECOM 1997, vol. 2, pp. 689–693 (1997)

A Performance

Twister was especially designed with 64-platforms in mind by making it possible to aggregate 8 times an 8-bit table lookup into one single 64-bit table lookup. The following performance measurements were conducted on:

Processor: Core2Duo $T7300$
Clock Speed: 2000 MHz
Memory: 2048 MB
Operating System: Linux, GNU Debian *Lenny*, Kernel 2.6.26-1 x64
Compiler: GCC 4.3, Optimization settings: -Os

For comparison, performance measurement results for SHA-2 on the this platform are given in Table 1.

Table 1. Performance comparison of Twister and SHA-2

		Output Size	
Algorithm	**Platform**	**224/256**	**384/512**
SHA-2	64	20.1	13.1
Twister	64	15.8	17.5
SHA-2	32	29.3	55.2
Twister	32	35.8	39.6
Twister	8	200	220

All value are measured in cycles per byte.

B Visualization of a `Mini-Round`

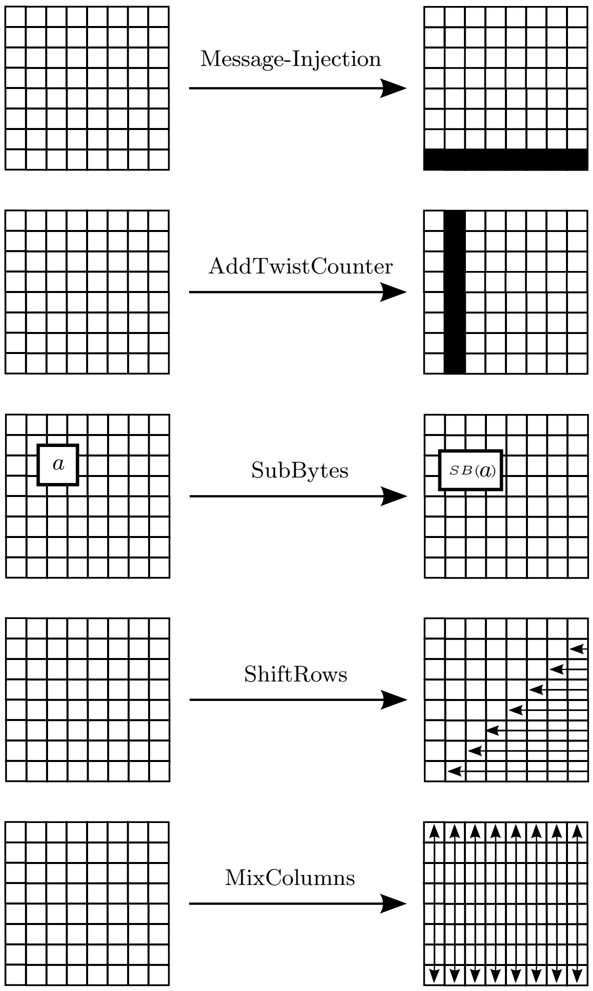


Fig. 5. A `Mini-Round`

Preimage Attack on Hash Function RIPEMD

Gaoli Wang[1,⋆] and Shaohui Wang[2]

[1] School of Computer Science and Technology, Donghua University,
Shanghai 201620, China
wanggaoli@dhu.edu.cn

[2] Nanjing University of Posts and Telecommunications,
Nanjing 210046, China
wangshaohui@njupt.edu.cn

Abstract. RIPEMD is a cryptographic hash function devised in the framework of the RIPE project (RACE Integrity Primitives Evaluation, 1988-1992). It consists of two parallel lines, and each line is identical to MD4 except for some internal constants. It has been broken by the collision attack, but no preimage attack was given. In this paper, we give a preimage attack on the compression function of the 26-step reduced RIPEMD with complexity 2^{110} compression function computations, and we extend the attack on the compression function to an attack on the 26-step reduced RIPEMD with complexity $2^{115.2}$ instead of 2^{128}. Then we extend the attack on 26 steps to the attack on 29 steps with the same complexity. Moreover, we can reduce the complexity of the preimage attack on the full RIPEMD without the padding rule by 1 bit compared with the brute-force attack.

Keywords: hash function, RIPEMD, cryptanalysis, preimage attack.

1 Introduction

Hash function is defined as a mapping $h : \{0,1\}^* \longrightarrow \{0,1\}^n$, where $\{0,1\}^*$ denotes the set of bit strings of arbitrary length, and $\{0,1\}^n$ denotes the set of strings of n-bit length.

From the security perspective, a hash function h with inputs x, $x^{'}$ and outputs y, $y^{'}$ should satisfy some or all of the properties. For the formal definition of them we refer to [1].

- Preimage Resistance: for any output y, it is computationally infeasible to get an input x such that $h(x) = y$.
- Second-preimage Resistance: for any input x, it is computationally infeasible to get another input $x^{'} (\neq x)$ such that $h(x) = h(x^{'})$.

⋆ Supported by Foundation of Donghua University No. 112-10-0044018.

– Collision Resistance: it is computationally infeasible to get any two distinct inputs x, $x^{'}$ such that $h(x) = h(x^{'})$.

Now most attacks on hash functions focus on finding collisions, and the most well-known generic attack to find collisions is the birthday attack. By a reasoning similar to the birthday paradox, there is a good chance to find a collision if we compute hash values of about $2^{\frac{n}{2}}$ messages, where the hash value is n-bit. While finding preimage is relatively harder, and the only generic attacks to find second preimage or preimage are the brute-force which both have the complexity of 2^n.

A hash function with n-bit hash value is considered academically broken if it is possible to find collisions, second preimage or preimage in less than $2^{\frac{n}{2}}$, 2^n or 2^n hash computations respectively.

There has been a great progress in finding the collision of hash functions in recent years. Wang et al. [3,4,5,6,7], Biham et al. [8] and Mendel et al. [9] found collisions of many hash functions based on MD4, such as HAVAL, MD5, RIPEMD, SHA-0 and SHA-1 etc.. These hash functions are considered not secure now. However, some of them are still widespread used when collision resistance is not required.

Preimage resistance is a weaker security notion than collision resistance. In some applications, collision resistance is unimportant, but preimage resistance is needed. The work on preimage attacks is much less than collision attacks in the cryptanalysis of hash functions. [10] showed that preimages of the first 2-round MD4 can be found, which was improved by [11]. They found the preimage of 2 rounds and 7 steps of MD4. [12] presented a preimage attack on the full MD4. [13] gave an example of the preimage attack against MD2. This work was improved in [14]. In [15], they presented preimage attacks on 3-pass HAVAL and 47-step MD5. [20] gave preimage attacks on 3, 4, and 5-pass HAVAL.

RIPEMD [2] was developed in the European RIPE project (RACE Integrity Primitives Evaluation, 1988-1992). Its compression function consists of two parallel lines of MD4 [17] compression function. In this paper, we present an inversion of the compression function of the 26-step reduced RIPEMD with the complexity of 2^{110} compression function computations. Then the preimage attack on the compression function is extended to a preimage attack on the 26-step reduced RIPEMD with complexity $2^{115.2}$ and memory requirement of 2^{23} bytes. The attack on 26-step reduced RIPEMD can be extended to 29 steps with the same complexity. Moreover, we present a preimage attack on the full RIPEMD without the padding rule with complexity 2^{127}, which optimizes the complexity order for brute-force attack.

The rest of the paper is organized as follows. Section 2 describes the RIPEMD algorithm. In Section 3, we show how to invert the compression function of the 26-step reduced RIPEMD and extend the preimage attack on the compression function to the preimage attack on hash function. Moreover, we extend the attack on 26-step reduced RIPEMD to 29-step reduced RIPEMD. Section 4 shows the preimage attack on the full RIPEMD without the padding rule by speeding up the brute force attack. Finally, we summarize the paper in Section 5.

2 Description of RIPEMD

The hash function RIPEMD compresses any arbitrary length message into a message with the length of 128 bits. Firstly, RIPEMD pads any given message into a message with the length of 512 bits multiple. For each 512-bit message block, RIPEMD compresses it into a 128-bit hash value by a compression function.

The initial value of RIPEMD is:

$$(a_0, b_0, c_0, d_0) = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476)$$

The compression function of RIPEMD consists of two parallel lines of MD4, which are denoted by Line1 operation and Line2 operation respectively. Each operation is identical to MD4 except for some internal constants. The nonlinear functions in each round are as follows:

$$\begin{aligned} F(X,Y,Z) &= (X \wedge Y) \vee (\neg X \wedge Z) \\ G(X,Y,Z) &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \\ H(X,Y,Z) &= X \oplus Y \oplus Z \end{aligned}$$

Here X, Y, Z are 32-bit words. The operations of the three functions are all bitwise. $\wedge$, $\oplus$ and $\vee$ are bitwise AND, XOR and OR respectively. $\neg$ represents the bitwise complement of X. Each round of the compression function is composed of 16-step operations.

In the following descriptions, $\lll s_i (i = 1, \cdots, 48)$ represents the circular shift s_i-bit positions to the left. + denotes addition modulo 2^{32}, - denotes subtract modulo 2^{32}.

Line 1 operation process. For a 512-bit block M, $M = (m_0, m_1, \ldots, m_{15})$, Line1 operation process is as follows:

1. Let (aa_0, bb_0, cc_0, dd_0) be the input of Line1 process for M. If M is the first block to be hashed, (aa_0, bb_0, cc_0, dd_0) is the initial value. Otherwise it is the output of the previous block compressing.
2. Perform the following 48 steps (three rounds):
   (a) For $i = 1, \cdots, 16$, do the following 16 operations:

$$\begin{aligned} aa_i &= dd_{i-1} \\ bb_i &= (aa_{i-1} + F(bb_{i-1}, cc_{i-1}, dd_{i-1}) + m_{\sigma(i)}) \lll s_i \\ cc_i &= bb_{i-1} \\ dd_i &= cc_{i-1} \end{aligned}$$

   (b) For $i = 17, \cdots, 32$, do the following 16 operations:

$$\begin{aligned} aa_i &= dd_{i-1} \\ bb_i &= (aa_{i-1} + G(bb_{i-1}, cc_{i-1}, dd_{i-1}) + m_{\sigma(i)} + 0x5a827999) \lll s_i \\ cc_i &= bb_{i-1} \\ dd_i &= cc_{i-1} \end{aligned}$$

(c) For $i = 33, \cdots, 48$, do the following 16 operations:

$$\begin{aligned} aa_i &= dd_{i-1} \\ bb_i &= (aa_{i-1} + H(bb_{i-1}, cc_{i-1}, dd_{i-1}) + m_{\sigma(i)} + 0x6ed9eba1) \lll s_i \\ cc_i &= bb_{i-1} \\ dd_i &= cc_{i-1} \end{aligned}$$

The orders of message words can be seen in Table 1. The details of the shift positions can be seen in Table 2.

Line 2 operation process. For a 512-bit block M, $M = (m_0, m_1, \ldots, m_{15})$, Line2 operation process is as follows:

1. Let $(aaa_0, bbb_0, ccc_0, ddd_0)$ be the input of Line2 process for M. If M is the first block to be hashed, $(aaa_0, bbb_0, ccc_0, ddd_0)$ is the initial value. Otherwise it is the output of the previous block compressing.
2. Perform the following 48 steps (three rounds) :
   (a) For $i = 1, \cdots, 16$, do the following 16 operations:

$$\begin{aligned} aaa_i &= ddd_{i-1} \\ bbb_i &= (aaa_{i-1} + F(bbb_{i-1}, ccc_{i-1}, ddd_{i-1}) + m_{\sigma(i)} + 0x50a28be6) \lll s_i \\ ccc_i &= bbb_{i-1} \\ ddd_i &= ccc_{i-1} \end{aligned}$$

   (b) For $i = 17, \cdots, 32$, do the following 16 operations:

$$\begin{aligned} aaa_i &= ddd_{i-1} \\ bbb_i &= (aaa_{i-1} + G(bbb_{i-1}, ccc_{i-1}, ddd_{i-1}) + m_{\sigma(i)}) \lll s_i \\ ccc_i &= bbb_{i-1} \\ ddd_i &= ccc_{i-1} \end{aligned}$$

   (c) For $i = 33, \cdots, 48$, do the following 16 operations:

$$\begin{aligned} aaa_i &= ddd_{i-1} \\ bbb_i &= (aaa_{i-1} + H(bbb_{i-1}, ccc_{i-1}, ddd_{i-1}) + m_{\sigma(i)} + 0x5c4dd124) \lll s_i \\ ccc_i &= bbb_{i-1} \\ ddd_i &= ccc_{i-1} \end{aligned}$$

The orders of message words can be seen in Table 1. The details of the shift positions can be seen in Table 2.

Add the output of Line1 to the output of Line2.

$$H_0 = b_0 + cc_{48} + ddd_{48}$$

$$H_1 = c_0 + dd_{48} + aaa_{48}$$

$$H_2 = d_0 + aa_{48} + bbb_{48}$$

$$H_3 = a_0 + bb_{48} + ccc_{48}$$

If M is the last message block of the message MM, then $H(MM) = H_0 * H_1 * H_2 * H_3$ is the hash value for the message MM, where $*$ denotes the bit concatenation. Otherwise take (H_0, H_1, H_2, H_3) as inputs, and repeat the compression process for the next 512-bit message block.

Table 1. Word Processing Orders

$round_1$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$round_2$	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
$round_3$	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

Table 2. Shift positions

$round_1$	3	7	11	19	3	7	11	19	3	7	11	19	3	7	11	19
$round_2$	3	5	9	13	3	5	9	13	3	5	9	13	3	5	9	13
$round_3$	3	9	11	15	3	9	11	15	3	9	11	15	3	9	11	15

3 Preimage Attack on the 26-Step Compression Function

In this section, first, we will recall some nice properties of the Boolean function F, then, we will describe our preimage attack on the compression function of the 26-step reduced RIPEMD, which is denoted by CRIPEMD. We can invert CRIPEMD by using the strategy of [12,15] and exploiting the nice properties of Boolean function F and the order of the message words. A preimage attack against CRIPEMD is given with the complexity of 2^{110} CRIPEMD computations. Then we extend the preimage attack on CRIPEMD to an preimage attack on the 26-step reduced RIPEMD with complexity $2^{115.2}$.

3.1 Some Basic Propositions

In this section we will recall some popular used properties of the nonlinear function F in our attack. These properties were presented in [10,16,3] etc.. We can use these key properties to absorb some difference.

Proposition. For the nonlinear function $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$, the following properties hold:

1. $F(x, y, z) = F(x, \neg y, z)$ if and only if $x = 0$.

2. $F(x, y, z) = F(x, y, \neg z)$ if and only if $x =$ `0xffffffff`.

3.2 Preimage Attack on CRIPEMD

The key tricks used in our attack are the absorption of changes in c_0 and the exploitation of the order of the message words. We will give the following observations.

Observation 1. From the algorithm of RIPEMD, it is easy to get the following conclusions about Line1 operation and Line2 operation:

1. At each step i $(1 \leq i \leq 48)$, only the chaining variable b_i is renewed, and $b_i = c_{i+1} = d_{i+2} = a_{i+3}$ $(0 \leq i \leq 45)$, $b_{46} = c_{47} = d_{48}$, $b_{47} = c_{48}$.
2. At each step i $(1 \leq i \leq 48)$, from the chaining variables a_i, b_i, c_i, d_i and message word $m_{\sigma(i)}$, we can compute a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1}.

Observation 2. From the order of the message words in Table 1, we know that m_2 is used at Step 3 (the very beginning of CRIPEMD) and Step 25 (the very end of CRIPEMD). Therefore, we can modify m_2 to alter cc_{26} (and don't alter ddd_{26}), so that $H_0 = b_0 + cc_{26} + ddd_{26}$ is equal to H_0^*. Then according to the Proposition, we know that F can "absorb" the change in c_0 at Step 1 and Step 2 if the conditions $b_0 = 0$ and $b_1 =$ `0xffffffff` are added respectively.

The general procedure of the attack is as follows. Firstly, we choose chaining variables (a_0, b_0, c_0, d_0) with certain constraints. Secondly, we choose a 512-bit message block $M = (m_0, \cdots, m_{15})$ with certain constraints. Thirdly, we modify the message word m_2 such that cc_{26} is changed (and ddd_{26} isn't changed) to ensure that $H_0 = b_0 + cc_{26} + ddd_{26}$ is equal to H_0^*. Fourthly, we correct c_0 such that the change in m_2 doesn't change b_1, b_2, b_3, b_4, so doesn't change the subsequent chaining variables.

Description of the Attack. Suppose we want to get the output of CRIPEMD $(H_0^*, H_1^*, H_2^*, H_3^*)$, our goal is to find a 512-bit message block M and input chaining variables (a_0, b_0, c_0, d_0) such that the compressing output is equal to $(H_0^*, H_1^*, H_2^*, H_3^*)$, i. e. CRIPEMD$((a_0, b_0, c_0, d_0), M) = (H_0^*, H_1^*, H_2^*, H_3^*)$.

We will describe the attack in Algorithm 1. The algorithm first sets $b_0 = 0$ and $bbb_1 =$ `0xffffffff` to ensure that a change in c_0 doesn't affect bbb_1 and bbb_2 of Line2 operation. If $bbb_1 =$ `0xffffffff` holds, then the Hamming weight (i. e. the number of "1" bits in the binary sequence) of bb_1 in Line1 operation is 18, which will be proved below. Therefore, a change in c_0 doesn't affect bb_2 of Line1 operation with the probability of 2^{-14}. Obviously, the condition $b_0 = 0$ ensure that a change in c_0 doesn't affect bb_1. Then the algorithm will modify m_2 to ensure that H_0 meets the requirement. Finally, c_0 is corrected to ensure that the change of m_2 doesn't affect bb_3 and bbb_3. So the modification of c_0 is absorbed with probability 2^{-14}, i. e. the probability of the forward stage unchanged with the new m_2 is 2^{-14}. If the forward stage unchange, then the 32-bit of the 128-bit image is satisfied, and we can get the compressing values $(H_0, H_1, H_2, H_3) = (H_0^*, H_1^*, H_2^*, H_3^*)$ by exhaustively search the remaining 96 bits values.

```
Algorithm 1 Preimage Attack on CRIPEMD
Input: $H_0^*$, $b_0^* = 0$.
Output: $M = (m_0, \cdots, m_{15})$ and $(a_0, b_0, c_0, d_0)$ st. $(H_0, H_1, H_2, H_3) = (H_0^*, H_1^*, H_2^*, H_3^*)$.
1. repeat
2.     Pick an initial variable with $b_0 = 0$ and $a_0$, $c_0$, $d_0$ being arbitrary values.
3.     Choose $m_0$ such that $bbb_1 = \texttt{0xffffffff}$.
4.     Choose arbitrary values for $m_1, \cdots, m_{15}$.
5.     Do the Line1 operation to get $aa_{26}$, $bb_{26}$, $cc_{26}$, $dd_{26}$.
6.     Do the Line2 operation to get $aaa_{26}$, $bbb_{26}$, $ccc_{26}$, $ddd_{26}$.
7.     Modify $m_2$ to change $cc_{26}$, so that $cc_{26} = H_0^* - b_0 - ddd_{26}$.
8.     Correct $c_0$ to keep $bb_3$ and $bbb_3$ unchanged.
9.     Compute the final compressing value $(H_0, H_1, H_2, H_3)$.
10.    If $(H_0, H_1, H_2, H_3) = (H_0^*, H_1^*, H_2^*, H_3^*)$, then return $(a_0, b_0, c_0, d_0)$ and $M = (m_0, \cdots, m_{15})$.
```

Correctness of the Attack

1. In order to make Line 3 of Algorithm 1 feasible, we modify m_0 as follows:

$$m_0 \leftarrow (\texttt{0xffffffff} >>> 3) - a_0 - F(b_0, c_0, d_0) - 0x50a28be6$$

2. We will show that if $bbb_1 = \texttt{0xffffffff}$, then the Hamming weight of bb_1 in Line1 operation is 18. From the algorithm of RIPEMD, we can get the equations (1) and (2), combined with the equation (3), we know that $bb_1 = (\texttt{0xffffffff} - 0x50a28be6) \lll 3 = 0xaf5d7419 \lll 3 = 0x7aeba0cd$. Therefore, the Hamming weight of bb_1 is 18.

$$bb_1 = (a_0 + F(b_0, c_0, d_0) + m_0) \lll 3 \quad (1)$$
$$bbb_1 = (a_0 + F(b_0, c_0, d_0) + m_0 + 0x50a28be6) \lll 3 \quad (2)$$
$$bbb_1 = \texttt{0xffffffff} \quad (3)$$

3. For Line2 operation, according to Proposition 1, we know that the condition $b_0 = 0$ ensures the change of c_0 results in no change in bbb_1, and according to Proposition 2, the condition $bbb_1 = \texttt{0xffffffff}$ ensures that the change of c_0 results in no change in bbb_2. Therefore, we can modify c_0 to correct a change in m_2, and in the same time not alter the chaining variables b_0 ,bbb_1, bbb_2, bbb_3, so not alter the subsequent chaining variables.
4. For Line1 operation, according to Proposition 1, we know that the condition $b_0 = 0$ ensures the change of c_0 results in no change in bb_1, and according to Proposition 2, the condition $bb_1 = 0x7aeba0cd$ ensures that the change of c_0 results in no change in bb_2 with the probability of 2^{-14}. Therefore, we can modify c_0 to correct a change in m_2 without altering the chaining variables b_0, bb_1, $\cdots$, bb_{26} with probability 2^{-14}.
5. In the second round of Line1 operation, changing m_2 results in altering cc_{26}. In the second round of Line2 operation, changing m_2 doesn't change ddd_{26}.

$$H_0 = b_0 + cc_{26} + ddd_{26} \quad (4)$$
$$cc_{26} = bb_{25} = (aa_{24} + G(bb_{24}, cc_{24}, dd_{24}) + m_2 + 0x5a827999) \lll 3 \quad (5)$$

Combined with equation (4) and equation (5), we can modify m_2 as follows in order to set $H_0 = H_0^*$.

$$m_2 \leftarrow (H_0^* - b_0 - ddd_{26}) >>> 3 - aa_{24} - G(bb_{24}, cc_{24}, dd_{24}) - 0x5a827999$$

6. We can change c_0 in order to correct the change of m_2 in Line1 operation, i. e. to keep bb_3 unchanged. $c_{0,1}^*$ denotes the c_0 after changed in Line1 operation. Denote $m_2^{'}$ by the original message word before changed. $m_2^{''}$ is the message word after changed. From the algorithm of RIPEMD, we can get the following equation (6) and equation (7):

$$\begin{aligned} bb_3 &= (aa_2 + F(bb_2, cc_2, dd_2) + m_2^{'}) \lll 11 \\ &= (c_0 + F(bb_2, bb_1, b_0) + m_2^{'}) \lll 11 \qquad (6) \\ bb_3 &= (c_{0,1}^* + F(bb_2, bb_1, b_0) + m_2^{''}) \lll 11 \qquad (7) \end{aligned}$$

From equation (6) and equation (7) we can get $c_0 + m_2^{'} = c_{0,1}^* + m_2^{''}$ (8)

7. Similarly, we can change c_0 to correct the change of m_2 in Line2 operation, i. e. to keep bbb_3 unchanged. $c_{0,2}^*$ denotes the c_0 after changed in Line2 operation. From the algorithm of RIPEMD, we can get the following equation (9) and equation (10):

$$bbb_3 = (c_0 + F(bbb_2, bbb_1, b_0) + m_2^{'}) \lll 11 \qquad (9)$$

$$bbb_3 = (c_{0,2}^* + F(bbb_2, bbb_1, b_0) + m_2^{''}) \lll 11 \qquad (10)$$

From equation (9) and equation (10) we can get $c_0 + m_2^{'} = c_{0,2}^* + m_2^{''}$ (11)
From equation (8) and equation (11), easily we get $c_{0,1}^* = c_{0,2}^*$.
Therefore, the change in m_2 can be corrected by changing c_0 as follows:
$c_0 \leftarrow (bb_3 >>> 11) - F(bb_2, bb_1, b_0) - m_2^{''} = (bbb_3 >>> 11) - F(bbb_2, bbb_1, b_0) - m_2^{''} - 0x50a28be6$
With the new value c_0, the chaining variables bb_3 in Line1 operation and bbb_3 in Line2 operation are not altered.

All in all, firstly, we modify m_2 to get the output H_0^*, then we alter c_0 to correct the change in m_2 in Line2 operation, and to correct the change in m_2 in Line1 operation with probability 2^{-14}. We can get the outputs H_1^*, H_2^* and H_3^* by exhaustively search with 2^{96} trials. Thus the total cost is $2^{110} (= 2^{96} \times 2^{14})$ trials.

3.3 Preimage Attack on the 26-Step Reduced RIPEMD

In order to extend the preimage attack on CRIPEMD to the 26-step reduced RIPEMD, we take into consideration of two aspects: one is the padding rule of RIPEMD, and the other is the standard initial value. The padding rule only forces some constraints on the last message words, and there aren't any restrictions on the last message words in our attack. So the padding rule is not an obstacle. However, the initial value is a problem, the initial value in our attack is different from the standard one in the RIPEMD algorithm. We use two approaches to extend our attack to the attack based on the standard initial value. These approaches are also used in [15].

Basic Meet-in-the-Middle Approach. We will use the basic meet-in-the-middle approach to turn the attack to the preimage attack on the 26-step reduced RIPEMD. The approach is similar to the unbalanced meet-in-the-middle attack in [18]. If the complexity of the attack on the compression function is 2^x, then we hash $2^{\frac{n+x}{2}}$ random message blocks with the standard initial value, and compute $2^{\frac{n-x}{2}}$ preimages for the targeting value H^* by using the preimage attack on the compression function. By the birthday paradox, we expect one match. The complexity of the preimage attack on hash functions is $2^{1+\frac{n+x}{2}}$.

In our attack, we have $x = 110$. We hash 2^{119} random message blocks with the standard initial value with complexity 2^{119}, and compute 2^9 preimages for the targeting value H^* with complexity $2^{119} = 2^9 \times 2^{110}$. Therefore, the complexity of our attack on 26-step reduced RIPEMD is about 2^{120} hash computations.

Tree Approach. This approach is the meet-in-the-middle approach combined with a tree-based approach. The details of the approach can refer to [12] which describes the approach exactly. [19] describes a similar approach. Firstly, we compute 2^{16} multi-block preimages of the targeting value H^* by using the tree-based approach, which costs $2^{115} = 16 \times 2^{111}$ computations. Then we hash 2^{112} random message blocks with the standard initial value, and we expect to find one match according to the birthday paradox. Therefore, the preimage attack on 26-step reduced RIPEMD has the time complexity of about $2^{115.2} (= 2^{115} + 2^{112})$, and needs to store 2^{17} message blocks (2^{23} bytes).

Delayed-Start Attack. We can extend the attack to 29-step reduced RIPEMD, which is from step 2 to 30 by using the message word m_3 instead of m_2. The attack has the same complexity as the attack on 26 steps.

4 Preimage Attack on the Full RIPEMD

In this section, by the approach of speeding up the brute-force attack proposed in [20], we can reduce the complexity of the preimage attack on the full RIPEMD without the padding rule by 1 bit.

We will recall the speeding up approach as follows. Given the initial value IV and hash value H, we want to find a message block such that the hash value is equal to H under the initial value IV. Assume m_i and m_j form a local collision in the first round, and m_i, m_j appear at steps i_1, j_1 ($i_1 < j_1$) in the second round, then the values of the chaining variables from step 1 to i_1 can be reused with all m_i and corresponding m_j. Let m_i, m_j appear at steps i_2, j_2 ($i_2 < j_2$) in the last round, then the results from step j_2 to the last can be reused. Similarly, if m_i and m_j form a local collision in the last round, then this technique can also be achieved. The memory requirement of the attack is negligible.

We make a local collision from step 12 to step 16 in Line2 operation. Then the results from step 1 to 30 can be reused. Assume the initial value is IV and the hash value is $H = (H_0 H_1 H_2 H_3)$. We use the speeding up approach to invert the full RIPEMD in the following steps.

1. Randomly choose the message words m_i (i=0,...,9), and compute the chaining variables aaa_i, bbb_i, ccc_i and ddd_i (i=1,...,10).
2. Choose m_{10} such that $bbb_{11} = bbb_{10}$, i. e. $ccc_{12} = ddd_{12}$.
3. Randomly choose m_{11}. Choose m_{12} such that $bbb_{13} = 0$. Choose m_{13} such that $bbb_{14} = 1$.
4. Randomly choose other message words. Compute aa_i, bb_i, cc_i and dd_i (i=1,...,11). Compute aaa_i, bbb_i, ccc_i and ddd_i (i=1,...,30). Store the values aaa_{30}, bbb_{30}, ccc_{30} and ddd_{30} in a table.
5. For all 2^{32} m_{11}, compute the corresponding m_{15} such that bbb_{16} doesn't change. Compute aa_i, bb_i, cc_i and dd_i for $i = 12, ..., 48$.
6. Let $aaa_{48} = H_0 - aa_{48}$, $bbb_{48} = H_1 - bb_{48}$, $ccc_{48} = H_2 - cc_{48}$ and $ddd_{48} = H_3 - dd_{48}$, compute aaa_i, bbb_i, ccc_i and ddd_i in the reverse direction and get the value aaa_{33}. Check whether $aaa_{33} = bbb_{30}$ is in the table or not. If it is in the table, compute bbb_{32}, bbb_{31} and bbb_{30} and check all values are matched. Otherwise, choose m_{11} and repeat the process.

The complexity of the attack is $2^{127} = 2^{96} \times 2^{31}$. The reason is that the complexity of the above procedure is about $2^{31} = 2^{32} \times \frac{(48-11)+(48-33)}{48+48}$, and the success probability is $2^{-96} = 2^{-128} \times 2^{32}$.

5 Conclusion

In this paper, firstly, we present an inversion of the compression function of the 26-step reduced RIPEMD with complexity 2^{110}, and extend the preimage attack on the compression function to a preimage attack on the 26-step reduced RIPEMD with complexity $2^{115.2}$ and memory requirement of 2^{23} bytes. Then we extend the attack on 26-step reduced RIPEMD to 29 steps with the same complexity. In the last, we give a preimage on the full RIPEMD without the padding rule and reduce the complexity by 1 bit compared with the brute-force attack. Though the attacks require very large number of compression function evaluations, they are successful attacks on reduced RIPEMD theoretically. Moreover, it shows that the hash functions composed of two parallel lines are not secure as we expected. As far as we know, this is the first preimage attack on RIPEMD.

References

1. Rogaway, P.: Formalizing human ignorance. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)
2. Bosselaers, A., Preneel, B. (eds.): RIPE 1992. LNCS, vol. 1007. Springer, Heidelberg (1995)

3. Wang, X.Y., Lai, X.J., Feng, D.G., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
4. 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)
5. Wang, X.Y., Yu, H.B., Lisa, Y.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
6. Wang, X.Y., Lisa, Y., 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)
7. Wang, X.Y., Feng, F.D., Yu, X.: An attack on HAVAL function HAVAL-128, Science in China Ser. F Information Sciences 48(5), 1–12 (2005)
8. 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)
9. Mendel, F., Rechberger, C., Rijmen, V.: Update on SHA-1. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622. Springer, Heidelberg (2007), http://rump2007.cr.yp.to/
10. Dobbertin, H.: The first two rounds of MD4 are not one-way. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 284–292. Springer, Heidelberg (1998)
11. De, D., Kumarasubramanian, A., Venkatesan, R.: Inversion attacks on secure hash functions using SAT solvers. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 377–382. Springer, Heidelberg (2007)
12. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
13. Muller, F.: The MD2 hash function is not one-way. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 214–229. Springer, Heidelberg (2004)
14. Knudsen, L.R., Mathiassen, J.E.: Preimage and collision attacks on MD2. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 255–267. Springer, Heidelberg (2005)
15. Aumasson1, J., Meier, W., Mendel, F.: Preimage Attacks on 3-Pass HAVAL and Step-Reduced MD5. In: SAC 2008 (accepted) (to appear 2008)
16. Vaudenay, S.: On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 286–297. Springer, Heidelberg (1995)
17. Rivest, R.L.: The MD4 message digest algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
18. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
19. Mendel, F., Rijmen, V.: Weaknesses in the HAS-V compression function. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 335–345. Springer, Heidelberg (2007)
20. Sasaki, Y., Aoki, K.: Preimage Attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)

Full Key-Recovery Attack on the HMAC/NMAC Based on 3 and 4-Pass HAVAL

Hongbo Yu[1] and Xiaoyun Wang[2,⋆]

[1] Center for Advanced Study,Tsinghua University, Beijing 100084, China
yuhongbo@mail.sdu.edu.cn
[2] Tsinghua University and Shandong University, China
xiaoyunwang@tsinghua.edu.cn, xywang@sdu.edu.cn

Abstract. In this paper, we give the full key-recovery attacks on the HMAC/NMAC instantiated with 3 and 4-Pass HAVAL using our new differential paths. The complexity to recover the inner key is about 2^{103} MAC queries for the 3-Pass HAVAL and 2^{123} MAC queries for the 4-Pass HAVAL. The complexity to recover the outer key is about 2^{69} MAC queries and 2^{198} offline computations for the 3-Pass HAVAL based HMAC/NMAC. For the 4-Pass HAVAL case, the number of MAC queries for outer key-recovery is about 2^{103} and the offline work is about 2^{180} 4-Pass HAVAL computations.

Keywords: HMAC, NMAC, key-recovery, 3-Pass HAVAL, 4-Pass HAVAL.

1 Introduction

Recently the analysis of the MACs based on hash functions have attracted extensive attention in the field of cryptographic analysis. A message authentication code(MAC) is a function which takes a message and a secret key as inputs and produces an output called an *authentication tag*. HMAC [2] is a standardized hash-based MAC algorithm which is widely used as a MAC algorithm and a pseudorandom function generator. NMAC [2] is a generalized version of HMAC. They are proven secure under the assumption that the compression function of the underlying hash function is a pseudorandom function [1]. However, the attacks on hash functions [8,9,10,11,12,13] have shown that the prevailing hash functions such as MD4, HAVAL, MD5, SHA-0, SHA-1 are not collision resistant. Therefore research on the HMAC/NMAC based on those hash functions has become a hot topic. There are three kinds of attacks on HMAC/NMAC: *distinguishing attack*, *forgery attack* and *key-recovery attack*. We focus on the key-recovery attack by trying to recover the inner and outer key of HMAC/NMAC instantiated by 3 and 4-Pass HAVAL.

At Asiacrypt'06, Contini and Yin [3] used collision techniques to obtain forgery and partial key-recovery attacks on HMAC/NMAC instantiated with

⋆ Supported by 973 Project(No.2007CB807902), 863 Project(No.2006AA01Z420) and the National Natural Science Foundation of China(NSFC Grant No.60803125).

MD4, MD5, SHA-0 and reduced SHA-1. At SCN 2006, Kim *et.al* [5] give distinguishing, forgery and partial key-recovery attacks on the HMAC/NMAC based on full or reduced HAVAL, MD4, MD5, SHA-0 and SHA-1. At FC 2007, Rechberger and Rijmen [7] proposed a full key-recovery attack in the related-key setting on NMAC with full MD5 and 34-step SHA-1. At Crypto 2007, Fouque *et.al* presented the first full key-recovery attacks on HMAC/NMAC based on MD4 using the IV-dependent collision differential path. Their attack can derive bits of outer key k_1 by observing whether or not collisions occurred for the outer MD4. At Eurocrypt 2008, Wang *et.al* [14] gave a new outer key-recovery attack on HMAC/NMAC-MD4 by using the near-collision differential path. They also extended the attack of [3] into a full key-recovery attack on NMAC-MD5 in the related-key setting. In FSE 2008 [6], Lee *et.al* found a 3-Pass HAVAL collision differential path with probability 2^{-114}, and that allowed to inner key-recovery attack on HMAC/NMAC based on 3-Pass HAVAL with 2^{122} MAC queries and 2^{96} offline computations.

In this paper, we extend the technique of Contini-Yin [3] to recover the inner key for the HMAC/NMAC based on 3 and 4-Pass HAVAL using our new differential paths. For the outer key-recovery, we joint the IV-dependent technique in [4] and the near-collision technique in [14] together to recover more key bits by using our new IV-dependent near-collision differential path. Our inner key-recovery attack on HMAC/NMAC based on 3-Pass HAVAL is more efficient than that of Lee *et.al*'s [6]. According to our knowledge, this is the first full key-recovery attack for the HMAC/NMAC based on 3 and 4-Pass HAVAL. The main results of this paper are listed in Table 1.

Table 1. Summary of our key-recovery attacks on HAVAL-based HMAC/NMAC

Hash functions	type	Probability of differentials	Online queries	Offline computations
3-Pass HAVAL	Inner K.	2^{-96}	2^{103}	2^{70}
4-Pass HAVAL	Inner K.	2^{-121}	2^{123}	2^{39}
3-Pass HAVAL	Outer K.	2^{-64}	2^{69}	2^{198}
4-Pass HAVAL	Outer K.	2^{-98}	2^{103}	2^{180}

This paper is organized as follows. In section 2, we describe the algorithms of HAVAL and HMAC/NMAC briefly. In section 3, we introduce our inner key-recovery attack on the HMAC and NMAC based on 3 and 4-Pass HAVAL. In section 4, we give the outer key-recovery attack on the HMAC/NMAC based on 3 and 4-Pass HAVAL. Finally we conclude this paper in section 5.

2 Algorithm Description

2.1 Description of 3 and 4-Pass HAVAL

The original description of the HAVAL algorithm can be found in [15]. In this section, we only describe the compression functions for the 3 and 4-Pass HAVAL for our purpose of attacks. The compression function of HAVAL takes a 1024-bit message $M = (m_0, m_1, ..., m_{31})$ and a 256-bit initial value $h_{in} = (a_0, b_0, ..., h_0)$ as inputs, and produces a 256-bit hash value h_{i+1} as output. The compression function is defined in the case of 3 and 4-Pass as follows:

– Initialize chaining variables $(a, b, ..., h)$ as $(a_0, b_0, ..., h_0)$.
– Perform the following 32×3 (or 4):

```
For $i$=0 to 2 (or 3)
    For $j = 0$ to 31
        $p := f_{i+1}(g, f, e, d, c, b, a)$
        $r := (p \ggg 7) + (h \ggg 11) + m_{ord(i,j)} + k_{i,j}$
        $(a, b, c, d, e, f, g, h) := (r, a, b, c, d, e, f, g)$
```

The constant $k_{i,j}$ and the order of the messages words in each pass can be found in [15]. The Boolean functions (round functions) involved in the 3 and 4-Pass HAVAL are defined in Table 4.
– Output the 256-bit value $h_{out} := (a + a_0, b + b_0, ..., h + h_0)$.

Some main properties of the round function f_1 of 3-Pass HAVAL that are used to find differential paths are listed in Tables 5 of Appendix. It is easy to deduce the similar properties of the other round functions for the 3 and 4-Pass HAVAL.

2.2 Description of NMAC and HMAC

HMAC and NMAC are both hash based MACs. Let H be the underlying hash function and f be the compression function. The basic design approach for NMAC is to replace the fixed IV in H with a secret key. Using the notations of the paper [2], let $f_k(x) = f(k, x)$ denote the keyed compression function and $H_k(x) = H(k, x)$ denote the keyed hash function. Let (k_1, k_2) be a pair of independent keys, the NMAC algorithm, on input message M and the secret key (k_1, k_2), is defined as :

$$NMAC_{(k_1,k_2)}(M) = H_{k_1}(H_{k_2}(M)).$$

HMAC is a variant of NMAC that uses a fixed IV so that it can be implemented by simply calling the existing hash functions. The function HMAC works on an arbitrary length input message M and a fixed length random string k as its keys:

$$HMAC_k(M) = H(\overline{k} \oplus opad || H((\overline{k} \oplus ipad) || M)),$$

where $\overline{k}$ is the key k padded to full b-bit block size by appending zero bits. *ipad* and *opad* are two b-bit constants. Let $k_1 = f(\overline{k} \oplus opad)$ and $k_2 = f(\overline{k} \oplus ipad)$, then HMAC can be seen as a special case of NMAC. So in the following attacks, we only focus on the key-recovery attack on the NMAC and it is also applicable to HMAC.

3 Inner Key-Recovery on NMAC Based on 3 and 4-Pass HAVAL

In this section, we extend Contini-Yin's inner key-recovery technique [3] on NMAC to the 3 and 4-Pass HAVAL.

3.1 Inner Key-Recovery on NMAC Based on 3-Pass HAVAL

In this section, we find five differential paths with almost the same probability 2^{-98} for inner key-recovery attack on NMAC based on 3-Pass HAVAL. The message difference $\Delta M = (\Delta m_i)_{(0 \le i \le 31)}$ selected for these paths is

$$\Delta m_i = \begin{cases} 2^{30}, i = 20 \\ 0, \quad 0 \le i \le 31, i \ne 20. \end{cases}$$

The five differential paths and their precise probabilities are shown in Table 6. The total probability of the differential paths in Table 6 is 2^{-96}, i.e, for any random message M, the message pair $(M, M + \Delta M)$ obeys one of the five paths with probability 2^{-96}.

For a reasonable success rate, we need $2^2 \times 2^{96} = 2^{98}$ chosen message pairs to get the collide messages $(M, M + \Delta M)$, and they obey one of the five paths in Table 6. No matter which path they obey, some variable conditions generated by M are determined. If we can recover the values of successive eight chaining variables a_i to a_{i+7} generated by M, the inner key k_2 can be recovered by backward computation. Let P_r is the probability of the differential path from step r to the last step. In order to carry out our work, we list the sufficient and necessary conditions in the chaining variables a_{20} to a_{27} and the corresponding probability P_r in Table 2.

Utilizing the conditions in a_{20}, a_{21}, a_{22}, a_{23}, a_{25} and a_{27}, we can recover the least significant 31 bits in these chaining variables by changing the corresponding bit in m_{19}. For a reasonable success rate, it needs about $\frac{2^3}{Pr_{i+1}}$ message pairs $(M^*, M^* + \Delta M)$ to recover one bit of a_i. The structure of the message M^* and the detail bit-recovery technique can be referred to [3]. The recovered bits in a_{20} to a_{27} and the corresponding data complexity are also given in Table 2.

The total number of recovered bits in the eight successive chaining variables a_{20} to a_{27} is $31 \times 6 = 186$. We guess the unknown $256 - 186 = 70$ bits and compute backward to get the input k_2. The right key k_2 will result to $H_{k_2}(M) = H_{k_2}(M + \Delta M)$.

The data complexity of this attack includes two parts: searching the collision message pairs and recovering the chaining variables. Adding two parts, we get a complexity about 2^{103}. The success rate of recovering a single bit in Table 2 is $1 - \frac{1}{e^3}$, which is very close to 1. In total, the success rate for recovering all 186 bits is about 0.94. The attack does not require any storage, its time complexity is only the time required to compute the MAC values for the chosen messages.

3.2 Inner Key-Recovery on NMAC Based on 4-Pass HAVAL

For 4-Pass HAVAL, we select the message difference $\Delta m_5 = 2^{19}$ and construct the 4-Pass differential paths. The paths are composed of two inner collisions, step 6 to 33 and step 95 to 123. The two inner collisions are shown in Tables 7 and 8. The first inner collision includes two paths and each of them has the probability 2^{-59}. The second inner collision from step 95-123 includes six paths

Table 2. Recovered bits from a_{20} to a_{27} for inner key-recovery attack on NMAC based on 3-Pass HAVAL

step	Output	Condition on relevant bit	Pr	Recovered bit	Data complexity
20	a_{20}	$a_{20,30} = a_{19,30}$	$P_{21} = 2^{-93.26}$	least significant 31 bits	$2^{102.16}$
21	a_{21}	$a_{21,30} = 0$	$P_{22} = 2^{-92.26}$	least significant 31 bits	$2^{101.16}$
22	a_{22}	$a_{22,30} = 0$	$P_{23} = 2^{-91.26}$	least significant 31 bits	$2^{100.16}$
23	a_{23}	$a_{23,19} = 0,\ a_{23,30} = 1$	$P_{24} = 2^{-89.26}$	least significant 31 bits	$2^{98.16}$
24	a_{24}	No conditions		0	
25	a_{25}	$a_{25,19} = 0,\ a_{25,30} = 0$	$P_{26} = 2^{-87.26}$	least significant 31 bits	$2^{96.16}$
26	a_{26}	No conditions		0	
27	a_{27}	$a_{27,19} = 0,\ a_{27,30} = 0$	$P_{28} = 2^{-85.26}$	least significant 31 bits	$2^{94.16}$

Table 3. Recovered bits on a_{22} to a_{29} for inner key-recovery attack on HMAC based on 4-Pass HAVAL

step	Output	Condition on relevant bit	Pr	Recovered bit	Data complexity
22	a_{22}	$a_{22,30} = 1$	$P_{23} = 2^{-69.27}$	least significant 31 bits	$2^{78.17}$
23	a_{23}	$a_{23,30} = 0$	$P_{24} = 2^{-68.27}$	least significant 31 bits	$2^{77.17}$
24	a_{24}	$a_{24,30} = 0$	$P_{25} = 2^{-67.27}$	least significant 31 bits	$2^{76.17}$
25	a_{25}	$a_{25,30} = 1$	$P_{26} = 2^{-66.27}$	least significant 31 bits	$2^{75.17}$
26	a_{26}	$a_{26,30} = 0$	$P_{27} = 2^{-65.27}$	least significant 31 bits	$2^{74.17}$
27	a_{27}	$a_{27,30} = 0$	$P_{28} = 2^{-64.27}$	least significant 31 bits	$2^{73.17}$
28	a_{28}	$a_{28,30} = 0$	$P_{29} = 2^{-63.27}$	least significant 31 bits	$2^{72.17}$
29	a_{29}	No conditions		0	

that contribute a total probability of $2^{-63.23}$. So the probability for the entire 4-Pass HAVAL differential paths is about 2^{-121}.

The first step for the inner key-recovery attack is to find a pair of collided messages $(M, M + \Delta M)$ and this step has a data complexity about $O(2^{123})$ according to the differential paths in Table 7 and 8. Using the Boolean function properties of 4-Pass HAVAL, we can deduce a set of sufficient and necessary conditions on the eight chaining variables $(a_{22}, a_{23}, ..., a_{29})$. Then we can give the recovered bits in Table 3.

Now we have recovered $31 \times 7 = 217$ bits of variables and the remaining 39 bits can be exhaustively searched. The data complexity is determined by searching the collision message pairs as for the case of the 4-Pass HAVAL. The success rate of this attack is about 0.93.

4 Outer Key-Recovery on NMAC Based on 3 and 4-Pass HAVAL

In this section, we find the new near-collision differential paths for 3 and 4-Pass HAVAL. Combining the IV-dependent technique in [4] and the near-collision technique in [14], we give the first outer key recovery attack on NMAC based on 3 and 4-Pass HAVAL using our new differentials.

4.1 Outer Key-Recovery on NMAC Based on 3-Pass HAVAL

Suppose that the inner key k_2 of NMAC based on 3-Pass HAVAL is known and we want to recover the outer key k_1, which will be decomposed as eight 32-bit variables k_a, k_b, k_c, k_d, k_e, k_f, k_g and k_h. Because HAVAL uses the Davies-Meyer mode, the 256-bit output of NMAC based on 3-Pass HAVAL is $(k_a+a_{96}, k_b+a_{95}, k_c+a_{94}, k_d+a_{93}, k_e+a_{92}, k_f+a_{91}, k_g+a_{90}, k_h+a_{89})$, denoted as $(h_a, h_b, h_c, h_d, h_e, h_f, h_g, h_h)$. Here, a_{96} down to a_{89} are the final values of the eight 32-bit chaining variables.

Select the message difference $\Delta m_5 = 2^i$, we can find eight IV-dependent near-collision differential paths named DP_1, DP_2, ..., DP_8 orderly in Table 9, and they are all based on a local collision from step 6 to 33. The message word m_5 occurs at step 6, 33 and 95. The differential paths DP_1, DP_2, DP_3 and DP_4 work for the cases $i = 0 \sim 2, 6 \sim 9, 11 \sim 14, 18 \sim 30$. And the differential paths DP_5, DP_6, DP_7 and DP_8 work for the cases $i = 0 \sim 9, 11 \sim 13, 17 \sim 24, 28 \sim 31$. The other values of i fail because of bit expansion. Each of the local collision path holds with probability about 2^{-67}. So the message pair $(M, M + \Delta M)$ obey one of the eight paths with probability 2^{-64}. These paths are all dependent with k_a of IV. The outer key-recovery attack on NMAC based on 3-Pass HAVAL can be divided into the following process.

Recovering the key k_a: According to the properties of the 3-Pass HAVAL round functions listed in Table 5, we know that any one-block message pair $(M, M + \Delta M)$ with $\Delta m_5 = 2^i$ $(i \neq 10, 15, 16)$ will obey one of the eight paths in Table 9 and cause the collision in $(h_c, h_d, h_e, h_f, h_g, h_h)$ with probability about 2^{-61} when $k_{a,i+1} = 0$, otherwise when $k_{a,i+1} \neq 0$, M and $M + \Delta M$ consist of a collision on the six output values with probability lower than 2^{-69}.

So for each i of the message difference $\Delta m_5 = 2^i$ where i runs from 0 to 31 and $i \neq 10, 15, 16$, we can recover one bit on k_a of IV using the technique in [4] as follows:

1. Generate pairs of message satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$ where ΔM is $\Delta m_5 = 2^i$.
2. Send M and M' to the NMAC based on 3-Pass HAVAL oracle. Once roughly 2^{61} pairs of messages (M, M') are queried, if a collision of $(h_c, h_d, h_e, h_f, h_g, h_h)$ is obtained, we judge the outer key bit $k_{a,i} = 0$, otherwise we tell $k_{a,i} = 1$. So with 2^{62} queries, we will recover one bit of k_a.
3. Change i, and repeat step 1 and 2 until all values of i are used.

In this way, we can recover 29 bits of k_a.

Detecting the bits in a_{95} and a_{92}: For those i that result to the *i-th* bit of k_a equal to zero, we use the near-collision technique in [14] to recover k_e and k_b as follows:

1. Generate the message pairs which make $H_{k_2}(M') = H_{k_2}(M) + \Delta M$ where ΔM is $\Delta m_5 = 2^i$.

2. Send such messages M and M' to the NMAC based on 3-Pass HAVAL oracle to obtain each of the following two kinds of near-collisions:
   - Pairs $(M_e^i, M_e^{i'})$: such that $\Delta h_a = 0$, $\Delta h_b = 2^i$, $\Delta h_c = 0$,..., $\Delta h_h = 0$.
   - Pairs $(M_b^i, M_b^{i'})$: such that $\Delta h_a = \pm 2^{i+1-7}$, $\Delta h_b = 2^i$, $\Delta h_c = 0$,..., $\Delta h_h = 0$.
3. Change i, and repeat step 1 and 2 until all values of i that result to the i-th of $k_a = 0$ are used.

Once we obtain the pairs $(M_e^i, M_e^{i'})$, according to Boolean function properties of 3-Pass HAVAL, we can tell $a_{92,i} = 1$. If we get the pairs $(M_b^i, M_b^{i'})$, we can judge $a_{95,i} = 1$.

In total, the number of the message pairs $(M_e^i, M_e^{i'})$ and $(M_b^i, M_b^{i'})$ that we have obtained such that $a_{92,i} = 1$ and $a_{95,i} = 1$ is about 29.

Recovering the key $\boldsymbol{k_b}$ and $\boldsymbol{k_e}$: We can recover the key k_b and k_e by the offline work using the technique in [14]. The methods to recover k_b and k_e are the same. We only pick k_b as an example.

1. Guess the bits $k_{b,10}, k_{b,15}, k_{b,16}$, and the bits $k_{b,i}$ corresponding to $k_{a,i} = 1$ that we fail obtaining M_b^i. On average, the total number of possibilities is about 2^{17}.
2. Calculate other bits of k_b from the least significant to the most significant bit using M_b^i. Suppose we have get the first t bits of k_b, the $(t+1)$-th bit of k_b will be calculated using the obtained M_b^{t+1}. Compare $k_{b,t\sim 0}$ with $h_{b,t\sim 0}$. If $k_{b,t\sim 0} > h_{b,t\sim 0}$, there exists a carry from bit t to $t+1$ during the computation of $k_b + a_{95}$. Otherwise, there will be no carry from bit t to $t+1$ during the computation of $k_b + a_{95}$. Since $a_{95,t+1} = 1$, the carry influence is known, and the value $h_{b,t+1}$ is known, so the value $k_{b,t+1}$ can be calculated.

Complexity computation: As explained in section 4.1 , the key number we can obtain on average is $29 + 29 = 58$ by online work, and the remaining keys can be search exhaustively. So the complexity of the offline exhaustive search needs 2^{198} 3-Pass HAVAL computations.

Now we only need to analyze the complexity of online work. To recover k_a, we need 29×2^{61} message pairs. The message $(M_b^i, M_b^{i'})$ can be obtained with the probability $2^{-61} \times 2^{-2} = 2^{-63}$. The message $(M_e^i, M_e^{i'})$ can be obtained with the probability $2^{-61} \times 2^{-3} = 2^{-64}$. In this way, the total number of message pairs required to recover the 58 key bits is about $29 \times 2^{64} < 2^{69}$. The cost of the message generation is the offline work because we have known k_2. Using the birthday attack, it need about 2^{163} 3-Pass HAVAL computations to get the 2^{69} message pairs needed in the outer functions.

Overall, the complexity to recover the outer key of NMAC based on 3-Pass HAVAL is about 2^{69} online MAC queries and 2^{198} offline operations.

4.2 Outer Key-Recovery on NMAC Based on 4-Pass HAVAL

Let the 256-bit output value of HMAC/NMAC based on 4-Pass HAVAL be $(h_a, h_b, h_c, h_d, h_e, h_f, h_g, h_h)$, where $h_a = k_a + a_{128}$, $h_b = k_b + a_{127}$, $h_c = k_c + a_{126}$, $h_d = k_d + a_{125}$, $h_e = k_e + a_{124}$, $h_f = k_f + a_{124}$, $h_g = k_g + a_{122}$, $h_h = k_h + a_{121}$.

In the case of 4-Pass HAVAL, we use the near-collision differential path in Table 10 to recover the out keys k_1 of the NMAC. The message difference is $\Delta m_5 = 2^i$. The near-collision differential path can be divided into two parts. The first local collision path is from step 6 to 33 which include two path and each have a probability 2^{-59}. And the second near collision is from step 95 to 128.

According to the Boolean functions properties of 4-Pass HAVAL, the near-collision differential paths in Table 10 is independent with the IV. We can recover the keys according to the shape of the near-collision.

Extracting the bit values a_{122}, a_{125}, a_{126} and a_{127} : For each $0 \leq i \leq 31$, $i \neq 10, 15, 16$, the key recovery processor as follows:

1. Generate pairs of message satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$ where ΔM is $\Delta m_5 = 2^i$.
2. Send such messages M and M' to the NMAC based on 4-Pass HAVAL oracle to obtain the following near-collisions shape:
   – (M_i, M_i'): $\Delta h_a = \pm 2^{i-18}$, $\Delta h_b = 2^{i-12}$, $\Delta h_c = 0$, $\Delta h_d = 0$, $\Delta h_e = 0$, $\Delta h_f = 2^i$, $\Delta h_g = 0$, $\Delta h_h = 0$.
3. Change i, and repeat step 1 and 2 until all values of i are used.

Once we obtain the pair (M_i, M_i'), we can judge that $a_{122,i} = 1$, $a_{125,i} = a_{123,i}$, $a_{126,i+1} = 1$, $a_{127,i-12} = 1$. So it easy to recover the 29 key bits of k_b, k_c, k_d and k_g utilizing the technique described in section 4.1.

Thus, we can recover $29 \times 4 = 116$ key bits by the online work. The left 140 bits can be obtained by exhaustive search. The probability to obtain a pair (M_i, M_i') is about $2^{-58} \times 2^{-40} = 2^{-98}$, where 2^{-58} is the probability of local collision from 6 to 33 and 2^{-40} is the near-collision probability from step 95 to 128. So the total online complexity to recover the 116 key bits is about $29 \times 2^{98} \approx 2^{103}$. In order to obtain the 2^{103} online message pairs, it needs about 2^{180} offline 4-Pass HAVAL computations.

5 Conclusion

This paper give the full key recovery attack on the HMAC/NMAC based on the 3 and 4-Pass HAVAL. Our main contribution is to find series of collision and near-collision differential paths which is appropriate to attack the HMAC/NMAC based on 3 and 4-Pass HAVAL utilizing the recent technique [3,4,14].

References

1. Bellare, M.: New Proofs for NMAC and HMAC: Security without Collision–Resistance, http://eprint.iacr.org/2006/043.pdf
2. 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)

3. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
4. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
5. Kim, J.-S., Biryukov, A., Preneel, B., Hong, S.H.: On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (Extended abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
6. Lee, E., Chang, D., Kim, J.-S., Sung, J., Hong, S.H.: Second preimage attack on 3-pass HAVAL and partial key-recovery attacks on HMAC/NMAC-3-pass HAVAL. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 189–206. Springer, Heidelberg (2008)
7. Rechberger, C., Rijmen, V.: On authentication with HMAC and non-random properties. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 119–133. Springer, Heidelberg (2007)
8. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
9. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
10. Wang, X., Feng, F., Yu, X.: An attack on HAVAL function HAVAL-128. Science in China Ser. F Information Sciences 48(5), 1–12 (2005)
11. Wang, X., Yu, H., 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)
12. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
13. Yu, H., Wang, X., Yun, A., Park, S.: Cryptanalysis of the full HAVAL with 4 and 5 passes. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 89–110. Springer, Heidelberg (2006)
14. Wang, L., Ohta, K., Kunihiro, N.: New key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 237–253. Springer, Heidelberg (2008)
15. Zheng, Y., Pieprzyk, J., Seberry, J.: HAVAL–A One-way Hashing Algorithm with Variable Length of Output. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 83–104. Springer, Heidelberg (1993)

Appendix

Table 4. Round functions of 3 and 4-Pass HAVAL

Pass	Round functions
3-Pass	$f_1(g,f,e,d,c,b,a) = cd \oplus ag \oplus bf \oplus ce \oplus e$
	$f_2(g,f,e,d,c,b,a) = adf \oplus bcf \oplus ef \oplus ef \oplus ac \oplus df \oplus bd \oplus bc \oplus fg \oplus g$
	$f_3(g,f,e,d,c,b,a) = def \oplus cf \oplus be \oplus dg \oplus ad \oplus a$
4-Pass	$f_1(g,f,e,d,c,b,a) = bd \oplus fg \oplus ce \oplus ad \oplus a$
	$f_2(g,f,e,d,c,b,a) = abg \oplus bcf \oplus bg \oplus cg \oplus bd \oplus af \oplus cf \oplus be \oplus e$
	$f_3(g,f,e,d,c,b,a) = acg \oplus cd \oplus ae \oplus bg \oplus fg \oplus f$
	$f_4(g,f,e,d,c,b,a) = bcf \oplus ace \oplus afg \oplus ab \oplus cg \oplus af \oplus ef \oplus fg \oplus ae \oplus ag \oplus ad \oplus d$

Table 5. Some properties for the first round function f_1 of the 3-Pass HAVAL

Δa	Δb	Δc	Δd	Δe	Δf	Δg	$\Delta f_1 = 0$	$\Delta f_1 = 1$	$\Delta f_1 = -1$
0	0	0	0	0	0	0	1	–	–
1	0	0	0	0	0	0	$g = 0$	$g = 1,\ cd \oplus bf \oplus ce \oplus e = 0$	$g = 1,\ cd \oplus bf \oplus ce \oplus e = 1$
-1	0	0	0	0	0	0	$g = 0$	$g = 1,\ cd \oplus bf \oplus ce \oplus e = 1$	$g = 1,\ cd \oplus bf \oplus ce \oplus e = 0$
0	1	0	0	0	0	0	$f = 0$	$f = 1,\ cd \oplus ag \oplus ce \oplus e = 0$	$f = 1,\ cd \oplus ag \oplus ce \oplus e = 1$
0	-1	0	0	0	0	0	$f = 0$	$f = 1, cd \oplus ag \oplus ce \oplus e = 1$	$f = 1,\ cd \oplus ag \oplus ce \oplus e = 0$
0	0	1	0	0	0	0	$e = d$	$e \neq d,\ ag \oplus bf \oplus e = 0$	$e \neq d,\ ag \oplus bf \oplus e = 1$
0	0	-1	0	0	0	0	$e = d$	$e \neq d,\ ag \oplus bf \oplus e = 1$	$e \neq d,\ ag \oplus bf \oplus e = 0$
0	0	0	1	0	0	0	$c = 0$	$c = 1,\ ag \oplus bf \oplus ce \oplus e = 0$	$c = 1,\ ag \oplus bf \oplus ce \oplus e = 1$
0	0	0	-1	0	0	0	$c = 0$	$c = 1,\ ag \oplus bf \oplus ce \oplus e = 1$	$c = 1,\ ag \oplus bf \oplus ce \oplus e = 0$
0	0	0	0	1	0	0	$c = 1$	$c = 0,\ cd \oplus ag \oplus bf = 0$	$c = 0,\ cd \oplus ag \oplus bf = 1$
0	0	0	0	-1	0	0	$c = 1$	$c = 0,\ cd \oplus ag \oplus bf = 1$	$c = 0,\ cd \oplus ag \oplus bf = 0$
0	0	0	0	0	1	0	$b = 0$	$b = 1,\ cd \oplus ag \oplus ce \oplus e = 0$	$b = 1,\ cd \oplus ag \oplus ce \oplus e = 1$
0	0	0	0	0	-1	0	$b = 0$	$b = 1,\ cd \oplus ag \oplus ce \oplus e = 1$	$b = 1,\ cd \oplus ag \oplus ce \oplus e = 0$
0	0	0	0	0	0	1	$a = 0$	$a = 1,\ cd \oplus bf \oplus ce \oplus e = 0$	$a = 1,\ cd \oplus bf \oplus ce \oplus e = 1$
0	0	0	0	0	0	-1	$a = 0$	$a = 1,\ cd \oplus bf \oplus ce \oplus e = 1$	$a = 1,\ cd \oplus bf \oplus ce \oplus e = 0$

Table 6. Collision differential paths for HMAC based on 3-Pass HAVAL with probability $2^{-96.26}$

Step	m_{i-1}	Δm_i	DP_1	DP_2	DP_3	DP_4	DP_5
21	m_{20}	2^{30}	a_{21}[30]	a_{21}[30]	a_{21}[30]	a_{21}[30]	a_{21}[30]
22	m_{21}		a_{22}	a_{22}	a_{22}	a_{22}	a_{22}
23	m_{22}		a_{23}	a_{23}	a_{23}	a_{23}	a_{23}
24	m_{23}		a_{24}	a_{24}	a_{24}	a_{24}	a_{24}
25	m_{24}		a_{25}	a_{25}	a_{25}	a_{25}	a_{25}
26	m_{25}		a_{26}	a_{26}	a_{26}	a_{26}	a_{26}
27	m_{26}		a_{27}	a_{27}	a_{27}	a_{27}	a_{27}
28	m_{27}		a_{28}	a_{28}	a_{28}	a_{28}	a_{28}
29	m_{28}		a_{29}[19]	a_{29}[19]	a_{29}[19]	a_{29}[19]	a_{29}[19]
30	m_{29}		a_{30}	a_{30}	a_{30}	a_{30}	a_{30}
31	m_{30}		a_{31}	a_{31}	a_{31}	a_{31}	a_{31}
32	m_{31}		a_{32}	a_{32}	a_{32}	a_{32}	a_{32}
33	m_5		a_{33}	a_{33}	a_{33}	a_{33}	a_{33}
34	m_{14}		a_{34}	a_{34}	a_{34}	a_{34}	a_{34}
35	m_{26}		a_{35}	a_{35}	a_{35}	a_{35}	a_{35}
36	m_{18}		a_{36}	a_{36}	a_{36}	a_{36}	a_{36}
37	m_{11}		$a_{37}[-8, -9, 10]$	$a_{37}[-8, -9, 10]$	$a_{37}[-8, -9, 10]$	$a_{37}[-8, -9, 10]$	$a_{37}[-8, -9, 10]$
38	m_{28}		$a_{38}[3, 4, 5, -6]$	a_{38}	a_{38}	a_{38}	a_{38}
39	m_7		a_{39}	$a_{39}[3, 4, 5, -6]$	a_{39}	a_{39}	a_{39}
40	m_{16}		a_{40}	a_{40}	$a_{40}[3, 4, 5, 6]*$	a_{40}	a_{40}
41	m_0		a_{41}	a_{41}	a_{41}	$a_{41}[3, 4, 5, 6]*$	a_{41}
42	m_{23}		a_{42}	a_{42}	a_{42}	a_{42}	$a_{42}[3, 4, 5, 6]*$
43	m_{20}	2^{30}	a_{43}	a_{43}	a_{43}	a_{43}	a_{43}
44	m_{22}		$a_{44}[-31]$	$a_{44}[-31]$	$a_{44}[-31]$	$a_{44}[-31]$	$a_{44}[-31]$
45	m_1		a_{45}	a_{45}	a_{45}	a_{45}	a_{45}
46	m_{10}		a_{46}	a_{46}	a_{46}	a_{46}	a_{46}
47	m_4		a_{47}	a_{47}	a_{47}	a_{47}	a_{47}
48	m_8		a_{48}	a_{48}	a_{48}	a_{48}	a_{48}
49	m_{30}		a_{49}	a_{49}	a_{49}	a_{49}	a_{49}
50	m_3		a_{50}	a_{50}	a_{50}	a_{50}	a_{50}
51	m_{21}		a_{51}	a_{51}	a_{51}	a_{51}	a_{51}
52	m_9		$a_{52}[-20]$	$a_{52}[-20]$	$a_{52}[-20]$	$a_{52}[-20]$	$a_{52}[-20]$
53	m_{17}		a_{53}	a_{53}	a_{53}	a_{53}	a_{53}
54	m_{24}		a_{54}	a_{54}	a_{54}	a_{54}	a_{54}
55	m_{29}		a_{55}	a_{55}	a_{55}	a_{55}	a_{55}
56	m_6		a_{56}	a_{56}	a_{56}	a_{56}	a_{56}
57	m_{19}		a_{57}	a_{57}	a_{57}	a_{57}	a_{57}
58	m_{12}		a_{58}	a_{58}	a_{58}	a_{58}	a_{58}
59	m_{15}		a_{59}	a_{59}	a_{59}	a_{59}	a_{59}
60	m_{13}		$a_{60}[-9]$	$a_{60}[-9]$	$a_{60}[-9]$	$a_{60}[-9]$	$a_{60}[-9]$
61	m_2		a_{61}	a_{61}	a_{61}	a_{61}	a_{61}
62	m_{25}		a_{62}	a_{62}	a_{62}	a_{62}	a_{62}
63	m_{31}		a_{63}	a_{63}	a_{63}	a_{63}	a_{63}
64	m_{27}		a_{64}	a_{64}	a_{64}	a_{64}	a_{64}
65	m_{19}		a_{65}	a_{65}	a_{65}	a_{65}	a_{65}
66	m_9		a_{66}	a_{66}	a_{66}	a_{66}	a_{66}
67	m_4		a_{67}	a_{67}	a_{67}	a_{67}	a_{67}
68	m_{20}	2^{30}	a_{68}	a_{68}	a_{68}	a_{68}	a_{68}
			pr=$2^{-98.96}$	pr=$2^{-98.96}$	pr=$2^{-97.21}$	pr=$2^{-99.21}$	pr=$2^{-100.52}$

Table 7. First inner collision differential path(step 6-33) for NMAC based on 4-Pass HAVAL with probability 2^{-58}

step	m_i	Δm_i	DP_1	DP_2
6	m_5	2^{19}	$a_6[-19, 20]$	$a_6[19]$
7	m_6		a_7	a_7
8	m_7		a_8	a_8
9	m_8		a_9	a_9
10	m_9		$a_{10}[13, 14, 15, -16]$	a_9
11	m_{10}		a_{11}	a_{11}
12	m_{11}		a_{12}	a_{12}
13	m_{12}		a_{13}	a_{13}
14	m_{13}		a_{14}	$a_{14}[-8, 9]$
15	m_{14}		a_{15}	a_{15}
16	m_{15}		a_{16}	a_{16}
17	m_{16}		$a_{17}[-9]$	a_{17}
18	m_{17}		a_{18}	$a_{18}[2, 3, 4, -5]$
19	m_{18}		a_{19}	a_{19}
20	m_{19}		a_{20}	a_{20}
21	m_{20}		a_{21}	a_{21}
22	m_{21}		a_{22}	a_{22}
23	m_{22}		a_{23}	a_{23}
24	m_{23}		a_{24}	a_{24}
25	m_{24}		$a_{25}[-30]$	$a_{25}[-30]$
26	m_{25}		a_{26}	a_{26}
27	m_{26}		a_{27}	a_{27}
28	m_{27}		a_{28}	a_{28}
29	m_{28}		a_{29}	a_{29}
30	m_{29}		a_{30}	a_{30}
31	m_{30}		a_{31}	a_{31}
32	m_{31}		a_{32}	a_{32}
33	m_5	2^{19}	a_{33}	a_{33}
			$pr = 2^{-59}$	$pr = 2^{-59}$

Table 8. Second inner collision differential path(step 95-123) for NMAC based on 4-Pass HAVAL with probability $2^{-63.27}$

step	m_i	Δm_i	DP_1	DP_2	DP_3	DP_4	DP_5	DP_6
95	m_5	2^{19}	$a_{95}[-19, 20]$	$a_{95}[-19, 20]$	$a_{95}[-19, 20]$	$a_{95}[19]$	$a_{95}[19]$	$a_{95}[19]$
96	m_2		a_{96}	a_{96}	a_{96}	a_{96}	a_{96}	a_{96}
97	m_{24}		a_{97}	a_{97}	a_{97}	a_{97}	a_{97}	a_{97}
98	m_4		a_{98}	a_{98}	a_{98}	a_{98}	a_{98}	a_{98}
99	m_0		a_{99}	a_{99}	a_{99}	a_{99}	a_{99}	a_{99}
100	m_{14}		$a_{100}[13, 14, 15, 16]^*$	a_{100}	a_{100}	a_{100}	a_{100}	a_{100}
101	m_2		a_{101}	$a_{101}[13, 14, 15, 16]^*$	a_{101}	a_{101}	a_{101}	a_{101}
102	m_7		a_{102}	a_{102}	$a_{102}[13, 14, 15, 16]^*$	a_{102}	a_{102}	a_{102}
103	m_{28}		a_{103}	a_{103}	a_{103}	$a_{103}[-8, 9]$	$a_{103}[-8, 9]$	$a_{103}[-8, 9]$
104	m_{23}		a_{104}	a_{104}	a_{104}	a_{104}	a_{104}	a_{104}
105	m_{26}		a_{105}	a_{105}	a_{105}	a_{105}	a_{105}	a_{105}
106	m_6		a_{106}	a_{106}	a_{106}	a_{106}	a_{106}	a_{106}
107	m_{30}		$a_{107}[-9]$	$a_{107}[-9]$	$a_{107}[-9]$	a_{107}	a_{107}	a_{107}
108	m_{20}		a_{108}	a_{108}	a_{108}	$a_{108}[2, 3, 4, 5]^*$	a_{108}	a_{108}
109	m_{18}		a_{109}	a_{109}	a_{109}	a_{109}	$a_{109}[2, 3, 4, 5]^*$	a_{109}
110	m_{25}		a_{110}	a_{110}	a_{110}	a_{110}	a_{110}	$a_{110}[2, 3, 4, 5]^*$
111	m_{19}		a_{111}	a_{111}	a_{111}	a_{111}	a_{111}	a_{111}
112	m_3		a_{112}	a_{112}	a_{112}	a_{112}	a_{112}	a_{112}
113	m_{22}		a_{113}	a_{113}	a_{113}	a_{113}	a_{113}	a_{113}
114	m_{11}		a_{114}	a_{114}	a_{114}	a_{114}	a_{114}	a_{114}
115	m_{31}		$a_{115}[-30]$	$a_{115}[-30]$	$a_{115}[-30]$	$a_{115}[-30]$	$a_{115}[-30]$	$a_{115}[-30]$
116	m_{21}		a_{116}	a_{116}	a_{116}	a_{116}	a_{116}	a_{116}
117	m_8		a_{117}	a_{117}	a_{117}	a_{117}	a_{117}	a_{117}
118	m_{27}		a_{118}	a_{118}	a_{118}	a_{118}	a_{118}	a_{118}
119	m_{12}		a_{119}	a_{119}	a_{119}	a_{119}	a_{119}	a_{119}
120	m_9		a_{120}	a_{120}	a_{120}	a_{120}	a_{120}	a_{120}
121	m_1		a_{121}	a_{121}	a_{121}	a_{121}	a_{121}	a_{121}
122	m_{29}		a_{122}	a_{122}	a_{122}	a_{122}	a_{122}	a_{122}
123	m_5	2^{19}	a_{123}	a_{123}	a_{123}	a_{123}	a_{123}	a_{123}
			$pr = 2^{-64.99}$	$pr = 2^{-67.47}$	$pr = 2^{-65.19}$	$pr = 2^{-65.77}$	$pr = 2^{-68.05}$	$pr = 2^{-65.77}$

Table 9. The near-collision differential path of NMAC based on 3-Pass HAVAL with probability 2^{-64}

Step	m_i	Δm_i	DP_1	DP_2	DP_3	DP_4
6	m_5	2^i	$a_6[-i, i+1]$	$a_6[-i, i+1]$	$a_6[-i, i+1]$	$a_6[-i, i+1]$
7	m_6		a_7	a_7	a_7	a_7
8	m_7		a_8	a_8	a_8	a_8
9	m_8		a_9	a_9	a_9	a_9
10	m_9		$a_{10}[i-6, i-5, i-4, -(i-3)]$	a_{10}	a_{10}	a_{10}
11	m_{10}		a_{11}	$a_{11}[i-6, i-5, i-4, -(i-3)]$	a_{11}	a_{11}
12	m_{11}		a_{12}	a_{12}	$a_{12}[i-6, i-5, i-4, -(i-3)]$	a_{12}
13	m_{12}		a_{13}	a_{13}	a_{13}	$a_{13}[i-6, i-5, i-4, -(i-3)]$
14	m_{13}		a_{14}	a_{14}	a_{14}	a_{14}
15	m_{14}		a_{15}	a_{15}	a_{15}	a_{15}
16	m_{15}		a_{16}	a_{16}	a_{16}	a_{16}
17	m_{16}		$a_{17}[-(i-10)]$	$a_{17}[-(i-10)]$	$a_{17}[-(i-10)]$	$a_{17}[-(i-10)]$
18	m_{17}		a_{18}	a_{18}	a_{18}	a_{18}
19	m_{18}		a_{19}	a_{19}	a_{19}	a_{19}
20	m_{19}		a_{20}	a_{20}	a_{20}	a_{20}
21	m_{20}		a_{21}	a_{21}	a_{21}	a_{21}
22	m_{21}		a_{22}	a_{22}	a_{22}	a_{22}
23	m_{22}		a_{23}	a_{23}	a_{23}	a_{23}
24	m_{23}		a_{24}	a_{24}	a_{24}	a_{24}
25	m_{24}		$a_{25}[-(i-21)]$	$a_{25}[-(i-21)]$	$a_{25}[-(i-21)]$	$a_{25}[-(i-21)]$
26	m_{25}		a_{26}	a_{26}	a_{26}	a_{26}
27	m_{26}		a_{27}	a_{27}	a_{27}	a_{27}
28	m_{27}		a_{28}	a_{28}	a_{28}	a_{28}
29	m_{28}		a_{29}	a_{29}	a_{29}	a_{29}
30	m_{29}		a_{30}	a_{30}	a_{30}	a_{30}
31	m_{30}		a_{31}	a_{31}	a_{31}	a_{31}
32	m_{31}		a_{32}	a_{32}	a_{32}	a_{32}
33	m_5	2^i	a_{33}	a_{33}	a_{33}	a_{33}
$\cdots$			$\cdots$			
95	m_5	2^i	$a_{95}[i]$	$a_{95}[i]$	$a_{95}[i]$	$a_{95}[i]$
96	m_2		a_{96}	a_{96}	a_{96}	a_{96}

Step	m_i	Δm_i	DP_5	DP_6	DP_7	DP_8
6	m_5	2^i	$a_6[i]$	$a_6[i]$	$a_6[i]$	$a_6[i]$
7	m_6		a_7	a_7	a_7	a_7
8	m_7		a_8	a_8	a_8	a_8
9	m_8		a_9	a_9	a_9	a_9
10	m_9		a_{10}	a_{10}	a_{10}	a_{10}
11	m_{10}		a_{11}	a_{11}	a_{11}	a_{11}
12	m_{11}		a_{12}	a_{12}	a_{12}	a_{12}
13	m_{12}		a_{13}	a_{13}	a_{13}	a_{13}
14	m_{13}		$a_{14}[-(i-11), i-10]$	$a_{14}[-(i-11), i-10]$	$a_{14}[-(i-11), i-10]$	$a_{14}[-(i-11), i-10]$
15	m_{14}		a_{15}	a_{15}	a_{15}	a_{15}
16	m_{15}		a_{16}	a_{16}	a_{16}	a_{16}
17	m_{16}		a_{17}	a_{17}	a_{17}	a_{17}
18	m_{17}		$a_{18}[i-17, i-16, i-15, -(i-14)]$	a_{18}	a_{18}	a_{18}
19	m_{18}		a_{19}	$a_{19}[i-17, i-16, i-15, -(i-14)]$	a_{19}	a_{19}
20	m_{19}		a_{20}	a_{20}	$a_{20}[i-17, i-16, i-15, -(i-14)]$	a_{20}
21	m_{20}		a_{21}	a_{21}	a_{21}	$a_{21}[i-17, i-16, i-15, -(i-14)]$
22	m_{21}		a_{22}	a_{22}	a_{22}	a_{22}
23	m_{22}		a_{23}	a_{23}	a_{23}	a_{23}
24	m_{23}		a_{24}	a_{24}	a_{24}	a_{24}
25	m_{24}		$a_{25}[-(i-21)]$	$a_{25}[-(i-21)]$	$a_{25}[-(i-21)]$	$a_{25}[-(i-21)]$
26	m_{25}		a_{26}	a_{26}	a_{26}	a_{26}
27	m_{26}		a_{27}	a_{27}	a_{27}	a_{27}
28	m_{27}		a_{28}	a_{28}	a_{28}	a_{28}
29	m_{28}		a_{29}	a_{29}	a_{29}	a_{29}
30	m_{29}		a_{30}	a_{30}	a_{30}	a_{30}
31	m_{30}		a_{31}	a_{31}	a_{31}	a_{31}
32	m_{31}		a_{32}	a_{32}	a_{32}	a_{32}
33	m_5	2^i	a_{33}	a_{33}	a_{33}	a_{33}
95	m_5	2^i	$a_{95}[i]$	$a_{95}[i]$	$a_{95}[i]$	$a_{95}[i]$
96	m_2		a_{96}	a_{96}	a_{96}	a_{96}

Table 10. The near-collision differential path of NMAC based on 4-Pass HAVAL with probability 2^{-98}

step	m_i	Δm_i	DP_1	DP_2
6	m_5	2^i	$a_6[-i, i+1]$	$a_6[i]$
7	m_6		a_7	a_7
8	m_7		a_8	a_8
9	m_8		a_9	a_9
10	m_9		$a_{10}[i-6, i-5, i-4, -(i-3)]$	a_{10}
11	m_{10}		a_{11}	a_{11}
12	m_{11}		a_{12}	a_{12}
13	m_{12}		a_{13}	a_{13}
14	m_{13}		a_{14}	$a_{14}[-(i-11), i-10]$
15	m_{14}		a_{15}	a_{15}
16	m_{15}		a_{16}	a_{16}
17	m_{16}		$a_{17}[-(i-10)]$	a_{17}
18	m_{17}		a_{18}	$a_{18}[i-17, i-16, i-15, -(i-14)]$
19	m_{18}		a_{19}	a_{19}
20	m_{19}		a_{20}	a_{20}
21	m_{20}		a_{21}	a_{21}
22	m_{21}		a_{22}	a_{22}
23	m_{22}		a_{23}	a_{23}
24	m_{23}		a_{24}	a_{24}
25	m_{24}		$a_{25}[-(i-21)]$	$a_{25}[-(i-21)]$
26	m_{25}		a_{26}	a_{26}
27	m_{26}		a_{27}	a_{27}
28	m_{27}		a_{28}	a_{28}
29	m_{28}		a_{29}	a_{29}
30	m_{29}		a_{30}	a_{30}
31	m_{30}		a_{31}	a_{31}
32	m_{31}		a_{32}	a_{32}
33	m_5	2^i	a_{33}	a_{33}
...	...			
95	m_5	2^i	$a_{95}[i]$	$a_{95}[i]$
96	m_2		a_{96}	a_{96}
97	m_{24}		a_{97}	a_{97}
98	m_4		a_{98}	a_{98}
99	m_0		a_{99}	a_{99}
100	m_{14}		a_{100}	a_{100}
101	m_2		a_{101}	a_{101}
102	m_7		a_{102}	a_{102}
103	m_{28}		$a_{103}[i-11]$	$a_{103}[i-11]$
104	m_{23}		a_{104}	a_{104}
105	m_{26}		a_{105}	a_{105}
106	m_6		a_{106}	a_{106}
107	m_{30}		a_{107}	a_{107}
108	m_{20}		a_{108}	a_{108}
109	m_{18}		a_{109}	a_{109}
110	m_{25}		a_{110}	a_{110}
111	m_{19}		$a_{111}[i-22]$	$a_{111}[i-22]$
112	m_3		a_{112}	a_{112}
113	m_{22}		a_{113}	a_{113}
114	m_{11}		a_{114}	a_{114}
115	m_{31}		a_{115}	a_{115}
116	m_{21}		a_{116}	a_{116}
117	m_8		a_{117}	a_{117}
118	m_{27}		a_{118}	a_{118}
119	m_{12}		$a_{119}[i-1]$	$a_{119}[i-1]$
120	m_9		a_{120}	a_{120}
121	m_1		a_{121}	a_{121}
122	m_{29}		a_{122}	a_{122}
123	m_5	2^i	$a_{123}[i]$	$a_{123}[i]$
124	m_{15}		a_{124}	a_{124}
125	m_{17}		a_{125}	a_{125}
126	m_{10}		a_{126}	a_{126}
127	m_{16}		$a_{127}[i-12]$	$a_{127}[i-12]$
128	m_{13}		a_{128}	a_{128}

Memoryless Related-Key Boomerang Attack on the Full Tiger Block Cipher

Ewan Fleischmann, Michael Gorski, and Stefan Lucks

Bauhaus-University Weimar, Germany
{Ewan.Fleischmann,Michael.Gorski,Stefan.Lucks}@uni-weimar.de

Abstract. In this paper we present the first attack on the full 24 round internal block cipher of Tiger [1]. Tiger is a hash function proposed by Biham and Anderson at FSE'96. It takes about ten years until the first cryptanalytic result was presented by Kelsey and Lucks [10] at FSE'06. Up to now, the best known attack on the internal block cipher of Tiger is able to break 22 rounds. Our attack on the full 24 rounds of the Tiger block cipher has a data complexity of $2^{3.5}$ chosen plaintexts and ciphertexts, which can be called memoryless. This is since we do not have to store all the data generated in our attack. The time complexity is about $2^{259.5}$ 24-round Tiger encryptions. Moreover, we have further reduced the time complexity using a bit fixing technique to $2^{195.5}$ 24-round encryptions.

Keywords: differential cryptanalysis, related-key boomerang attack, Tiger block cipher.

1 Introduction

Tiger [1] is a 160-bit hash function based on MERKLE-DAMGÅRD [13, 6], which operates on 512-bit message blocks.

The first cryptanalytic result for Tiger was a collision attack on 17 of 24 rounds of Tiger proposed by Kelsey and Lucks [10] using 2^{49} compression function calls. They also presented a near-collision attack on 20-round Tiger with complexity 2^{49}. Mendel et al. [11] found a collision attack on 19-round Tiger with complexity 2^{62} and also a pseudo-near-collision for 22-round Tiger which needs about 2^{44} compression function calls. At AsiaCrypt'07 Mendel and Rijmen [12] improved the previous results and presented a pseudo-near-collision attack on the full 24-round Tiger hash function with complexity 2^{47} and a pseudo collision with the same complexity. Indesteege and Preneel [9] proposed a pre-image attack on 12 and 13 round Tiger and also a (2nd) pre-image attack on Tiger reduced to 12 rounds. The best cryptanalytic result on the Tiger block cipher is a 22-round related-key boomerang and rectangle attack [7].

In this paper we present the first attack that can break the full 24-round Tiger encryption mode. We use a related-key boomerang attack which has a data complexity of $2^{3.5}$ chosen plaintexts and ciphertexts. The time complexity is about $2^{259.5}$ 24-round Tiger encryptions. Moreover we have reduced the time complexity using a bit fixing technique to $2^{195.5}$ 24-round Tiger encryptions.

The paper is organized as follows: In Section 2 we give a brief description of the Tiger encryption mode. In Section 3 we describe the related-key boomerang attack. In Section 4, we present our related-key boomerang attack on the full Tiger encryption mode. Section 5 concludes the paper.

2 Description of the Tiger Block Cipher

The following notations are used in this paper:

$\boxplus$: addition modulo 2^{64} operation
$\boxminus$: subtraction modulo 2^{64} operation
$\boxtimes$: multiplication modulo 2^{64} operation
$X \ll i$: shift of word X by i bits to the left
$X \gg i$: shift of word X by i bits to the right

Tiger's compression function is based on applying an internal "block cipher like" function, which takes a 192-bit "plaintext" and a 512 bit key to compute a 192-bit "ciphertext". The "block cipher like" function is applied according to the Davies-Meyer construction: a 512-bit message block is basically used as a key to encrypt the 192-bit chaining value, and then the input chaining value is fed forward to make the whole thing non-invertible. In the remainder of this section, we will describe Tiger in sufficient detail to follow the course of our attack; for a more detailed description of the hash function and its design rationale, the reader is referred to [1]. If, for a given input chaining value, we generate two different keys with the same output chaining value, then we have found a collision for Tiger.

Tiger has been designed with 64-bit architectures in mind. Accordingly, we will denote a 64-bit unsigned integer as a "word". We will represent a word as a hexadecimal number. Tiger uses arithmetic operations (addition, subtraction and multiplication by small constants), bit-wise XOR, NOT, logical shift operations and S-Box applications. The arithmetic operations over words are modulo 2^{64}. The chaining value is represented internally as three 64-bit words, the message block as eight 64-bit words.

2.1 The Tiger Round Function

In the terminology of [14], Tiger's block cipher like function is a "target-heavy unbalanced Feistel cipher". The block is broken into three words, labeled A, B, and C. Each round, a message word X is XORed into C:

$$C := C \oplus X.$$

Then A and B are modified:

$$\begin{aligned} A &:= A \boxminus \mathbf{even}(C), \\ B &:= B \boxplus \mathbf{odd}(C), \\ B &:= B \boxtimes (\text{const}), \end{aligned}$$

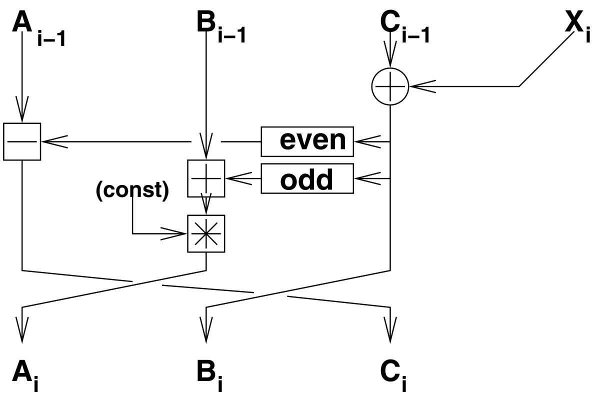


Fig. 1. The round function of Tiger

with a round-dependent constant (const) $\in \{5, 7, 9\}$. The results are then shifted around, so that A,B,C becomes B,C,A. See Figure 1. For the definition of **even** and **odd**, consider the word C being split into eight bytes C_0,..., C_7, with C_0 as the most significant byte. The functions **even** and **odd** employ four S-Boxes $T_1, \ldots, T_4 : \{0,1\}^8 \rightarrow \{0,1\}^{64}$ as follows:

$$\mathbf{even}(C) := T_1[C_0] \oplus T_2[C_2] \oplus T_3[C_4] \oplus T_4[C_6],$$

$$\mathbf{odd}(C) := T_1[C_7] \oplus T_2[C_5] \oplus T_3[C_3] \oplus T_4[C_1].$$

We will refer to C_0, C_2, C_4, and C_6 as the "even bytes of C".

The round function spreads changes around very quickly. A one-bit difference introduced into C in the first round will change about half the bits of the block by the end of the third round.

2.2 The Key Schedule

Tiger consists of 24 rounds. Each round uses one message word X_i as its round key. The first eight round keys $X_0, \ldots, X_7$ are identical to the 512-bit cipher key (or rather, to the 512-bit message block). The remaining 16 round keys are generated by applying the key schedule function:

$$(X_8, \ldots, X_{15}) := \text{KeySchedule}(X_0, \ldots, X_7)$$
$$(X_{16}, \ldots, X_{23}) := \text{KeySchedule}(X_8, \ldots, X_{15})$$

The key schedule function uses logical shifts on words, denoted by $\ll$ and $\gg$. E.g.,

$$1111\,5555\,9999\,FFFF \ll 5 = 222A\,AAB3\,333F\,FFE0 \quad \text{and}$$
$$222A\,AAB3\,333F\,FFE0 \gg 9 = 0011\,1555\,5999\,9FFF.$$

Further, it uses the bit-wise NOT function, e.g. for $X = \text{EEEE AAAA 6666 0000}$, the negation $\overline{X}$ of X is $\overline{X} = \text{1111 5555 9999 FFFF}$. The key schedule modifies its input $(x_0, \ldots, x_7)$ in two passes:

first pass	second pass
1. $x_0 := x_0 \boxminus (x_7 \oplus \text{Const}_1)$	9. $x_0 := x_0 \boxplus x_7$
2. $x_1 := x_1 \oplus X_0$	10. $x_1 := x_1 \boxminus (x_0 \oplus (\overline{x_7} \ll 19))$
3. $x_2 := x_2 \boxplus X_1$	11. $x_2 := x_2 \oplus x_1$
4. $x_3 := x_3 \boxminus (x_2 \oplus (\overline{x_1} \ll 19))$	12. $x_3 := x_3 \boxplus x_2$
5. $x_4 := x_4 \oplus x_3$	13. $x_4 := x_4 \boxminus (x_3 \oplus \overline{x_2} \gg 23))$
6. $x_5 := x_5 \boxplus x_4$	14. $x_5 := x_5 \oplus x_4$
7. $x_6 := x_6 \boxminus (x_5 \oplus (\overline{x_4} \gg 23))$	15. $x_6 := x_6 \boxplus x_5$
8. $x_7 := x_7 \oplus x_6$	16. $x_7 := x_7 \boxminus (x_6 \oplus \text{Const}_2)$

The final values $(x_0, \ldots, x_7)$ are used as the key schedule output. The constants are $\text{Const}_1 = \text{A5A5} \ldots \text{A5A5}$ and $\text{Const}_2 = 0123 \ldots \text{CDEF}$.

3 The Related-Key Boomerang Attack

The related-key boomerang attack was first published in [3]. It is a combination of the related-key attack [2] and the boomerang attack [15].

Related-key attacks consider the information that can be extracted from encryptions using related but unknown keys. Such ciphers with a weak key schedule are vulnerable to this kind of attack. The key scheduling algorithms of many block ciphers inherit obvious relationships between keys, which are called related-keys.

The boomerang attack is an extension to differential cryptanalysis [4] using adaptive chosen plaintexts and ciphertexts to attack block ciphers. An attacker can only apply a chosen plain- and ciphertext attack while choosing the relation between related, but unknown, keys.

We now describe the related-key boomerang attack in more detail. But first, we have to give some definitions.

Definition 1. *Let P, P' be two bit strings of the same length. The bit-wise XOR of P and P', $P \oplus P'$, is called the* difference *of P, P'.*

Definition 2. *$\alpha \to \beta$ is called a* differential *if α is the plaintext difference $P \oplus P'$ before some non-linear operation $f(\cdot)$ and β is the difference after applying these operation, i.e, $f(P) \oplus f(P')$. The probability p is linked on a differential saying that an α difference turns into a β difference with probability p.*

Theorem 1. *([15, Sec. 6].) Let $\alpha \to \beta$ be a differential with probability p as in Definition 2. Then the backward direction of this differential $\beta \to \alpha$ has also probability p.*

Two texts (P, P') are called a *pair*, while two pairs (P, P', O, O') are called a *quartet*. Differential cryptanalysis exploits high probability differentials $\alpha \to \beta$ between the first and a later round of a cipher. In general, an attacker tries to find high probability differentials over the first $N-1$ out of N rounds to exploit some subkey bits of the last (i.e. N'th) round. An attacker first computes the expected output of the $N-1$-th round differential and then guesses some subkey bits of the last round. He decrypts some ciphertexts under the guessed key bits one round and compares the resulting difference with the output difference of the $N-1$ round differential. For every matching he increases a counter to the used key bits and takes the key bits with the highest counter as the 'correct' one. If the differential is well chosen, i.e., its probability is high enough, these subkey bits can be computed much faster than applying exhaustive search.

The related-key boomerang attack extends differential cryptanalysis. The cipher is split into two sub-ciphers and the attacker tries to exploit a differential for each sub-cipher. These two differentials alone only cover a few rounds but together the whole cipher. Regularly, the differential probability decreases the more rounds are included. Therefore two short differential covering only a few rounds each will be used instead of a long one covering the whole cipher. In this way key bits can be computed with much fewer plaintexts than compared to classic differential cryptanalysis, since the differential probabilities are normally higher. Related-keys are used to exploit some weaknesses of the key schedule to enhance the probability of the differentials being used. We call such differentials related-key differentials, since different but related keys are used for encryption and decryption. We split the related-key boomerang attack into two steps. The *related-key boomerang distinguisher step* and the *key recovery step*. The related-key boomerang distinguisher is used to find all plaintexts sharing a desired difference that depends on the choice of the differential. These plaintexts are used in the key recovery step afterwards to recover subkey bits for the initial round key.

Distinguisher Step. During the *distinguisher step* we assume that the cipher can be treated as a cascade of two sub-ciphers $E_K(P) = E1_K(E0_K(P))$, where K is the key used for encryption and decryption. We assume that the related-key differential $\alpha \rightarrow \beta$ for $E0$ occurs with the probability p, while the related-key differential $\gamma \rightarrow \delta$ for $E1$ occurs with probability q, where α, β, γ and δ are differences. The backward direction $E0^{-1}$ and $E1^{-1}$ of the related-key differential for $E0$ and $E1$ are denoted by $\alpha \leftarrow \beta$ and $\gamma \leftarrow \delta$ and occur, because of Theorem 1, with probability p and q respectively. The related-key boomerang distinguisher involves four or more unknown but related keys

$$\Delta K' = K^a \oplus K^b = K^c \oplus K^d,$$
$$\Delta K^* = K^a \oplus K^c = K^b \oplus K^d,$$

where $\Delta K'$ and ΔK^* are a known key differences. The attack works as follows:

- For $i = 1, 2, \ldots, s$ do
  1. Choose plaintext $P^a_{0,i}$ and plaintext $P^b_{0,i} = P^a_{0,i} \oplus \alpha$, where the subindex 0 defines the plaintext.
  2. Ask for the encryption of $P^a_{0,i}$ under K^a, i.e., $P^a_{n,i} = E_{K^a}(P^a_{0,i})$ and $P^b_{0,i}$ under K^b, i.e., $P^b_{n,i} = E_{K^b}(P^b_{0,i})$.

3. Compute the new ciphertexts $P^c_{n,i} = P^a_{n,i} \oplus \delta$ and $P^d_{n,i} = P^b_{n,i} \oplus \delta$.
4. Ask for decryption of $P^c_{n,i}$ under K^c, i.e., $P^c_{0,i} = E^{-1}_{K^c}(P^c_{n,i})$ and $P^d_{n,i}$ under K^d, i.e., $P^d_{0,i} = E^{-1}_{K^d}(P^d_{n,i})$.
5. If $P^c_{0,i} \oplus P^d_{0,i} = \alpha$ store the quartet $(P^a_{0,i}, P^b_{0,i}, P^c_{0,i}, P^d_{0,i})$ in the set Θ.

Assume that a pair $(P^a_{0,i}, P^b_{0,i})$, $i \in \{1, \ldots, s\}$ with the difference α satisfies the differential $\alpha \rightarrow \beta$ with the probability p. Let $0 < b < n \leq N$ be an intermediate round, where N denotes the final round. Subcipher $E0$ covers rounds 1 to b and subcipher $E1$ rounds $b+1$ to N. The output of $E0$ is $P^a_{b,i}$ and $P^b_{b,i}$, i.e., $P^a_{b,i} = E0_{K^a}(P^a_{0,i})$ and $P^b_{b,i} = E0_{K^b}(P^b_{0,i})$. $P^a_{n,i}$ and $P^b_{n,i}$ have a difference $\beta = P^a_{b,i} \oplus P^b_{b,i}$ with probability p. Using the ciphertexts $P^a_{n,i}$ and $P^b_{n,i}$ (encrypted via E_{K^a} and E_{K^b}) we can compute the new ciphertexts $P^c_{n,i} = P^a_{n,i} \oplus \delta$ and $P^d_{n,i} = P^b_{n,i} \oplus \delta$. Let $P^c_{b,i} = E1^{-1}_{K^c}(P^c_{n,i})$ and $P^d_{b,i} = E1^{-1}_{K^d}(P^d_{n,i})$ be the decryption of $P^c_{n,i}$ and $P^d_{n,i}$ under $E1$. A difference δ turns into a difference γ after passing $E1^{-1}$ with probability q. Since $\delta = P^a_{n,i} \oplus P^c_{n,i} = P^b_{n,i} \oplus P^d_{n,i}$ we know that $P^a_{b,i} \oplus P^c_{b,i} = \gamma$ and $P^b_{b,i} \oplus P^d_{b,i} = \gamma$ with probability q^2. As we know that $P^a_{b,i} \oplus P^b_{b,i} = \beta$ with probability p, it follows that $(P^c_{b,i} \oplus P^d_{b,i}) = (P^c_{b,i} \oplus P^a_{b,i}) \oplus (P^a_{b,i} \oplus P^b_{b,i}) \oplus (P^b_{b,i} \oplus P^d_{b,i}) = \gamma \oplus \beta \oplus \gamma = \beta$ holds with probability pq^2. A β difference turns into an α difference after passing the differential $E0^{-1}$ with probability p. Thus, a pair of ciphertexts $(P^a_{0,i}, P^b_{0,i})$ with $P^a_{0,i} \oplus P^b_{0,i} = \alpha$ generates a new pair of plaintexts $(P^c_{0,i}, P^d_{0,i})$ where $P^c_{0,i} \oplus P^d_{0,i} = \alpha$ with probability $(pq)^2$. A quartet containing these two pairs is defined as:

Definition 3. *A quartet* $(P^a_{0,i}, P^b_{0,i}, P^c_{0,i}, P^d_{0,i})$ *which satisfies*

$$\begin{aligned} P^a_{0,i} \oplus P^b_{0,i} &= P^c_{0,i} \oplus P^d_{0,i} = \alpha, \\ P^a_{b,i} \oplus P^b_{b,i} &= P^c_{b,i} \oplus P^d_{b,i} = \beta, \\ P^a_{b,i} \oplus P^c_{b,i} &= P^b_{b,i} \oplus P^d_{b,i} = \gamma, \\ P^a_{n,i} \oplus P^c_{n,i} &= P^b_{n,i} \oplus P^d_{n,i} = \delta, \end{aligned}$$

is called a correct related-key boomerang quartet *which occurs with probability* $Pr_c = (pq)^2$. *A quartet* $(P^a_{0,i}, P^b_{0,i}, P^c_{0,i}, P^d_{0,i})$ *which only satisfies the condition* $P^a_{0,i} \oplus P^b_{0,i} = \alpha = P^c_{0,i} \oplus P^d_{0,i}$ *is called a* false related-key boomerang quartet*.*

Figure 2 displays the structure of the related-key boomerang distinguisher step.

Any attacker that applies a related-key boomerang distinguisher does not know the internal states $P^a_{b,i}, P^b_{b,i}, P^c_{b,i}, P^d_{b,i}$, and since he can only apply a chosen plaintext and ciphertext attack on the cipher. The set Θ, which is the output of the related-key boomerang distinguisher, therefore contains correct and false related-key boomerang quartets. It is impossible to form another distinguisher which separates the correct and the false related-key boomerang quartets, since the interior differences β and γ cannot be computed.

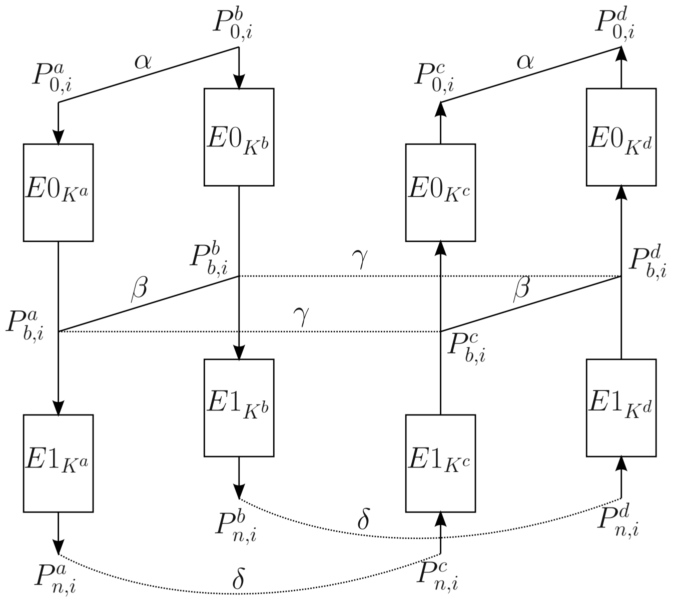


Fig. 2. The related-key boomerang distinguisher

Key Recovery Step. The second step of the related-key boomerang attack is the key recovery step. From now on, an attacker operates on the set Θ that was stored by the related-key boomerang distinguisher. Let $K^a_0, K^b_0, K^c_0, K^d_0$ be the 0-th round keys derived from the master keys K^a, K^b, K^c, K^d. Let $enc_{K^i_0}(P)$ be the one round partial encryption of P under the round key K^i_0, $i \in \{a, b, c, d\}$. The round keys are related as

$$\Delta K'_0 = K^a_0 \oplus K^b_0 = K^c_0 \oplus K^d_0,$$
$$\Delta K^*_0 = K^a_0 \oplus K^c_0 = K^b_0 \oplus K^d_0,$$

where ΔK^i_0, $i \in \{a, b, c, d\}$ is the key difference of the 0-th round keys. The key recovery step works as follows:

- For each key-bit combination of K^a_0, K^b_0, K^c_0 and K^d_0
  1. Initialize a counter with zero.
  - For all quartets $(P^a_{0,i}, P^b_{0,i}, P^c_{0,i}, P^d_{0,i})$ stored in Θ
    2. Encrypt the plaintext quartet $(P^a_{0,i}, P^b_{0,i}, P^c_{0,i}, P^d_{0,i})$ one round under the guessed round keys K^a_0, K^b_0, K^c_0 and K^d_0 respectively, i.e., $P^a_{1,i} = enc_{K^a_0}(P^a_{0,i}), P^b_{1,i} = enc_{K^b_0}(P^b_{0,i}), P^c_{1,i} = enc_{K^c_0}(P^c_{0,i})$ and $P^d_{1,i} = enc_{K^d_0}(P^d_{0,i})$.
    3. Test whether the differences $P^a_{1,i} \oplus P^b_{1,i}$ and $P^c_{1,i} \oplus P^d_{1,i}$ have a desired difference an attacker would expect depending on the related-key

differential being used. Increase a counter for the used key-bits if the difference is fulfilled in both pairs.

4. Output the round keys K_0^a, K_0^b, K_0^c and K_0^d with the highest counter as the correct one.

In Step 3, four cases can be distinguished, since Θ contains correct and false related-key boomerang quartets and the round keys K_0^a, K_0^b, K_0^c and K_0^d can either be correct or false. A correct related-key boomerang quartet encrypted using the correct round key will have the desired difference needed to pass the test in Step 3. Hence, the counter for the correct round key is increased.

The other three cases are: a correct related-key boomerang quartet is used with a false round key (Pr_{cK_f}), a false related-key boomerang quartet is used with a correct round key (Pr_{fK_c}) or a false related-key boomerang quartet is used with a false round key (Pr_{fK_f}). We assume that the cipher acts like a random permutation. In these case we can assume that

$$Pr_{cK_f} = Pr_{fK_c} = Pr_{fK_f} =: Pr_{\mathit{filter}}.$$

The probability that a quartet in one of the three undesirable cases is counted for a certain round key is Pr_{filter}. The related-key differentials have to be chosen such that the counter for the correct round key is significantly higher than the counter for each false round key. If the differentials have a high probability the key recovery step outputs the correct round key in Step 4 with a high probability much faster than exhaustive key search.

4 A Memoryless Related-Key Boomerang Attack on the Full Tiger Block Cipher

In this section, we propose a 22-round related-key boomerang distinguisher, which is used for our memoryless related-key boomerang attack on the full 24-round Tiger encryption mode. We make extensive use of the following property in our attack.

Property 1. Switching between an additive and an XOR difference holds with some probablitiy. e.g., if $X - Y = 2^i \bmod 2^{64}$, then $\Pr[X \oplus Y = 2^i] = 2^{-1}$. If $i = 63$, however, we have $\Pr[X \oplus Y = 2^i] = 1$. I.e., switching between the additive difference $I = 2^{63}$ and the XOR-difference I is for free.

4.1 A 22-Round Related-Key Boomerang Distinguisher

Let K be a master key which can be written as $K = x_0, x_1, \ldots, x_7$, where x_i is a 64-bit word. We use four different but related master keys K^a, K^b, K^c and K^d to mount our related-key rectangle attack on the full Tiger encryption mode. The master key differences are as follows:

$$\Delta K' = K^a \oplus K^b = K^c \oplus K^d = (I, I, 0, 0, 0, I, 0, 0),$$
$$\Delta K^* = K^a \oplus K^c = K^b \oplus K^d = (0, 0, 0, I, 0, 0, 0, I)$$

Since the key schedule of Tiger offers a high degree of linearity we can determine most of the round key differences derived from the master key differences $\Delta K'$ and ΔK^* respectively. Using the above key schedule we can derive the round key differences from the master key differences $\Delta K'$ and ΔK^*. The round key differences for $E0$ propagate as:[1]

$$(I, I, 0, 0, 0, I, 0, 0) \longrightarrow (0, 0, 0, 0, 0, I, 0, I) \longrightarrow (I, 0, I, 0, ?, ?, ?, ?)$$

The ? indicates an unknown value of a key difference. We obtain the round key differences for $E1$ as:[2]

$$(0, 0, 0, I, 0, 0, 0, I) \longrightarrow (0, I, 0, 0, 0, 0, 0, I) \longrightarrow (0, 0, 0, 0, 0, 0, 0, I)$$

For our attack we use an 11-round related-key differential from round 1 to 12 for $E0$ ($\alpha \rightarrow \beta$) using the master key difference $\Delta K'$. The related-key differential is:

$$(0, 0 \boxminus odd(I), I) \rightarrow (0, 0, 0)$$

The differential for $E0^{-1}$ ($\beta \rightarrow \alpha$) is the differential for $E0$ in reverse direction. The related-key differential $E0$ and $E0^{-1}$ are shown in Table 1. We exploit

Table 1. The Related-Key Differentials $E0$ and $E0^{-1}$ in reverse order

Round(i)	ΔA_i	ΔB_i	ΔC_i	Δk_i	Prob.
1	0	$0 \boxminus odd(I)$	I	–	2^{-32}
2	0	I	0	0	1
3	I	0	0	0	1
4	0	0	I	0	1
5	0	0	0	I	1
6	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮
12	0	0	0	0	–

another 11 rounds related-key differential for $E1^{-1}$ ($\delta \rightarrow \gamma$) that covers round 23 to 12 using the master key difference ΔK^*. The related-key differential is:

$$(odd(I), I, 0) \rightarrow (0, I, 0)$$

The related-key differential $E1^{-1}$ is shown in Table 2. Our 22-round related-key boomerang distinguisher holds with probability 2^{-128}, since we have $p = 2^{-32}$ and $q = 2^{-32}$ which leads to $(p \cdot q)^2 = 2^{-128}$.

[1] This related keys were also used in the attack on the Tiger encryption mode from [7].

[2] The same master key differential was used by pseudo-collision attack of Mendel and Rijmen [12].

Table 2. The Related-Key Differential $E1^{-1}$

Round(i)	ΔA_i	ΔB_i	ΔC_i	Δk_i	Prob.
23	$odd(I)$	I	0	–	2^{-32}
22	0	0	0	I	1
21	0	0	0	0	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
16	0	0	0	0	1
15	0	0	0	0	1
14	0	0	I	I	1
13	I	0	0	0	1
12	0	I	0	0	–

4.2 The Attack

The attack works as follows:

1. Guess two 64-bit round keys k_0^a, k_1^a and compute k_0^i, k_1^i, $i \in \{b, c, d\}$ using the known round key differences $\Delta K'$ and ΔK^*.
   - For $i = 1, 2, \ldots, 2^{129.5}$ do
     2. Set $A = odd(I)$. Choose a plaintext $P_{1,i}^a$ uniformly at random and compute $P_{1,i}^b = P_{1,i}^a \oplus \alpha$, where $\alpha = (0, 0 \boxminus A, I)$. Decrypt $P_{1,i}^a$ and $P_{1,i}^b$ under k_1^a, k_0^a and k_1^b, k_0^b respectively and obtain the plaintexts $P_{-1,i}^a$ and $P_{-1,i}^b$. With a chosen plaintext attack scenario, encrypt the plaintexts $P_{-1,i}^a$ and $P_{-1,i}^b$ under K^a and K^b respectively to obtain the ciphertexts $P_{23,i}^a$ and $P_{23,i}^b$.
     3. Compute the ciphertexts $P_{23,i}^c = P_{23,i}^a \oplus \delta$ and $P_{23,i}^d = P_{23,i}^b \oplus \delta$, $\delta = (A, I, 0)$. With a chosen ciphertext scenario, decrypt the ciphertexts $P_{23,i}^c, P_{23,i}^d$ under K^c and K^d respectively and obtain the plaintexts $P_{-1,i}^c$ and $P_{-1,i}^d$.
     4. Partially encrypt $P_{-1,i}^c, P_{-1,i}^d$ under k_0^c, k_1^c and k_0^d, k_1^d respectively and obtain $P_{1,i}^c$ and $P_{1,i}^d$. Check if $P_{1,i}^c \oplus P_{1,i}^d = \alpha$. If true, store the quartet $(P_{1,i}^a, P_{1,i}^b, P_{1,i}^c, P_{1,i}^d)$ in Θ.
5. If the number of quartets in Θ is at least two, record the round keys k_0^i, k_1^i $(i \in \{a, b, c, d\})$ and go to Step 6. Otherwise, go to Step 1 with another key candidates.
6. For a suggested (k_0^a, k_1^a), do an exhaustive search for the remaining $512 - 3 \cdot 64 = 384$ key bits by trial encryption. If a 512-bit key is suggested, output it as the master key K^a of the full Tiger encryption mode (as well as $K^b = K^a \oplus K'$, $K^c = K^a \oplus K^*$ and $K^d = K^a \oplus K' \oplus K^*$). Otherwise restart the algorithm.

4.3 Analysis of the Attack

From $\#Q = 2^{129.5}$ boomerang quartets we expect about $\#C = 2^{129.5} \cdot 2^{-128} \approx 3$ correct boomerang quartets in Θ, since the probability of the related-key boomerang distinguisher is 2^{-128}. A random permutation of a difference $P_{1,i}^c \oplus$

$P^d_{1,i}$ has difference α with probability $Pr_f = 2^{-192}$. We expect that $\#F = \#Q \cdot Pr_f = 2^{129.5} \cdot 2^{-192} = 2^{-62.5}$ false related-key boomerang quartets will pass the test in Step 4.

The data complexity of Step 1 to 4 is about $2^2 \cdot 2^{1.5} = 2^{3.5}$ chosen plain- and ciphertexts, since we expect to have only $2^{1.5}$ quartets stored in Θ that satisfy the condition in Step 4, not $2^{129.5}$ as one might respect since we do not have to store all the quartets. Step 1 will be applied 2^{128} times in the worst case. The time complexity of Step 2 is determined by two 24-round Tiger encryptions. The time complexity of Step 3 is negligible, while the time complexity of Step 4 is the same as for Step 3. Step 1 to 4 will be computed $2^{129.5}$ times. And thus the overall time complexity is bounded by $2^{128} \cdot 2^{129.5} \cdot 2^2 = 2^{259.5}$.

A false combination of quartets is counted in Step 6 with probability $Pr_{filter} = 2^{-2 \cdot 192}$. At least $\#cK_c \approx 3$ correct related-key boomerang quartets and additionally $\#fK_c = \#F \cdot Pr_{filter} = 2^{-62.5} \cdot 2^{-384} = 2^{-446.5}$ false related-key boomerang quartets are counted with the correct key bits. Note that a correct boomerang quartet is always counted with probability one for a correct round key. We double the filtering since we can filter on both pairs of a quartet. About $\#cK_c + \#fK_c \approx 3 + 2^{-446.5} \approx 3$ quartets are counted for the correct key bits.

About $\#cK_f = \#C \cdot Pr_{filter} \approx 3 \cdot 2^{-384} = 2^{-382.5}$ correct related-key boomerang quartets and $\#FK_f = \#F \cdot Pr_{filter} = 2^{-62.5} \cdot 2^{-384} = 2^{-446.5}$ false related-key boomerang quartets are counted with a false round key, which are approximately $\#cK_f + \#fK_f \approx 2^{-382.5} + 2^{-446.5} \approx 2^{-382.5}$ counts for each false round key.

Using the Poisson distribution we can compute the success rate of our attack. The probability that the number of remaining quartets for each false round key combination is larger than 1 is $Y \sim Poisson(\mu = 2^{-446.5})$, $\Pr(Y \geq 2) \approx 0$, since the expected number of quartets counted with a false round key is $2^{-446.5}$. We expect to have a count of at least 3 quartets for the correct key bits. The probability that the number of quartets counted for the correct round keys is at least 2 is $Z \sim Poisson(\mu = 3)$, $\Pr(Z \geq 2) \approx 0.8$. The data complexity of our attack is $2^{3.5}$ chosen plaintexts and ciphertexts, while the time complexity is about $2^{259.5}$ 24-round Tiger encryptions. The attack has a success rate of 0.8.

4.4 Further Improvements

The time complexity of our attack can be reduced using the following techniques. One can fix some of the plaintext bits that form an α difference. This will lead to a probability one differential for $E0$. The complexity can be reduced by a factor of about 2^{64} in this way. The time complexity of the improved attack is therefore bounded by $2^{195.5}$ 24-round Tiger encryptions.

5 Conclusion

In this paper we present the first cryptographic attack on the full 24-round Tiger encryption mode. Our related-key boomerang attack uses less memory, which is of about $2^{3.5}$ chosen plaintexts and ciphertexts. The time complexity of our attack is $2^{259.5}$ 24-round Tiger encryptions. Moreover we shown how time complexity of the attack can be reduced to $2^{195.5}$ encryptions by fixing some of the plaintext bits.

Our result shows that Tiger used as a block cipher can be distinguished from an ideal cipher. Nevertheless, we point out that Tiger used as a hash function offers some weaknesses, which should be considered.

References

[1] Anderson, R.J., Biham, E.: TIGER: A Fast New Hash Function. In: Gollmann [8], pp. 89–97

[2] Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. J. Cryptology 7(4), 229–246 (1994)

[3] Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)

[4] Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)

[5] Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)

[6] Damgård, I.: A Design Principle for Hash Functions. In: Brassard [5], pp. 416–427 (1989)

[7] Doganaksoy, A., Ozen, O., Varc, K.: On the Security of the Encryption Mode of Tiger (unpublished)

[8] Gollmann, D. (ed.): FSE 1996. LNCS, vol. 1039. Springer, Heidelberg (1996)

[9] Indesteege, S., Preneel, B.: Preimages for Reduced-Round Tiger (unpublished), http://www.cosic.esat.kuleuven.be/publications/article-930.ps

[10] Kelsey, J., Lucks, S.: Collisions and Near-Collisions for Reduced-Round Tiger. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 111–125. Springer, Heidelberg (2006)

[11] Mendel, F., Preneel, B., Rijmen, V., Yoshida, H., Watanabe, D.: Update on tiger. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 63–79. Springer, Heidelberg (2006)

[12] Mendel, F., Rijmen, V.: Cryptanalysis of the Tiger Hash Function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 536–550. Springer, Heidelberg (2007)

[13] Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [5], pp. 428–446 (1989)

[14] Schneier, B., Kelsey, J.: Unbalanced Feistel Networks and Block Cipher Design. In: Gollmann [8], pp. 121–144 (1996)

[15] Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

Memoryless Related-Key Boomerang Attack on 39-Round SHACAL-2

Ewan Fleischmann, Michael Gorski, and Stefan Lucks

Bauhaus-University Weimar, Germany
{Ewan.Fleischmann,Michael.Gorski,Stefan.Lucks}@uni-weimar.de

Abstract. SHACAL-2 is a 64-round block cipher based on the compression function of the hash function standard SHA-256. It has a 256-bit block size and a variable length key of up to 512 bits. Up to now, all attacks on more than 37 rounds require at least 2^{235} bytes of memory. Obviously such attacks will never become of practical interest due to this high amount of space. In this paper we adopt the relate-key boomerang attack and present the first memoryless attack on 39-round SHACAL-2. Our attack only employs $2^{8.5}$ bytes of memory and thus improves the data complexity of comparable attacks up to a factor of at least 2^{230}, which is a substantial improvement. We do not need to store all the data which gives this low data complexity. The related-key boomerang attack presented in this paper can also be seen as a starting point for more advanced attacks on SHACAL-2. The main advantage of our new attack is that we can proceed the data sequentially instead of parallel as needed for other attacks, which reduces the memory requirements dramatically.

Keywords: SHACAL-2, block cipher, differential cryptanalysis, related-key boomerang attack, memoryless attacks.

1 Introduction

SHACAL-2 [4] is a 256-bit block cipher with 512-bit keys which is based on the compression function of the hash function standard SHA-256 [15]. SHACAL-2 as well as SHACAL-1, which is based one SHA-1 [14], were submitted to the NESSIE (New European Schemes for Signatures, Integrity and Encryption) project [13] . SHACAL-2 was recommended to be one of the NESSIE projection selections, while SHACAL-1 was not selected, because of some concerns of the key schedule of SHACAL-1.

SHACAL-2 has 64-rounds and supports key lengths from 128 up to 512 bits. For key lengths below 512 bit, the key gets simply padded with zeros. The first cryptanalytic result on SHACAL-2 is an impossible differential attack [5] on a 30-round reduced version of SHACAL-2. A differential-nonlinear attack [16] and a square-nonlinear attack [16] were introduced in the following which are able to break up to 32 and 28 rounds. Including related keys leads to an improved attack of up to 37 rounds which was called the related-key differential-nonlinear attack [9]. The best cryptanalytic results one SHACAL-2 are the related-key rectangle attacks on 42 rounds by Lu et al. [12] and a 43-round attack by Wang [18]. The disadvantage of this attacks is the huge memory requirements which we will address in this paper. We present the first memoryless attack on SHACAL-2 which can break up to 39 rounds using two related keys. For our results we use the related-key boomerang attack, which improves the data complexity of comparable attacks by a factor of at least $2^{226.5}$. This new technique allows to proceed the data sequentially and thus reduces the memory requirements. This is also the main advantage of our related-key boomerang attack, since all other attacks have to store a huge amount of data so that the attack will work. Table 1 summarizes the results known from literature and our new results on SHACAL-2.

Table 1. Comparison of attacks on SHACAL-2

Attack	**# Rounds**	**Data**	**Time**	**Memory**	**Source**
Square-Nonlinear	28	$463 \cdot 2^{32}$ CP	$2^{494.1}$	$2^{45.9}$	[16]
Impossible Differential	30	744 CP	$2^{495.1}$	$2^{14.5}$	[5]
Differential-Nonlinear	32	$2^{43.4}$ CP	$2^{504.2}$	$2^{48.4}$	[16]
RK Differential-Nonlinear	35	$2^{42.32}$ RK-CP	$2^{452.10}$	$2^{47.32}$	[9]
RK Rectangle	37	$2^{235.16}$ RK-CP	$2^{486.95}$	$2^{240.16}$	[9]
RK Boomerang	39	$2^{3.5}$ RK-CPCC	$2^{483.5}$	$2^{8.5}$	Sec. 4
RK Rectangle	40	$2^{243.38}$ RK-CP	$2^{448.43}$	$2^{247.38}$	[12]
RK Rectangle	42	$2^{243.38}$ RK-CP	$2^{488.37}$	$2^{247.38}$	[12]
RK Rectangle†	43	$2^{240.38}$ RK-CP	$2^{480.4}$	$2^{245.38}$	[18]
RK Rectangle	44	2^{233} RK-CP	$2^{497.2}$	2^{238}	[11]

RK: Related-Key, CP: Chosen Plaintexts, CC: Chosen Ciphertexts.

† : The attack has a flaw pointed out in [11].

Boomerang Attack. The boomerang attack [17] is a strong extension to differential cryptanalysis for breaking more rounds than differential attacks can do, since the cipher is treated as a cascade of two sub-ciphers, using short differentials in each sub-cipher. These differentials are combined in an adaptive chosen plaintext and ciphertext attack to exploit properties of the cipher that have a high probability.

Related-Key Attack. The related-key attack [1, 10] applies differential cryptanalysis to the cipher with different, but related, keys and considers the information that can be extracted from encryptions using these keys. Ciphers with a weak key schedule are vulnerable to this kind of attack. The idea of related-key differentials was presented in [7], while two encryptions under two related-keys are used. Several combinations of related-key and differential attacks were introduced in the following.

Related-Key Boomerang and Rectangle Attack. The related-key boomerang and rectangle attack was published first in [2, 6, 8]. The related-key boomerang attack utilizes the fact, that the boomerang attack uses short differentials and so the lower diffusion in the differences of the round keys can be better exploited than in the original related-key differential attack. The attack combines features of the boomerang and the related-key attack.

Outline. In Section 2 we briefly describe the SHACAL-2 block cipher. In Section 3 we introduce the related-key boomerang attack, which is used in Section 4 to break 39 rounds of SHACAL-2 without using any noteworthy amount of memory. Finally, we conclude our paper in Section 5.

2 Description of SHACAL-2

SHACAL-2 is composed out of 64 rounds using a variable key length of at least 128 bits and up to 512 bits. For keys smaller than 512 bits zeros are padded until the key length reaches 512 bits. A 256-bit plaintext

$$P_0 = A_0||B_0||C_0||D_0||E_0||F_0||G_0||H_0$$

is divided into eight 32-bit words $A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0$. The corresponding ciphertext P_{64} is denoted by

$$P_{64} = A_{64}||B_{64}||C_{64}||D_{64}||E_{64}||F_{64}||G_{64}||H_{64}.$$

The following notations are used in this paper:

$\oplus$: bitwise XOR operation
$\wedge$: bitwise AND operation
$+$: addition modulo 2^{32} operation
$\neg$: bitwise complement operation
e_i : a 32-bit word with zeros in all positions except for bit i, $(0 \leq i \leq 31)$
$e_{i_1,\ldots,i_l}$: $e_{i_1} \oplus \cdots \oplus e_{i_l}$

The round function of round i can be described as follows:

$$\begin{aligned}
T^1_{i+1} &= K_i + \Sigma_1(E_i) + Ch(E_i, F_i, G_i) + H_i + W_i,\\
T^2_{i+1} &= \Sigma_0(A_i) + Maj(A_i, B_i, C_i),\\
H_{i+1} &= G_i,\\
G_{i+1} &= F_i,\\
F_{i+1} &= E_i,\\
E_{i+1} &= D_i + T^1_{i+1},\\
D_{i+1} &= C_i,\\
C_{i+1} &= B_i,\\
B_{i+1} &= A_i,\\
A_{i+1} &= T^1_{i+1} + T^2_{i+1},
\end{aligned}$$

where K_i is the i-th round key and W_i is the i-th round constant. The four functions in the encryption algorithm are defined as follows:

$$\begin{aligned}
Ch(X,Y,Z) &= (X \wedge Y) \oplus (\neg X \wedge Z),\\
Maj(X,Y,Z) &= (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z),\\
\Sigma_0(X) &= S_2(X) \oplus S_{13}(X) \oplus S_{22}(X),\\
\Sigma_1(X) &= S_6(X) \oplus S_{11}(X) \oplus S_{25}(X),
\end{aligned}$$

where $S_j(X)$ represents the right rotation of X by j bits. The bit positions of a 32-bit word are labeled as $31, 30, \ldots, 1, 0$, where bit 31 is the most significant bit and bit 0 is the least significant bit.

The key schedule algorithm of SHACAL-2 takes as input a 512-bit master key. As stated above, as many zeros as necessary will be padded to get a full 512-bit master key. The 512-bit master key K is divided into sixteen 32-bit words $K_0, K_1, \ldots, K_{15}$. These are the round keys for the first 16 rounds. The i-th round key $(16 \leq i \leq 63)$ is computed as

$$\begin{aligned}
K_i &= \sigma_1(K_{i-2}) + K_{i-7} + \sigma_0(K_{i-15}) + K_{i-16},\\
\sigma_0(X) &= S_7(X) \oplus S_{18}(X) \oplus R_3(X),\\
\sigma_1(X) &= S_{17}(X) \oplus S_{19}(X) \oplus R_{10}(X),
\end{aligned}$$

where $R_j(X)$ represents right shift of X by j bits.

In the following we present some basic properties of the $Ch(\cdot)$ and $Maj(\cdot)$ functions which will be needed in our attack.

Proposition 1. *(from [18]) For the nonlinear function $Ch(X,Y,Z) = (X \wedge Y) \oplus (\neg X \wedge Z)$, there are the following properties:*

1. *$Ch(x,y,z) = Ch(\neg x,y,z)$ if and only if $y = z$.*
   $Ch(0,y,z) = 0$ and $Ch(1,y,z) = 1$ if and only if $y = 1$ and $z = 0$.
   $Ch(0,y,z) = 1$ and $CH(1,y,z) = 0$ if and only if $y = 0$ and $z = 1$.
2. *$Ch(x,y,z) = Ch(x,\neg y,z)$ if and only if $x = 0$.*
   $Ch(x,0,z) = 0$ and $Ch(x,1,z) = 1$ if and only if $x = 1$.
3. *$Ch(x,y,z) = Ch(x,y,\neg z)$ if and only if $x = 1$.*
   $Ch(x,y,0) = 0$ and $Ch(x,y,1) = 1$ if and only if $x = 0$.

Proposition 2. *(from [18]) For the nonlinear function $Maj(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$, there are the following properties:*

1. *$Maj(x,y,z) = Maj(\neg x,y,z)$ if and only if $y = z$.*
   $Maj(0,y,z) = 0$ and $Maj(1,y,z) = 1$ if and only if $y = \neg z$.
2. *$Maj(x,y,z) = Maj(x,\neg y,z)$ if and only if $x = z$.*
   $Maj(x,0,z) = 0$ and $Maj(x,1,z) = 1$ if and only if $x = \neg z$.
3. *$Maj(x,y,z) = Maj(x,y,\neg z)$ if and only if $x = y$.*
   $Maj(x,y,0) = 0$ and $Maj(x,y,1) = 1$ if and only if $x = \neg y$.

3 The Related-Key Boomerang Attack

Recall that the related-key boomerang attack was first published in [2]. It is a combination of the related-key attack [1] and the boomerang attack [17].

Related-key attacks consider the information that can be extracted from encryptions using related but unknown keys. Such ciphers with a weak key schedule are vulnerable to this kind of attack. The key scheduling algorithms of many block ciphers inherit obvious relationships between keys, which are called related-keys.

The boomerang attack is an extension to differential cryptanalysis [3] using adaptive chosen plaintexts and ciphertexts to attack block ciphers. An attacker can only apply a chosen plain- and ciphertext attack while choosing the relation between related, but unknown, keys.

We now describe the related-key boomerang attack in more detail. But first, we have to give some definitions.

Definition 1. *Let P, P' be two bit strings of the same length. The bit-wise XOR of P and P', $P \oplus P'$, is called the* difference *of P, P'.*

Definition 2. *$\alpha \rightarrow \beta$ is called a* differential *if α is the plaintext difference $P \oplus P'$ before some non-linear operation $f(\cdot)$ and β is the difference after applying these operation, i.e, $f(P) \oplus f(P')$. The probability p is linked on a differential saying that an α difference turns into a β difference with probability p.*

Theorem 1. *([17, Sec. 6].) Let $\alpha \rightarrow \beta$ be a differential with probability p as in Definition 2. Then the backward direction of this differential $\beta \rightarrow \alpha$ has also probability p.*

Two texts (P, P') are called a *pair*, while two pairs (P, P', O, O') are called a *quartet*. Differential cryptanalysis exploits high probability differentials $\alpha \rightarrow \beta$ between the first and a later round of a cipher. In general, an attacker tries to find high probability differentials over the first $N-1$ out of N rounds to exploit some subkey bits of the last (i.e. N'th) round. An attacker first computes the expected output of the $N-1$-th round differential and then guesses some subkey bits of the last round. He decrypts some ciphertexts under the guessed key bits one round and compares the resulting difference with the output difference of the $N-1$ round differential. For every matching he increases a counter to the used key bits and takes the key bits with the highest counter as the 'correct' one. If the differential is well chosen, i.e., its probability is high enough, these subkey bits can be computed much faster than applying exhaustive search.

The related-key boomerang attack extends differential cryptanalysis. The cipher is split into two sub-ciphers and the attacker tries to exploit a differential for each sub-cipher. These two differentials alone only cover a few rounds but together the whole cipher. Regularly, the differential probability decreases the more rounds are included. Therefore two short differential covering only a few rounds each will be used instead of a long one covering the whole cipher. In this way key bits can be computed with much fewer plaintexts than compared to classic differential cryptanalysis, since the differential probabilities are normally higher. Related-keys are used to exploit some weaknesses of the key schedule to enhance the probability of the differentials being used. We call such differentials related-key differentials, since different but related keys are used for encryption and decryption. We split the related-key boomerang attack into two steps. The *related-key boomerang distinguisher step* and the *key recovery step*. The related-key boomerang distinguisher is used to find all plaintexts sharing a desired difference that depends on the choice of the differential. These plaintexts are used in the key recovery step afterwards to recover subkey bits for the initial round key.

Distinguisher Step. During the *distinguisher step* we assume that the cipher can be treated as a cascade of two sub-ciphers $E_K(P) = E1_K(E0_K(P))$, where K is the key used for encryption and decryption. We assume that the related-key differential $\alpha \rightarrow \beta$ for $E0$ occurs with the probability p, while the related-key differential $\gamma \rightarrow \delta$ for $E1$ occurs with probability q, where α, β, γ and δ are differences. The backward direction $E0^{-1}$ and $E1^{-1}$ of the related-key differential for $E0$ and $E1$ are denoted by $\alpha \leftarrow \beta$ and $\gamma \leftarrow \delta$ and occur, because of Theorem 1, with probability p and q respectively. The related-key boomerang distinguisher involves two different unknown – but related – keys $K, K^* = K \oplus \Delta K$, where ΔK is a known key differences. The attack works as follows:

- For $i = 1, 2, \ldots, s$ do
  1. Choose plaintext $P^1_{0,i}$ and plaintext $P^2_{0,i} = P^1_{0,i} \oplus \alpha$, where the subindex 0 defines the plaintext.
  2. Ask for the encryption of $P^1_{0,i}$ under K, i.e., $P^1_{n,i} = E_K(P^1_{0,i})$ and $P^2_{0,i}$ under K^*, i.e., $P^2_{n,i} = E_{K^*}(P^2_{0,i})$.
  3. Compute the new ciphertexts $P^3_{n,i} = P^1_{n,i} \oplus \delta$ and $P^4_{n,i} = P^2_{n,i} \oplus \delta$.
  4. Ask for decryption of $P^3_{n,i}$ under K, i.e., $P^3_{0,i} = E^{-1}_K(P^3_{n,i})$ and $P^4_{n,i}$ under K^*, i.e., $P^4_{0,i} = E^{-1}_{K^*}(P^4_{n,i})$.
  5. If $P^3_{0,i} \oplus P^4_{0,i} = \alpha$ store the quartet $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ in the set Θ.

Assume that a pair $(P^1_{0,i}, P^2_{0,i})$, $i \in \{1, \ldots, s\}$ with the difference α satisfies the differential $\alpha \rightarrow \beta$ with the probability p. Let $0 < b < n \leq N$ be an intermediate round, where N denotes the final round. Subcipher $E0$ covers rounds 1 to b and subcipber $E1$ rounds $b + 1$ to N. The output of $E0$ is $P^1_{b,i}$ and $P^2_{b,i}$, i.e., $P^1_{b,i} = E0_K(P^1_{0,i})$ and $P^2_{b,i} = E0_{K^*}(P^2_{0,i})$. $P^1_{n,i}$ and $P^2_{n,i}$ have a difference $\beta = P^1_{b,i} \oplus P^1_{b,i}$ with probability p. Using the ciphertexts $P^1_{n,i}$ and $P^2_{n,i}$ (encrypted via E_K and E_{K^*}) we can compute the new ciphertexts $P^3_{n,i} = P^1_{n,i} \oplus \delta$ and $P^4_{n,i} = P^2_{n,i} \oplus \delta$. Let $P^3_{b,i} = E1^{-1}_K(P^3_{n,i})$ and $P^4_{b,i} = E1^{-1}_{K^*}(P^4_{n,i})$ be the decryption of $P^3_{n,i}$ and $P^4_{n,i}$ under $E1$. A difference δ turns into a difference γ after passing $E1^{-1}$ with probability q. Since $\delta = P^1_{n,i} \oplus P^3_{n,i} = P^2_{n,i} \oplus P^4_{n,i}$ we know that $P^1_{b,i} \oplus P^3_{b,i} = \gamma$ and $P^2_{b,i} \oplus P^4_{b,i} = \gamma$ with probability q^2. As we know that $P^1_{b,i} \oplus P^2_{b,i} = \beta$ with probability p, it follows that $(P^3_{b,i} \oplus P^4_{b,i}) = (P^3_{b,i} \oplus P^1_{b,i}) \oplus (P^1_{b,i} \oplus P^2_{b,i}) \oplus (P^2_{b,i} \oplus P^4_{b,i}) = \gamma \oplus \beta \oplus \gamma = \beta$ holds with probability pq^2. A β difference turns into an α difference after passing the differential $E0^{-1}$ with probability p. Thus, a pair of ciphertexts $(P^1_{0,i}, P^2_{0,i})$ with $P^1_{0,i} \oplus P^2_{0,i} = \alpha$ generates a new pair of plaintexts $(P^3_{0,i}, P^4_{0,i})$ where $P^3_{0,i} \oplus P^4_{0,i} = \alpha$ with probability $(pq)^2$. A quartet containing these two pairs is defined as:

Definition 3. *A quartet* $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ *which satisfies*

$$\begin{aligned}
P^1_{0,i} \oplus P^2_{0,i} &= P^3_{0,i} \oplus P^4_{0,i} = \alpha,\\
P^1_{b,i} \oplus P^2_{b,i} &= P^3_{b,i} \oplus P^4_{b,i} = \beta,\\
P^1_{b,i} \oplus P^3_{b,i} &= P^2_{b,i} \oplus P^4_{b,i} = \gamma,\\
P^1_{n,i} \oplus P^3_{n,i} &= P^2_{n,i} \oplus P^4_{n,i} = \delta,
\end{aligned}$$

is called a correct related-key boomerang quartet *which occurs with probability* $Pr_c = (pq)^2$*. A quartet* $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ *which only satisfies the condition* $P^1_{0,i} \oplus ^2_{0,i} = \alpha = P^3_{0,i} \oplus P^4_{0,i}$ *is called a* false related-key boomerang quartet*.*

Figure 1 displays the structure of the related-key boomerang distinguisher step.

Any attacker that applies a related-key boomerang distinguisher does not know the internal states $P^1_{b,i}, P^2_{b,i}, P^3_{b,i}, P^4_{b,i}$, and since he can only apply a chosen plaintext and ciphertext attack on the cipher. The set Θ, which is the output of the related-key boomerang distinguisher, therefore contains correct and false related-key boomerang quartets. It is impossible to form another distinguisher which separates the correct and the false related-key boomerang quartets, since the interior differences β and γ cannot be computed.

Key Recovery Step. The second step of the related-key boomerang attack is the key recovery step. From now on, an attacker operates on the set Θ that was stored by the related-key boomerang distinguisher. Let K_0, K_0^* be the 0-th round keys derived from the master keys K, K^*. Let $enc_{K_0}(P)$ be the one round partial encryption of P under the round key K_0. The round keys are related as $K_0^* = K_0 \oplus \Delta K_0$, where ΔK_0 is the key difference of the 0-th round keys. The key recovery step works as follows:

- For each key-bit combination of K_0 and K_0^*
  1. Initialize a counter with zero.
  - For all quartets $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ stored in Θ
    2. Encrypt the plaintext quartet $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ one round under the guessed round keys K_0 and K_0^* respectively, i.e., $P^1_{1,i} = enc_{K_0}(P^1_{0,i}), P^2_{1,i} = enc_{K_0^*}(P^2_{0,i})$,
       $P^3_{1,i} = enc_{K_0}(P^3_{0,i})$ and $P^4_{1,i} = enc_{K_0^*}(P^4_{0,i})$.
    3. Test whether the differences $P^1_{1,i} \oplus P^2_{1,i}$ and $P^3_{1,i} \oplus P^4_{1,i}$ have a desired difference an attacker would expect depending on the related-key differential being used. Increase a counter for the used key-bits if the difference is fulfilled in both pairs.

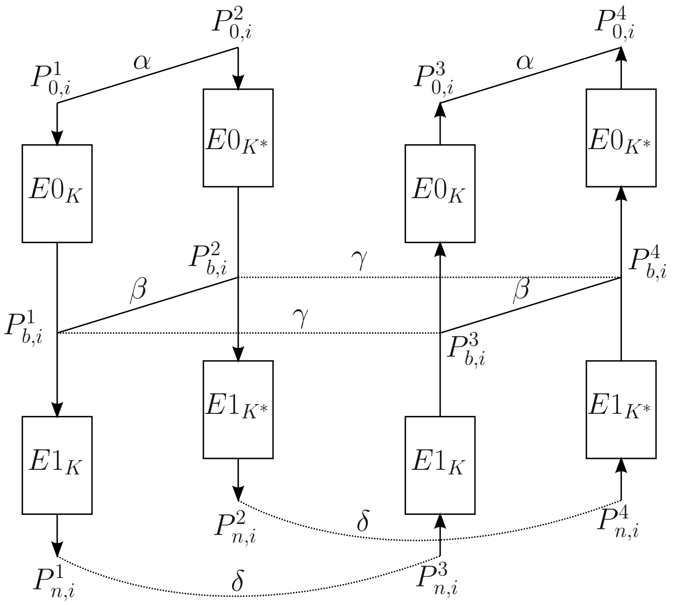


Fig. 1. The related-key boomerang distinguisher

4. Output the round keys K_0 and K_0^* with the highest counter as the correct one.

In Step 3, four cases can be distinguished, since Θ contains correct and false related-key boomerang quartets and the round keys K_0, K_0^* can either be correct or false. A correct related-key boomerang quartet encrypted using the correct round key will have the desired difference needed to pass the test in Step 3. Hence, the counter for the correct round key is increased.

The other three cases are: a correct related-key boomerang quartet is used with a false round key (Pr_{cK_f}), a false related-key boomerang quartet is used with a correct round key (Pr_{fK_c}) or a false related-key boomerang quartet is used with a false round key (Pr_{fK_f}). We assume that the cipher acts like a random permutation. In these case we can assume that

$$Pr_{cK_f} = Pr_{fK_c} = Pr_{fK_f} =: Pr_{\mathit{filter}}.$$

The probability that a quartet in one of the three undesirable cases is counted for a certain round key is Pr_{filter}. The related-key differentials have to be chosen such that the counter for the correct round key is significantly higher than the counter for each false round key. If the differentials have a high probability the key recovery step outputs the correct round key in Step 4 with a high probability much faster than exhaustive key search.

4 Memoryless Related-Key Boomerang Attack on 39-Round SHACAL-2

In this section we propose a 39-round memoryless related-key boomerang attack on SHACAL-2 using two related keys. The cipher E can be treated as a cascade of two sub-ciphers $E = E1 \circ E0$, $E0$ is a sub-cipher containing rounds 1 to 25 while the sub-cipher $E1$ covers rounds 25 to 35. Our related-key differentials are based on the differentials used in [18]. We use these differentials to build a 35-round related-key boomerang distinguisher, which can be used in a memoryless attack to break 39 rounds of SHACAL-2. The main advantage of our attack is that we do not need to store all the quartets as in the related-key rectangle attack of [9, 12, 18, 11]. Thus, we only require a very small amount of memory to successfully mount our attack.

The notations used in the attack will be as follows:

- K, K^* master keys (512 bit).
- K_i, K_i^* round keys of round i, where $i \in \{0, 1, 2, \ldots, 38\}$ (32 bit).
- ΔK is the master key difference, $\Delta K = (e_{31}, 0, 0, 0, 0, 0, 0, 0, 0, e_{31}, 0, 0, 0, 0, 0, 0)$.
- ΔK_i is the i-th round key difference derived from ΔK

In the following we will describe the related-key differentials used during our attack.

4.1 The Related-Key Differential $E0$ and $E0^{-1}$

The related-key differential for $E0$ covers rounds 1 to 25, which is

$$(e_{6,9,18,20,25,29}, e_{31}, 0, 0, e_{6,20,25}, e_{31}, 0, 0) \rightarrow (0, 0, e_M, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}),$$

where $M = \{6, 9, 18, 20, 25, 29\}$ and $\Sigma_1(E_0 \oplus e_{9,13,19}) - \Sigma_1(E_0) + \Delta_H = 0$, where Δ_H is a 32-bit word that is needed to satisfy the equation. The key schedule of SHACAL-2 has a low difference propagation for the first several rounds. Thus, if two master keys K and K^* are different in only two round keys in the first 16 rounds, i.e. $\Delta K_0 = e_{31}$ and $\Delta K_9 = e_{31}$, we get a zero difference up to ΔK_{23}. Wang [18] improves the 25-round differential characteristic introduced by Lu et al. [12] such that one does not have to guess the first round keys K_0 and K_0^*. This can be done by using Proposition 1 and Proposition 2 and fixing 16-bit conditions in the plaintexts to obtain a probability of 1 for the first round of the related-key differential. The bit conditions are presented in Table 2. The overall differential probability increases but due to the reduced costs of round key guessing one can improve a distinguisher using the new differential. We assume that the master keys K and K^* are related as $\Delta K = (e_{31}, 0, 0, 0, 0, 0, 0, 0, 0, e_{31}, 0, 0, 0, 0, 0, 0)$. The related-key differential $E0$ is shown in Table 3. Since we do not deal with truncated differentials,

Table 2. The fixed plaintext bits for SHACAL-2

$A^1_{0,i}$	$B^1_{0,i}$	$E^1_{0,i}$	$F^1_{0,i}$
$a^1_{31} = b^1_{31}$, $a^1_k = c^1_k$ $(k = 6, 9, 18, 20, 25, 29)$	$b^1_k = \neg f^1_k$ $(k = 19, 30)$ $b^1_9 = e^1_9$	$e^1_{31} = 0$, $e^1_k = 0$ $(k = 18, 29)$	$f^1_k = g^1_k$ $(k = 9, 13, 19)$

a^1_k, b^1_k, c^1_k, e^1_k, f^1_k, g^1_k are the k-th bits of $A^1_{0,i}, B^1_{0,i}, C^1_{0,i}, E^1_{0,i}, F^1_{0,i}, G^1_{0,i}$.

Table 3. The Related-Key Differential $E0$ and $E0^{-1}$ in reverse order

i	ΔA_i	ΔB_i	ΔC_i	ΔD_i	ΔE_i	ΔF_i	ΔG_i	ΔH_i	ΔK_i	Prob.
0	0	e_M	e_{31}	0	$e_{9,13,19}$	$e_{18,29}$	e_{31}	Δ_H	e_{31}	1
1	0	0	e_M	e_{31}	0	$e_{9,13,19}$	$e_{18,29}$	0	0	2^{-11}
2	e_{31}	0	0	e_M	0	0	$e_{9,13,19}$	$e_{18,29}$	0	2^{-10}
3	0	e_{31}	0	0	$e_{6,20,29}$	0	0	$e_{9,13,19}$	0	2^{-7}
4	0	0	e_{31}	0	0	$e_{6,20,25}$	0	0	0	2^{-4}
5	0	0	0	e_{31}	0	0	$e_{6,20,25}$	0	0	2^{-3}
6	0	0	0	0	e_{31}	0	0	$e_{6,20,25}$	0	2^{-4}
7	0	0	0	0	0	e_{31}	0	0	0	2^{-1}
8	0	0	0	0	0	0	e_{31}	0	0	2^{-1}
9	0	0	0	0	0	0	0	e_{31}	e_{31}	1
10	0	0	0	0	0	0	0	0	0	1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
23	0	0	0	0	0	0	0	0	0	1
24	0	0	0	0	0	0	0	0	$\cdot$	2^{-6}
25	$e_{13,24,28}$	0	0	0	$e_{13,24,28}$	0	0	0	$\cdot$	$-$

$M = \{6, 9, 18, 20, 25, 29\}$, $\Sigma_1(E_0 \oplus e_{9,13,19}) - \Sigma_1(E_0) + \Delta_H = 0$.

we can assume that the probability for a differential remains the same if it runs in backward direction (Theorem 1). Thus, $\Pr(\alpha \to \beta) = \Pr(\alpha \leftarrow \beta)$ always holds for the related-key differential $E0$ and $E^{0^{-1}}$. Thus, Table 3 also represents the related-key differential $E0^{-1}$, which is starts from the bottom of the table (round 25) and goes up to the top (round 1). The related-key differential $E0$ occurs with probability $\Pr(\alpha \to \beta) = 2^{-47}$ and therefore the related-key differential $E0^{-1}$ occurs also with probability $\Pr(\alpha \leftarrow \beta) = 2^{-47}$ too.

4.2 The Related-Key Differential $E1^{-1}$

Our related key differential $E1^{-1}$ covers rounds 35 to 25, which can be written as

$$(e_{6,9,18,20,25,29}, e_{31}, 0, 0, e_{6,20,25}, e_{31}, 0, 0) \to (e_{31}, e_{31}, e_{M'}, 0, 0, e_{9,13,19}, e_{18,29,31}, 0).$$

The related-key differential $E1^{-1}$ occurs with probability $\Pr(\gamma \leftarrow \delta) = 2^{-65}$ and can be found in Table 4.

Table 4. The Related-Key Differential $E1^{-1}$

i	ΔA_i	ΔB_i	ΔC_i	ΔD_i	ΔE_i	ΔF_i	ΔG_i	ΔH_i	Prob.
35	$e_{6,9,18,20,25,29}$	e_{31}	0	0	$e_{6,20,25}$	e_{31}	0	0	2^{-11}
34	e_{31}	0	0	0	e_{31}	0	0	0	1
33	0	0	0	0	0	0	0	e_{31}	2^{-1}
32	0	0	0	0	0	0	e_{31}	e_{31}	1
31	0	0	0	0	0	e_{31}	e_{31}	e_{31}	2^{-4}
30	0	0	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	2^{-7}
29	0	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	2^{-8}
28	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	0	2^{-7}
27	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	0	$e_{9,13,19}$	2^{-12}
26	e_{31}	e_{31}	e_{31}	$e_{M'}$	0	0	$e_{9,13,19}$	$e_{18,29,31}$	2^{-15}
25	e_{31}	e_{31}	$e_{M'}$	0	0	$e_{9,13,19}$	$e_{18,29,31}$	0	–

$M' = \{6, 9, 18, 20, 25, 29, 31\}$.

4.3 The Attack

The attack works as follows:

- For $i = 1, 2, \ldots, 2^{225.5}$ do
  - Guess two 128-bit round keys $(K_{38}, K_{37}, K_{36}, K_{35})$ and $(K^*_{38}, K^*_{37}, K^*_{36}, K^*_{35})$ and do the following:
    1. Choose plaintext $P^1_{0,i}$ and ΔH_i, such that the 16 bit conditions presented in Table 2 and the condition $\Sigma_1(E_0 \oplus e_{9,13,19}) - \Sigma_1(E_0) + \Delta H_i = 0$ are satisfied. Ask for encryption of $P^1_{0,i}$ under K to obtain the plaintext $P^1_{39,i}$.
    2. Compute the intermediate value $P^1_{35,i}$ by decrypting $P^1_{39,i}$ under $(K_{38}, K_{37}, K_{36}, K_{35})$.
    3. Compute the intermediate value $P^3_{35,i} = P^1_{35,i} \oplus \delta$, where $\delta = (e_{6,9,18,20,25,29}, e_{31}, 0, 0, e_{6,20,25}, e_{31}, 0, 0)$. Compute $P^3_{39,i}$ by encrypting $P^3_{35,i}$ using $(K_{35}, K_{36}, K_{37}, K_{38})$.
    4. Ask for decryption of $P^3_{39,i}$ under K and obtain $P^3_{0,i}$.
    5. Now, use the previously chosen $P^1_{0,i}$ and the difference $\alpha_i = (0, e_M, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}, \Delta H_i)$, $M = \{6, 9, 18, 20, 25, 29\}$, to compute $P^2_{0,i} = P^1_{0,i} \oplus \alpha_i$. Ask for encryption of $P^2_{0,i}$ under K^* to obtain $P^2_{39,i}$.
    6. Compute the intermediate value $P^2_{35,i}$ by decrypting $P^2_{39,i}$ under $(K^*_{38}, K^*_{37}, K^*_{36}, K^*_{35})$ respectively.
    7. Compute the intermediate value $P^4_{35,i} = P^2_{35,i} \oplus \delta$. Compute $P^4_{39,i}$ by encrypting $P^4_{35,i}$ with keys $(K^*_{35}, K^*_{36}, K^*_{37}, K^*_{38})$.
    8. Ask for decryption of $P^4_{39,i}$ under K^* to obtain $P^4_{0,i}$.
    9. Check if $P^3_{0,i} \oplus P^4_{0,i} = \alpha_i$. If true, store the quartet $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ in Θ as well as the used partial keys $K_0, K_{35}, K_{36}, K_{37}, K_{38}$.
- For all 64-bit round keys (K_0, K^*_0) do

10. Set $counter = 0$.
   - For all quartets $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ in Θ
     11. Encrypt $(P^1_{0,i}, P^2_{0,i}, P^3_{0,i}, P^4_{0,i})$ one round with K_0 and K_0^* respectively.
     12. Check if $P^1_{1,i} \oplus P^2_{1,i} = P^3_{1,i} \oplus P^4_{1,i} = \tau$, where $\tau = (0, 0, e_M, e_{31}, 0, e_{9,13,19}, e_{18,29}, 0)$. If true, increase the $counter$ by one.
     13. Take the guess of K_0 and K_0^* with $counter \geq 2$ as the correct round key pair.
14. For a suggested $(K_0, K_{35}, K_{36}, K_{37}, K_{38})$ (as also saved in Θ for every quartet), do an exhaustive search for the remaining $512 - 5 \cdot 32 = 352$ key bits by trial encryption. If a 512-bit key is suggested, output it as the master key of the 39-round SHACAL-2. Otherwise restart the algorithm.

4.4 Analysis of the Attack

A correct related-key boomerang quartet occurs with probability

$$\begin{aligned} Pr_c &= \Pr(\alpha \to \beta) \cdot (\Pr(\gamma \leftarrow \delta))^2 \cdot \Pr(\alpha \leftarrow \beta) \\ &= 2^{-47} \cdot (2^{-65})^2 \cdot 2^{-47} = 2^{-224}, \end{aligned}$$

since all related-key differential conditions are fulfilled. A random permutation of a difference $P^3_{0,i} \oplus P^4_{0,i}$ has difference α with probability $Pr_f = 2^{-256}$. Let $\#Q = 2^{225.5}$ be the number of initially chosen quartets and $\#C$ be the number of correct quartets as tested in Step 9. We expect that $\#C = \#Q \cdot Pr_c = 2^{225.5} \cdot 2^{-224} = 2^{1.5} \approx 3$ correct quartets are found and that $\#F = \#Q \cdot Pr_f = 2^{225.5} \cdot 2^{-256} = 2^{-30.5}$ false quartets are found. The data complexity of Step 1 to 9 is about $2^2 \cdot 2^{1.5} = 2^{3.5}$ chosen plain– and ciphertexts, since we expect to have about $\#C = 2^{1.5}$ correct quartets and $\#F = 2^{-30.5}$ false quartets stored in Θ. This amount occurs since we have a 256-bit filtering condition on one side of the boomerang. The time complexity is determined by $2^2 \cdot 2^{225.5} \cdot 2^{8 \cdot 32} = 2^{483.5}$ 39-round SHACAL-2 encryptions.

A false combination of quartets and key bits is counted in Step 12 with probability $Pr_{\mathit{filter}} = 2^{-2 \cdot 32}$. At least $\#cK_c \approx 3$ correct related-key boomerang quartets and additionally $\#fK_c = \#F \cdot Pr_{\mathit{filter}} = 2^{-30.5} \cdot 2^{-128} = 2^{-158.5}$ false related-key boomerang quartets are counted with the correct key bits. We double the filtering since we can filter on both pairs of a quartet. About $\#cK_c + \#fK_c \approx 3 + 2^{-158.5} \approx 3$ quartets are counted for the correct key bits. Note that we need only $\Theta \approx 2^{1.5}$ quartets for our key recovery step, since Θ contains only correct quartets. This means that the correct key bits alway get a count of $2^{1.5}$ which will distinguish them from all false key bit combinations.

About $\#cK_f = \#C \cdot Pr_{\mathit{filter}} \approx 3 \cdot 2^{-64} = 2^{-62.5}$ correct related-key boomerang quartets and $\#FK_f = \#F \cdot Pr_{\mathit{filter}} = 2^{-30.5} \cdot 2^{-64} = 2^{-94.5}$ false related-key boomerang quartets are counted with the false key bits, which are approximately $\#cK_f + \#fK_f \approx 2^{-62.5} + 2^{-94.5} \approx 2^{-62.5}$ counts for each false key bit combination. Step 13 takes the correct key bits since we expect about $2^{-62.5}$ counts for each false key bit combination and about 3 counts for the correct key bits. The time complexity of Steps 10 to 13 is negligible small under these circumstances. ($\approx 2^{64}$).

Using the Poisson distribution we can compute the success rate of our attack. The probability that the number of remaining quartets for each false key bit combination is larger than 1 is $Y \sim \mathit{Poisson}(\mu = 2^{-62.5})$, $\Pr(Y \geq 2) \approx 0$, since the expected number of quartets counted with a false key bit combination is $2^{-62.5}$. We expect to have a count of at least 3 quartets for the correct key bits. The probability that the number of quartets counted for the correct key bits is at least 2 is $Z \sim \mathit{Poisson}(\mu = 3)$, $\Pr(Z \geq 2) \approx 0.8$. The data complexity of our attack is $2^{3.5}$ chosen plaintexts and ciphertexts, while the time complexity time complexity is about $2^{483.5}$ 39-round SHACAL-2 encryptions. Our attack has a success rate of 0.8.

5 Conclusion

In this paper we present the first memoryless attack on SHACAL-2. Using a related-key boomerang distinguisher we can mount an attack on 39-round SHACAL-2. This improves the memory requirements of existing attacks by a factor of about 2^{235}. Furthermore our attack can be seen as a starting point/building block to form new attacks on SHACAL-2 using the related-key boomerang technique. This is the first application of the related-key boomerang attack used as a memoryless variant on a block cipher. The main advantage of our attack is that the data can be proceeded sequentially instead of parallel as needed for all the other attacks.

Further research. An improvement of our attack would be the following scenario: Instead of using one differential for $E0$ and $E1$ one could try to use all possible differentials which have the same input and output difference. This will decrease the time complexity but does not lead to attack more rounds. Another improvement might be to use more than two related keys which may give somewhat better differentials.

Acknowledgements

The authors wish to thank the anonymous reviewers for helpful comments.

References

[1] Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. J. Cryptology 7(4), 229–246 (1994)

[2] Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)

[3] Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
[4] Handschuh, H., Naccache, D.: SHACAL: A Family of Block Ciphers. Submission to the NESSIE project (2002), http://www.cosic.esat.kuleuven.be/nessie/tweaks.html
[5] Hong, S.H., Kim, J.-S., Kim, G., Sung, J., Lee, C.-H., Lee, S.-J.: Impossible Differential Attack on 30-Round SHACAL-2. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 97–106. Springer, Heidelberg (2003)
[6] Hong, S.H., Kim, J.-S., Lee, S.-J., Preneel, B.: Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 368–383. Springer, Heidelberg (2005)
[7] Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 233–246. Springer, Heidelberg (1997)
[8] Kim, J., Kim, G., Hong, S., Lee, S., Hong, D.: The Related-Key Rectangle Attack - Application to SHACAL-1. In: Wang, et al. (eds.) [19], pp. 123–136 (2004)
[9] Kim, J.-S., Kim, G., Lee, S.-J., Lim, J.-I., Song, J.: Related-Key Attacks on Reduced Rounds of SHACAL-2. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 175–190. Springer, Heidelberg (2004)
[10] Knudsen, L.R.: Cryptanalysis of LOKI91. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 196–208. Springer, Heidelberg (1993)
[11] Lu, J., Kim, J.: Attacking 44 rounds of the shacal-2 block cipher using related-key rectangle cryptanalysis. IEICE Transactions 91-A(9), 2588–2596 (2008)
[12] Lu, J., Kim, J.-S., Keller, N., Dunkelman, O.: Related-Key Rectangle Attack on 42-Round SHACAL-2. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 85–100. Springer, Heidelberg (2006)
[13] NESSIE, http://www.cosic.esat.kuleuven.be/nessie/
[14] National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard (April 1995), http://csrc.nist.gov
[15] National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard (August 2002), http://csrc.nist.gov
[16] Shin, Y., Kim, J., Kim, G., Hong, S., Lee, S.: Differential-Linear Type Attacks on Reduced Rounds of SHACAL-2. In: Wang, et al. (eds.) [19], pp. 110–122 (2004)
[17] Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
[18] Wang, G.: Related-Key Rectangle Attack on 43-Round SHACAL-2. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 33–42. Springer, Heidelberg (2007)
[19] Wang, H., Pieprzyk, J., Varadharajan, V. (eds.): ACISP 2004. LNCS, vol. 3108. Springer, Heidelberg (2004)

Some New Observations on the SMS4 Block Cipher in the Chinese WAPI Standard

Wentao Zhang, Wenling Wu, Dengguo Feng, and Bozhan Su

State Key Laboratory of Information Security,
Institute of Software, Chinese Academy of Sciences, Beijing 100190, P. R. China
zhangwt06@yahoo.com,
{wwl,feng,subozhan}@is.iscas.ac.cn

Abstract. SMS4 is a 128-bit block cipher used in the WAPI standard in wireless networks in China. The cipher has attracted much attention in the past two years. This paper consists of two parts. The first part is on the design of the linear diffusion layer L of SMS4. Some new observations on L are present, which open out the design rationales of L and such class functions to a great extent. The second part is on the differential attack against SMS4. A class of 18-round differential characteristics with a higher probability is given. Then a simple differential attack on 22-round SMS4 is present, which is an improvement of the previous work, thus our attack becomes the best known one on SMS4. Furthermore, we make a remark on the construction of differential characteristics of SMS4.

Keywords: WAPI, Block Cipher, SMS4, Diffusion Transformation, Differential Cryptanalysis.

1 Introduction

WAPI (**W**LAN **A**uthentication and **P**rivacy **I**nfrastructure) is the Chinese national standard for WLANs (**W**ireless **L**ocal **A**rea **N**etworks) products. SMS4 is the underlying block cipher used in the WAPI standard. WAPI has been submitted to ISO for recognition as an international standard, but rejected in March 2006, partially because of uncertainties regarding the security of the unrevealed SMS4 cipher. However, the WAPI standard continues to be widely used in Chinese industries.

Chinese government released the SMS4 [1] cipher in January 2006 in a Chinese version, and [3] presented an English translation of [1]. SMS4 has a 128-bit block size, a 128-bit user key, and a total of 32 rounds. It employs a kind of generalized Feistel network structure, and only 32 bits are modified in each round. The main part of its round function (i.e., T function) is composed of 3 steps, firstly the subkey XOR-addition, next 4 S-boxes in parallel, lastly a linear transformation L.

Since its introduction, the SMS4 cipher has attracted much attention from cryptanalytic researchers, due to its simplicity and Chinese standard prominence. There are several public papers on SMS4 in the past two years. The first open analysis [10] uncovers the origin of its S-box, and presents an integral attack on 13-round SMS4. Next, some properties on a class of linear transformations based on cyclic-shift and Xor operations are presented [12], the results can be directly applied to the diffusion layer of SMS4[1]. Then, a rectangle attack on 14 rounds and an impossible differential attack on 16 rounds are presented [11]. Later, a differential attack on 21 rounds [14] and a linear attack on 22 rounds [5] are respectively revealed. Recently, a linear attack and a differential attack both on 22-round SMS4 [9] are presented based on the work of [14]. Other papers on SMS4 include [4,7,15]. The previous best known attack are due to [5] and [9].

This paper consists of two parts. The first part is on the design of the linear transformation L of SMS4. L is very simple, composed of only 5 left rotations and 4 XORs. For all we know, the analogous modules only appeared in the hash standard SHA-256, namely σ_i and Σ_i $(i = 0, 1)$ functions of SHA-256, which are all composed of 3 right rotations (or right shifts) and 2 XORs. In the following, we will present 3 observations on the linear transformation L of SMS4, which provides some insight into the design rationales of L and such class functions.

The second part is on the differential cryptanalysis of SMS4. We will show that there are 12 18-round differential characteristics for SMS4 each with a higher probability of 2^{-114}, which are the best differential characteristics for SMS4 so far. Then we present a simple differential attack on 22-round SMS4 based on any one of the 12 differential characteristics, which improves the previous best known attacks due to J. Etrog and M.Robshaw[5] and T.Kim et.al.[9] with regard to attack complexity, thus our attack becomes the best one so far. Finally, we make a remark on constructing differential characteristics of SMS4.

2 Description of SMS4

SMS4 is a 32-round block cipher with a 128-bit block size and a 128-bit key size. Its overall structure is a kind of generalized Feistel network, thus the encryption and decryption have the same procedure except that the round subkeys for decryption are used in the reverse order.

2.1 Notation

The following notations are used throughout this paper.
- Z_2^{32}: the set of 32-bit words; Z_2^8: the set of 8-bit bytes; $Z_2 = \{0, 1\}$.
- $\lll i$: left rotation by i bits; $\ggg i$: right rotation by i bits.
- For any $U \in Z_2^{32}$, let $U = (U_0, U_1, U_2, U_3) \in (Z_2^8)^4$.
- K_i: the 32-bit subkey in Round i, $K_i = (K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3}) \in (Z_2^8)^4$, $(i = 0, 1, \ldots, 31)$.
- $Prob_F(\alpha \to \beta)$: the probability that output difference of function F is β given input difference is α, F can be omitted when the context is clear.

[1] Actually, Theorem 3.1 in this paper can be directly obtained from Theorem 1 of [12]. We know this after we have obtained Theorem 3.1 and Theorem 3.2. Whereas the proof of Theorem 1 is not presented in [12], we remain Theorem 3.1 and its proof in this paper.

2.2 Encryption Procedure of SMS4

Let $(X_0, X_1, X_2, X_3) \in (Z_2^{32})^4$ and $(Y_0, Y_1, Y_2, Y_3) \in (Z_2^{32})^4$ denote the 128-bit plaintext and ciphertext respectively. Let $K_i \in Z_2^{32}$ $(i = 0, 1, 2, \dots 31)$ denote the round subkeys.

The encryption procedure of SMS4 is as follows:

$$X_{i+4} = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus K_i), \quad \text{for} \quad i = 0, 1, \dots, 31$$

$$(Y_0, Y_1, Y_2, Y_3) = (X_{35}, X_{34}, X_{33}, X_{32}).$$

where the function $T : Z_2^{32} \mapsto Z_2^{32}$ is composed of a non-linear transformation τ and a linear transformation L, i.e., $T(\cdot) = L(\tau(\cdot))$.

Fig.1 depicts one round of the encryption procedure of SMS4.

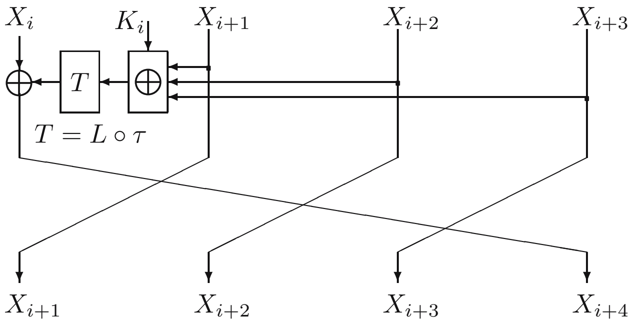


Fig. 1. The i-th round of SMS4

τ applies 4 S-boxes in parallel. Let $A = (a_0, a_1, a_2, a_3) \in (Z_2^8)^4$ be the input, where each a_i is a byte. Then the output $B = \tau(A) = (b_0, b_1, b_2, b_3) \in (Z_2^8)^4$ is given by

$$(b_0, b_1, b_2, b_3) = (Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3))$$

where $Sbox$ is a 8×8 S-box.

The 32-bit output B of τ will be the input of the linear transformation L. The output $C = L(B) \in (Z_2)^{32}$ is given by

$$C = L(B) = B \oplus (B \lll 2) \oplus (B \lll 10) \oplus (B \lll 18) \oplus (B \lll 24).$$

3 Observations on the Linear Transformation L

In this section, we will present 3 observations on the linear transformation L.

3.1 Optimality of L

Branch number introduced by J.Daemen [2] has now become a widely-accepted concept for measuring the diffusion effect of a transformation (usually a linear transformation).

The following gives the definition of the branch number of L.

Definition 3.1. (Branch number) Let $W(\cdot)$ denote the byte weight function, i.e., the number of nonzero bytes. The branch number of a function $F : Z_2^{32} \to Z_2^{32}$ is defined by

$$\beta(F) = \min_{X \neq 0, X \in Z_2^{32}} (W(X) + W(F(X))).$$

It is easy to know that $\beta(F) \leqslant 5$. As a rule, the larger the branch number, the better its diffusion effect. When $\beta(F)$ reaches the maximum 5, we call F maximal.

For the linear transformation L of SMS4, the branch number reaches the maximum 5, which can be easily verified by a computer program.

For a diffusion transformation of a block cipher or a hash function, the implementation is another important factor besides its branch number. L is composed of 5 left-rotations and 4 XORs. The question posed here is : can we use less left-rotations and XORs to get a transformation whose branch number is also the best possible 5? In the following, we will show that the answer is "NO".

Let $F(X, k, r_1, \ldots, r_k) = \bigoplus_{i=1}^{k} (X \lll r_i)$, where $X \in Z_2^{32}$, k is a natural number and $0 \leqslant r_1 < r_2 < \ldots < r_{k-1} < r_k \leqslant 31$. That is, $F(X, k, r_1, \ldots, r_k)$ is composed of k left rotations and $(k-1)$ XORs, and the rotations are specified by $(r_1, \ldots, r_k)$.

Theorem 3.1. If $F(X, k, r_1, \ldots, r_k)$ is maximal for some value of $(k, r_1, \ldots, r_k)$, then $k \geqslant 5$.

Proof. Let $X = (x_{31}, x_{30}, \ldots, x_1, x_0) \in Z_2^{32}$, its bit index $31, 30, \ldots, 1, 0$ can be divided into 4 subsections of one byte each: $31 \ldots 24$, $23 \ldots 16$, $15 \ldots 8$ and $7 \ldots 0$, we call them byte-index subsections.

Let $X \in Z_2^{32}$ and $Y = X \lll r$, then the ith bit of Y is $y_i = x_{(i-r) mod\ 32}$ $(i = 0, 1, \ldots, 31)$. Let e_i be the 32-bit string whose i-th bit equals to 1 and the other 31 bits all equal to 0, then we can get the value of $F(e_i, k, r_1, \ldots, r_k)$: the $(i+r_1)$th, $(i+r_2)$th, $\ldots$, $(i+r_k)$th bits equal to 1 and the other $(32-k)$ bits equal to 0. Therefore, if $F(X, k, r_1, \ldots, r_k)$ is maximal, then it must require $W(F(e_i, k, r_1, \ldots, r_k)) = 4$ as $W(e_i) = 1$. That is to say, for each of the 4 byte-index subsections, at least one element of $\{r_1, \ldots, r_k\}$ is falling into it (when considering e_0), which also means $k \geqslant 4$.

On the other hand, it's easy to see that

$$F(0xffffffff, k, r_1, \ldots, r_k) = \begin{cases} 0xffffffff & \text{if } k \text{ is odd} \\ 0 & \text{if } k \text{ is even} \end{cases}$$

Thus, the branch number of $F(X, k, r_1, \ldots, r_k)$ is not more than 4 if k is even.

Therefore, it must require $k \geqslant 5$ for a maximal $F(X, k, r_1, \ldots, r_k)$. $\square$

According to Theorem 3.1, we can know that the selection of the linear transformation L of SMS4 is optimal between the branch number and the number of rotations. Furthermore, we will show that there are few maximal $F(X, k, r_1, \ldots, r_k)$ when $k = 5$.

Let $F(X, k, r_1, \ldots, r_k) \lll h$ denote the function $(\bigoplus_{i=1}^{k}(X \lll r_i)) \lll h$, it is easy to see that:

$$F(X, k, r_1, \ldots, r_k) \lll h = \bigoplus_{i=1}^{k}(X \lll (r_i + h))$$

where "+" denotes addition modulo 32.

Firstly, we give the following property:

Property 3.1. For each value of k and each value of $(r_1, \ldots, r_k)$, the branch number of $F(X, k, r_1, \ldots, r_k) \lll 8$, $F(X, k, r_1, \ldots, r_k) \lll 16$ and $F(X, k, r_1, \ldots, r_k) \lll 24$ are all equal to the branch number of $F(X, k, r_1, \ldots, r_k)$.

The reason of Property 3.1 is that the output of each of the three functions is just a byte - permutation of the output of $F(X, k, r_1, \ldots, r_k)$ given the same input X, so the branch number is unchanged.

When $k = 5$, there are 5 parameters r_1, r_2, r_3, r_4, r_5. If $F(X, 5, r_1, \ldots, r_5)$ is maximal, then it must be that 2 parameters lie in a same byte-index subsection, and the other 3 parameters lie in the other three byte-index subsections each.

According to Property 3.1, we will assume:

$0 \leqslant r_1 < r_2 \leqslant 7, \quad 8 \leqslant r_3 \leqslant 15, \quad 16 \leqslant r_4 \leqslant 23, \quad 24 \leqslant r_5 \leqslant 31$

Thus, there are $\frac{8\times7}{2} \times 8 \times 8 \times 8 = 14336$ different transformations $F(X, 5, r_1, \ldots, r_5)$ in all in the sense of Property 3.1. An experiment is made to test which are maximal among all the 14336 transformations, which spent several days on two computers. The result is as follows:

Theorem 3.2. Among all of the 14336 transformations $F(X, 5, r_1, \ldots, r_5)$, only two are maximal, one is L, the other is :

$L^*(X) = X \oplus (X \lll 6) \oplus (X \lll 14) \oplus (X \lll 22) \oplus (X \lll 24)$

$L^*(X)$ and $L(X)$ are related in the following way: substitute "$\ggg$" for "$\lll$" in $L^*(X)$, we have

$$\begin{aligned} R^*(X) &= X \oplus (X \ggg 6) \oplus (X \ggg 14) \oplus (X \ggg 22) \oplus (X \ggg 24) \\ &= L(X) \lll 24 \end{aligned}$$

It's easy to know $R^*(X)$ and $L^*(X)$ have the same branch number. Also $L(X)$ and $L(X) \lll 24$ have the same branch number. Hence, our experimental result shows that the linear transformation L of SMS4 is the unique maximal one in essence among all the transformations $F(X, 5, r_1, \ldots, r_5)$, in view of Property 3.1 and diffusion effect.

3.2 Bijectivity of $F(X, k, r_1, \ldots, r_k)$ Function

In [6], the authors proved that Σ_i $(i = 0, 1)$ functions of SHA-256 are all one to one, their proof need the specific parameters of the two functions. In the following, we will present a more general result, which makes the above result in [6] a simple corollary.

Theorem 3.3 Let X be a 32-bit variable. For any $(r_1, \ldots, r_k)$, satisfying $0 \leqslant r_1 < r_2 < \ldots < r_{k-1} < r_k \leqslant 31$, $F(X, k, r_1, \ldots, r_k)$ is bijective if k is odd, and non-bijective if k is even.

Proof. Let $f^{\,n}(\cdot)$ denote the n-th iteration of $f(\cdot)$, i.e., $f^{\,n+1} = f \circ f^{\,n}$, where $f \circ g$ denotes function composition, that is, $(f \circ g)(x) = f(g(x))$.

We have

$$\begin{aligned} F^{\,2}(X, k, r_1, \ldots, r_k) &= \bigoplus_{j=1}^{k}((\bigoplus_{i=1}^{k}(X \lll r_i)) \lll r_j) = \bigoplus_{i,j=1}^{k}(X \lll (r_i + r_j)) \\ &= \bigoplus_{i=1}^{k}(X \lll (2 \times r_i)) \end{aligned} \tag{1}$$

where "+" denotes addition modulo 32 and "×" multiplication modulo 32.

Because X is a 32-bit variable, it is easy to see that $X \lll 32 = X$ holds. Then, we can get the following equation by repeatedly using equation (1) 5 times:

$$F^{\,32}(X, k, r_1, \ldots, r_k) = \bigoplus_{i=1}^{k}(X \lll (32 \times r_i)) = \bigoplus_{i=1}^{k} X = \begin{cases} X & \text{if } k \text{ is odd} \\ 0 & \text{if } k \text{ is even} \end{cases}$$

Hence, if k is odd, then the inverse function of $F(X, k, r_1, \ldots, r_k)$ is $F^{31}(X, k, r_1, \ldots, r_k)$, thus F function must be a bijection. If k is even, then F must be non-bijective. □

Theorem 3.3 suggests that k must be odd when $F(X, k, r_1, \ldots, r_k)$ is used as a function of cryptographic ciphers, such as block ciphers and hash functions.

3.3 Distribution of Input-Output Patterns of *L*

Let x be a byte, define the function $\delta(x)$ as:

$$\delta(x) = \begin{cases} 1 \text{ if } x \neq 0 \\ 0 \text{ if } x = 0 \end{cases}$$

Let $X = (x_0, x_1, x_2, x_3) \in (Z_2^8)^4$, the pattern of X is defined by $Pattern(X) = (\delta(x_0), \delta(x_1), \delta(x_2), \delta(x_3))$, i.e., $Pattern(X)$ specifies the active and passive byte positions of X. It's easy to see that there are 15 possible patterns for a non-zero X.

Table 1 gives the distribution of input pattern and output pattern of $Y = L(X)$, where the first column denotes the 15 possible input patterns of X, the first row denotes the 15 output patterns of Y, the element at position $(i, j)(i, j \in \{0001, 0010, \ldots 1111\})$ denotes the probability of $Pattern(Y) = j$ given $Pattern(X) = i$, where

$$\begin{aligned} &\varepsilon_1 = \tfrac{1}{255}, \varepsilon_2 = \tfrac{251}{255}, \\ &\varepsilon_3 = \tfrac{1}{65025}, \varepsilon_4 = \tfrac{251}{65025}, \varepsilon_5 = \tfrac{64015}{65025}, \\ &\varepsilon_6 = \tfrac{1}{16581375}, \varepsilon_7 = \tfrac{251}{16581375}, \varepsilon_8 = \tfrac{64015}{16581375}, \varepsilon_9 = \tfrac{16323805}{16581375}. \end{aligned}$$

Table 1. Distribution of Input-Output Patterns of L

	0001	0010	0100	1000	0011	0101	0110	1001	1010	1100	0111	1011	1101	1110	1111
0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0011	0	0	0	0	0	0	0	0	0	0	ε_1	ε_1	ε_1	ε_1	ε_2
0101	0	0	0	0	0	0	0	0	0	0	ε_1	ε_1	ε_1	ε_1	ε_2
0110	0	0	0	0	0	0	0	0	0	0	ε_1	ε_1	ε_1	ε_1	ε_2
1001	0	0	0	0	0	0	0	0	0	0	ε_1	ε_1	ε_1	ε_1	ε_2
1010	0	0	0	0	0	0	0	0	0	0	ε_1	ε_1	ε_1	ε_1	ε_2
1100	0	0	0	0	0	0	0	0	0	0	ε_1	ε_1	ε_1	ε_1	ε_2
0111	0	0	0	0	ε_3	ε_3	ε_3	ε_3	ε_3	ε_3	ε_4	ε_4	ε_4	ε_4	ε_5
1011	0	0	0	0	ε_3	ε_3	ε_3	ε_3	ε_3	ε_3	ε_4	ε_4	ε_4	ε_4	ε_5
1101	0	0	0	0	ε_3	ε_3	ε_3	ε_3	ε_3	ε_3	ε_4	ε_4	ε_4	ε_4	ε_5
1110	0	0	0	0	ε_3	ε_3	ε_3	ε_3	ε_3	ε_3	ε_4	ε_4	ε_4	ε_4	ε_5
1111	ε_6	ε_6	ε_6	ε_6	ε_7	ε_7	ε_7	ε_7	ε_7	ε_7	ε_8	ε_8	ε_8	ε_8	ε_9

The concept of branch number only roughly reflect the diffusion effect of a transformation. Whereas a distribution table as Table 1 reflect the diffusion effect more comprehensively. Such a distribution table may be very helpful in analyzing a cipher against differential, truncated differential or linear cryptanalysis,etc. For example, Zhang etc. [14] have used the pattern of $Pattern(X) = Pattern(Y) = (0, 1, 1, 1)$ in their differential attack on 21-round SMS4.

Phan [13] has presented the distribution of input-output patterns of the Mix-Column transformation of AES, we have also experimentally verified this distribution. It's interesting to find that the distribution of input-output patterns of L of SMS4 is the same as that of the MixColumn transformation of AES. Hence, we can say that the diffusion effect of L is the same with that of the MixColumn transformation of AES.

4 A Simple Differential Attack on 22-Round SMS4

In [14], a class of 5-round iterative differential characteristics of SMS4 (the amount is 7905) is presented, with an average probability of 2^{-42} each. Then, a class of 18-round differential characteristics are easily derived by simply iterating the 5-round iterative differentials, with an average probability of 2^{-126} each. Eventually, the authors present a differential attack on 21-round SMS4, which uses all of the 7905 18-round differential characteristics. It's worth noting that there are a few 5-round iterative differential characteristics with a higher probability of 2^{-38} each, as the authors of [14] have also pointed out and used one to construct a 14-round rectangle distinguisher.

In this section, we will show that there are 3 out of 7905 5-round differential characteristics, which have a probability of 2^{-38} each. Then 18-round differential characteristics are easily derived, with a probability of 2^{-114} each. Using any of these 3 18-round differential characteristics, we can present a simple differential attack on 22-round SMS4.

4.1 18-Round Differential Characteristics with a Probability of 2^{-114} Each

The 5-round iterative differential characteristics are as follows [14]:

$$(\alpha, \alpha, \alpha, 0) \xrightarrow{5\ Rounds} (\alpha, \alpha, \alpha, 0)$$

with a probability of $(Prob_T(\alpha \to \alpha))^2$.

Because the branch number of L is 5, so it require that $H(\alpha) \geqslant 3$ in order for $Prob_T(\alpha \to \alpha) \neq 0$. It's obvious that $Prob_T(\alpha \to \alpha)$ is larger when $H(\alpha) = 3$. In [14], the authors show that there are 7905 candidates for α when $Pattern(\alpha) = (0, 1, 1, 1)$. One can refer to [14] for more details.

Through an experiment, we find that there are only 3 out of the 7905 candidates for α, which make $Prob_T(\alpha \to \alpha)$ reach the maximal value 2^{-19}. The three values are: $0x002cf5cd$, $0x00d2c822$ and $00c30290$. Furthermore, let Y be one of the three values, i.e., $Y \in \{0x002cf5cd,\ 0x00d2c822,\ 00c30290\}$. Then, when $\alpha = (Y \lll 8)$ or $\alpha = (Y \lll 16)$ or $\alpha = (Y \lll 24)$, the corresponding 5-round iterative differential characteristic also holds with a probability of 2^{-38}, which is also verified by a computer program.

Iterating a 5-round differential characteristic three and a half times, then a 18-round differential characteristic is derived. Hence, there are 12 18-round differentials in total, with a probability of $2^{-38*3} = 2^{-114}$ each, which are the best distinguishers for SMS4 so far.

4.2 A Differential Attack on 22-Round SMS4

In this section, we fix α as any of the three values: $0x002cf5cd$, $0x00d2c822$ or $00c30290$. Then, we have the following 18-round differential characteristic:

$$(\alpha, \alpha, \alpha, 0) \xrightarrow{18\ Rounds} (0, \alpha, \alpha, \alpha)$$

with a probability of 2^{-114}.

Attack Procedure. We use Round 0, Round1, ..., Round 21 to number the rounds of the 22-round SMS4. Applying the above 18-round differential at Rounds $0 \sim 17$. If the output difference of Round 17 is $(0, \alpha, \alpha, \alpha)$, then the input difference of T function in Round 18 equals to α. For the S-box of SMS4, there are 127 possible output differences for any nonzero input difference, thus the output difference of T in Round 18 has only about $2^{7*3} = 2^{21}$ possible values. Let Λ denote the set of these 2^{21} possible values, then the output difference of Round 18 must belong to $(\alpha, \alpha, \alpha, \Lambda)$. Similarly, the output difference of Round 19 must belong to $(\alpha, \alpha, \Lambda, ?)$, the output difference of Round 20 must belong to $(\alpha, \Lambda, ?, ?)$, and the output difference of Round 21 must belong to $(\Lambda, ?, ?, ?)$, where ? denotes an unknown word.

The attack procedure is as follows.

1. Let (P, P^*) denote a plaintext pair, and (C, C^*) the corresponding ciphertext pair. Select 2^m plaintext pairs, satisfying $P \bigoplus P^* = (\alpha, \alpha, \alpha, 0)$ for each pair.
2. For each ciphertext pair (C, C^*), check if the first word of the ciphertext difference belongs to the set Λ. If not, discard the pair. Store the filtered ciphertext pairs in a table, about 2^{m-11} ciphertext pairs are expected to be stored after this test.
3. For every guess of the 0th byte of K_{21}, i.e., $K_{21,0}$, do as follows:
   (a) For each remaining ciphertext pair (C, C^*), let $C = (C_0, C_1, C_2, C_3) \in (Z_2^{32})^4$, $C^* = (C_0^*, C_1^*, C_2^*, C_3^*) \in (Z_2^{32})^4$, partially decrypt to get the output difference of the 0th Sbox in Round 21: $\gamma = Sbox((C_0 \oplus C_1 \oplus C_2)_0 \oplus K_{21,0}) \oplus Sbox((C_0^* \oplus C_1^* \oplus C_2^*)_0 \oplus K_{21,0})$, and compute $\delta = (L^{-1}(C_3 \oplus C_3^* \oplus \alpha))_0$. If the guess of $K_{21,0}$ is correct, then it must be $\gamma = \delta$ for right pairs. If this is not the case, discard the pair. About 2^{m-19} ciphertext pairs are expected to remain after this test.
   (b) For each of the other 3 bytes of K_{21}, the 4 bytes of K_{20} and the 4 bytes of K_{19}, continue making a guess, then compute γ and δ with the guessed subkey byte. Similar to step 3.a, discard the disqualified ciphertext pairs. About 2^{m-107} ciphertext pairs are expected to remain after these tests.
   (c) Guess $K_{18,0}$, do similarly to step 3.a. But a probability of about $\frac{1}{2}$ pairs are already discarded because of the selection of the set Λ, hence about $2^{m-107-7} = 2^{m-114}$ ciphertext pairs are expected to remain after this test.
   (d) For each of the other three bytes of K_{18}, make a guess and do similarly to step 3.a. About 2^{m-135} ciphertext pairs are expected to remain after these tests. If $m = 116$, thus the expectation of the remaining ciphertext pairs for a wrong key guess is about $2^{116-135} = 2^{-19}$, and the expectation of the remaining ciphertext pairs for the right key guess is about $2^{116-114} = 2^2$. Hence, if the number of the remaining ciphertext pairs is larger than 2, then keep the guess of $K_{21}, K_{20}, K_{19}, K_{18}$ as the candidates of the right subkeys.
   (e) For the survived candidates of the right subkeys in step 3.c, compute the seed key K using the key schedule of SMS4. Then, do an exhaustive search to find the unique right seed key.

The data complexity of the above attack is about 2^{117} chosen plaintexts, and the memory complexity is about 2^{110} bytes.

By the Poisson distribution, $X \sim Poi(\lambda = 2^{-19})$, $Prob[X > 2] \approx 2^{-59.5}$, thus the expectation of subkey candidates suggested in step 3.e is about $2^{128-59.5} = 2^{68.5}$. By the Poisson distribution, $X \sim Poi(\lambda = 2^2)$, $Prob[X > 2] \approx 76.2\%$, so the success rate of the above attack is about 76.2%.

The time complexity is dominated by step 3.a - 3.d. In step 3.a, 2^{105} ciphertext pairs are treated with 2^8 subkey candidates for $K_{21,0}$, so the time complexity is about $2^{105+8+1} \times \frac{1}{22} \times \frac{1}{4} \approx 2^{107.54}$ 22-round SMS4 encryptions. Similarly, the time complexity of step 3.b is about $2^{107.54} \times 11$, the time complexity of step 3.c is about $2^{107.54}$, and the time complexity of step 3.d is about

Table 2. Comparison of Some Previous Attacks with Our New Attack

Source	Number of Rounds	Data Complexity	Time Complexity	Attack Type
Ref.[10]	13	2^{16} CP	2^{114}	Integral
Ref.[11]*	14	$2^{121.82}$ CP	$2^{116.66}$	Rectangle
	16	2^{105}CP	2^{107}	Imp.Diff
Ref.[4]	14	$2^{106.89}$ CP	$2^{87.97}$	Rectangle
	16	$2^{117.06}$CP	$2^{95.09}$	Imp.Diff
Ref.[14]	16	2^{125} CP	2^{116}	Rectangle
	21	2^{118}CP	$2^{126.6}$	Differential
Ref.[5]	22	$2^{118.4}$ KP	2^{117}	Linear
Ref.[9]	18	2^{120} ACPS	$2^{116.83}$	Boomerang
	18	2^{124}CP	2^{128}	Rectangle
	22	2^{117}KP	$2^{109.86}Enc. + 2^{120.39}A.O.$	Linear
	22	2^{118}CP	2^{123}	Differential
This paper	22	2^{117} KP	$2^{112.3}$	Differential

CP – Chosen plaintext, KP – Known plaintext.
ACPS – Adaptive Chosen Plaintext and Ciphertext.
A.O. – Arithmetic operation.
Time complexity is measured in encryption units (Enc.).

$(2^{2+112+1} + 2^{-5+120+1} + 2^{-12+128+1}) \times \frac{1}{22} \times \frac{1}{4}$. Hence, the total time complexity is about $2^{112.3}$ 22-round SMS4 encryptions.

We summarize our attack along with the previously known ones against SMS4 in Table 2. Notice that our attack is based on the work of [14] and [9]. The key point is that we exploit a single differential characteristic with a much higher probability, which also makes the attack more effective and more clear. Whereas a class of 7905 differential characteristics are exploited in [14] and [9], with a much lower average probability.

5 A Remark on Constructing Differential Characteristics of SMS4

Zhang etc. [14] present the following 5-round iterative differential characteristics

$$(\alpha, \alpha, \alpha, 0) \xrightarrow{5\ Rounds} (\alpha, \alpha, \alpha, 0)$$

In fact, let $\Delta = \{(\chi_0, \chi_1, \chi_2, \chi_3) | \chi_i = 0 \ or \ \chi_i = \alpha \ for \ each \ 0 \leqslant i \leqslant 3\}$. For each of the non-zero elements $(\chi_0, \chi_1, \chi_2, \chi_3) \in \Delta$, we can get the following 5-round iterative differential characteristic by restricting the output difference of T function is α when its input difference is α:

$$(\chi_0, \chi_1, \chi_2, \chi_3) \xrightarrow{5\ Rounds} (\chi_0, \chi_1, \chi_2, \chi_3)$$

There are 15 non-zero elements in Δ, thus 15 different 5-round iterative differential characteristics. Through a calculation, we find there are 10 differentials with a probability of $(Prob_T(\alpha \to \alpha))^2$ each, the other 5 with a lower probability.

For the 10 differentials, there are two best ones when we consider iterating them for more rounds, corresponding respectively to $(\chi_0, \chi_1, \chi_2, \chi_3) = (\alpha, \alpha, \alpha, 0)$ and $(\chi_0, \chi_1, \chi_2, \chi_3) = (0, \alpha, \alpha, \alpha)$. This is because the first 3-round differential holds with probability 1 when $(\chi_0, \chi_1, \chi_2, \chi_3) = (\alpha, \alpha, \alpha, 0)$, and the last 3-round differential holds with probability 1 when $(\chi_0, \chi_1, \chi_2, \chi_3) = (0, \alpha, \alpha, \alpha)$, thus we can add 3 rounds for free (one in the end, the other in the beginning).

Moreover, it is easy to show that there exists no 4-round differential characteristic with probability 1.

We have tried to construct other differential characteristics of SMS4, but no more than 18 rounds, and the 18-round differentials presented in section 4.1 are the best.

In [8], the authors present the upper bounds of the maximum differential and linear characteristic probabilities of Camellia, which would inspire a similar study on SMS4. But the overall structure of SMS4 makes the study so complex, the workload is so huge, we can't achieve this goal at present.

6 Summary

In the first part, we present 3 observations on the design of the linear transformation L of SMS4. Firstly, we prove that L is optimal between the branch number and the number of rotations, furthermore we show there are essentially only two transformations whose branch number reaches the maximum 5 if using only 5 rotations and 4 XORs. Secondly, we present a sufficient and necessary condition for such class functions to be bijective. Thirdly, we show that L has the same diffusion effect as the MixColumn transformation of AES, by comparing their distributions of input-output patterns. These observations reveal the design rationales of L and such class functions to a great extent, and also helpful in analyzing the security of ciphers using such functions.

In the second part, firstly we show that there are 12 18-round differential characteristics for SMS4, each with a higher probability of 2^{-114}, which are the best differential characteristics for SMS4 so far. Then, we present a simple differential attack on 22-round SMS4. Compared with the attack complexity of the 22-round attacks in [5,9], ours is better, hence our attack is the best known one on SMS4 in the literature so far. Furthermore, we make a remark on the construction of differential characteristics of SMS4, which shows that the class of 18-round differential characteristics presented in section 4.1 (the amount is 12) with a probability of 2^{-114} each maybe the best effective ones. Finally, we stress that the full 32-round SMS4 provides a sufficient safety margin against differential cryptanalysis, since the best attack can only reach 22 rounds so far.

Acknowledgment

We would like to thank anonymous referees for their helpful comments and suggestions. The research presented in this paper is supported by the National Natural Science Foundation of China (No.60873259); the National Basic Research 973 Program of China (No.2007CB311202); the National High-Tech Research and Development 863 Plan of China (No.2007AA01Z470).

References

1. Specification of SMS4, Block Cipher for WLAN Products – SMS4 (in Chinese), http://www.oscca.gov.cn/UpFile/200621016423197990.pdf
2. Daemen, J.: Cipher and Hash Function Design Strategies Based on Linear and Differential Cryptanalysis, Doctoral Dissertation, K.U.Leuven (March 1995)
3. Diffie, W., Ledin, G. (translators): SMS4 Encryption Algorithm for Wireless Networks Cryptology ePrint Archive, report 2008/329, received (July 29, 2008), http://eprint.iacr.org/
4. Dunkelman, O., Toz, D.: Analysis of the Attacking Reduced-Round Versions of the SMS4. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 141–156. Springer, Heidelberg (2008)
5. Etrog, J., Robshaw, M.J.B.: The Cryptanalysis of Reduced-Round SMS4. In: Proceedings of SAC 2008 (2008)
6. Gilbert, H., Handschuh, H.: Security Analysis of SHA-256 and Sisters. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 175–193. Springer, Heidelberg (2004)
7. Ji, W., Hu, L.: New description of SMS4 by an embedding overGF(2^8). In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 238–251. Springer, Heidelberg (2007)
8. Kanda, M.: Practical security evaluation against differential and linear attacks for Feistel ciphers with SPN round function. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 168–179. Springer, Heidelberg (2001)
9. Kim, T., Kim, J., Hong, S., Sun, J.: Linear and Diffrential Cryptanalysis of Reduced SMS4 Block Cipher, Cryptology ePrint Archive, report 2008/281, http://eprint.iacr.org/
10. Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R.-P.: Analysis of the SMS4 block cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007)
11. Lu, J.: Attacking reduced-round versions of the SMS4 block cipher in the chinese WAPI standard. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 306–318. Springer, Heidelberg (2007)
12. Wang, J.B.: The Optimal Permutation in Cryptography Based on Cyclic - Shift Linear Transform. In: ChinaCrypt 2007, pp. 306–307 (2007) (in Chinese)
13. Phan, R.C.W., Siddiqi, M.U.: Generalized Impossible Differentials of Advanced Encryption Standard. Electronics Letters 37(14), 896–898 (2001)
14. Zhang, L., Zhang, W.T., Wu, W.L.: Cryptanalysis of reduced-round SMS4 block cipher. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 216–229. Springer, Heidelberg (2008)
15. Zhang, L., Wu, W.L.: Differential fault attack on SMS4. Chinese Journal of Computers 29(9) (2006) (in Chinese)

On the Correctness of an Approach against Side-Channel Attacks

Peng Wang[1], Dengguo Feng[2], Wenling Wu[2], and Liting Zhang[2]

[1] State Key Laboratory of Information Security
Graduate University of Chinese Academy of Sciences, Beijing 100049, China
wp@is.ac.cn

[2] State Key Laboratory of Information Security
Institution of Software of Chinese Academy of Sciences, Beijing 100080, China
{feng,wwl,zhangliting}@is.iscas.ac.cn

Abstract. Side-channel attacks are a very powerful cryptanalytic technique. Li and Gu [ProvSec'07] proposed an approach against side-channel attacks, which states that a symmetric encryption scheme is IND-secure in side-channel model, if it is IND-secure in black-box model and there is no adversary who can recover the whole key of the scheme computationally in side-channel model, i.e. WKR-SCA $\wedge$ IND $\rightarrow$ IND-SCA. Our researches show that it is not the case. We analyze notions of security against key recovery attacks and security against distinguishing attacks, and then construct a scheme which is WKR-SCA-secure and IND-secure, but not IND-SCA-secure in the same side-channel environment. Furthermore, even if the scheme is secure again partial key recovery attacks in side-channel model, this approach still does not hold true.

Keywords: Provable security, Side-channel attack, Symmetric encryption.

1 Introduction

In traditional cryptanalysis, an adversary has only black-box access to cryptographic algorithms, i.e. the adversary can query the keyed cryptographic algorithm with input of its choice and get the corresponding output, but it can not get any other information of what's going on during the computation of the output. Unfortunately, in physical implementations, this kind of information sometimes can be easily obtained, such as timing information [8], power consumption [7], electromagnetic leakage [4], etc. We call the attacks based on this leaked information side-channel attacks. The researches in the last decade show that side-channel attacks are a very powerful cryptanalytic technique. The security of some cryptographic algorithms may collapse suddenly given some tiny side-channel information, even though they are very secure in traditional cryptanalysis.

The black-box model illustrates a theoretical world in which we focus on design in the level of algorithm, on the other hand the side-channel model illustrates a practical world in which we have to face the leaked information in implementations. We have millions of experiences on designing secure cryptographic algorithms in black-box model. *But how to guarantee the security of the algorithms in side-channel model?*

Let's first look at some notions of security. The minimal security requirement is the privacy of the secret key. Key recovery attacks are often used to analyze the security of cryptographic primitives such as block cipher both in black-box model [2,3] and side-channel model [6]. Recently the researches on how to establish a unified framework to evaluate the implementation security also focus on key recovery attacks [10,11,12]. If no adversary can recover the whole key computationally, we say it is WKR-secure (in black-box mode) or WKR-SCA-secure (in side-channel model)[1]. If no adversary can recover any part of key computationally, we say it is PKR-secure (in black-box mode) or PKR-SCA-secure (in side-channel model).

But as to a concrete cryptographic scheme, we need a corresponding security notion. For example, an encryption scheme requires the privacy of the plaintext, i.e. any adversary can not learn any information of the plaintext (except the length) computationally given a challenge ciphertext. This notion was firstly defined by Goldwasser and Micali as semantic security [5], which is equivalent to indistinguishability of the ciphertexts [5,1]. If no information about the plaintext (except the length) is revealed computationally by the ciphertext, we say the scheme is IND-secure (in black-box mode) or IND-SCA-secure (in side-channel model).

Li and Gu proposed an approach against side-channel attacks in ProvSec 2007 [9], which states that if a symmetric encryption scheme is both WKR-SCA-secure and IND-secure, then it is IND-SCA-secure. In other words, in order to guarantee the practically security of the scheme, we only need to guarantee the theoretically security of the scheme and there is no adversary who can recover the whole key of the scheme computationally in the practical world.

Unfortunately, it is not the case.

Our results are based on the following two basic observations:

– The notion of WKR (WKR-SCA) is much weaker than the notion of IND (IND-SCA).
– The security of the scheme in side-channel model is closely related to the leaked information.

Our Contributions. We first analyze the relations among PKR, WKR and IND both in black-box model and side-channel model. Please see Figure 1 and Figure 2, which show that WKR is the weakest notion, and PKR and IND are incomparable. We give proofs for all the implications of the notions and construct concrete examples for all the separations of the notions.

We then explore the security of notion combination. Our results show that WKR-SCA $\wedge$ IND does not imply IND-SCA. Given a symmetric encryption scheme which is IND-secure, we can construct a scheme which is both

[1] The notion of WKR is the same as the notion of UB in [9], which means "unbreakability of the key".

IND-secure and WKR-SCA-secure, but not IND-SCA-secure in the same side-channel environment.

Furthermore, we show that even the scheme is PKR-SCA-secure and IND-secure, this approach still does not hold. Based on a symmetric encryption scheme which is both PKR-secure and IND-secure, we construct a new scheme which is both PKR-SCA-secure and IND-secure, but not IND-SCA-secure in the same side-channel environment.

2 Preliminaries

Notations. We write $s \xleftarrow{\$} S$ to denote choosing a random element s from a set S by uniform distribution. An *adversary* is an (randomized) algorithm with access to one or more oracles which are written as superscripts. We write the adversary $\mathbf{A}$ with oracle $\mathcal{O}$ outputing a bit b as $\mathbf{A}^{\mathcal{O}} \Rightarrow b$. $\mathbf{Adv}^{\mathrm{GGG}}_{SSS}(\mathbf{A})$ denotes the advantage of $\mathbf{A}$ attacking a scheme “SSS” with a goal of “GGG”.

$\mathsf{A} \to \mathsf{B}$ means any scheme meeting notion A also meets notion B and $\mathsf{B} \not\to \mathsf{A}$ means there exists a scheme meeting notion B but do not meets notion A.

Symmetric Encryption Scheme. A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. The randomized *key generation algorithm* $\mathcal{K}$ generates a key K, denoted as $K \leftarrow \mathcal{K}$. The randomized or stateful *encryption algorithm* $\mathcal{E}$ takes the key K and a plaintext M to return a ciphertext C, denoted as $C \leftarrow \mathcal{E}_K(M)$. The deterministic and sateless *decryption algorithm* $\mathcal{D}$ takes the key K and a string C to return the corresponding plaintext M or the symbol $\perp$, denoted as $x \leftarrow \mathcal{D}$, where $x \in \{0,1\}^* \cup \{\perp\}$. We require that $\mathcal{D}_K(\mathcal{E}_K(M)) = M$ for any plaintext M.

In this paper, we focus on the security of symmetric encryption scheme under chosen plaintext attacks.

Side-Channel Leakage Function. In side-channel attacks, the adversary not only can query the encryption oracle $\mathcal{E}_K$ and get the corresponding ciphertext, but also can get some side-channel information during the computation of the ciphertext. We notice that the side-channel information is relevant to the key K and the queried plaintext M, so we treat it as a function $L(K, M)$ and call it a leakage function. We define a new oracle $\mathcal{E}^+_K(M) = (\mathcal{E}_K(M), L(K, M))$ which returns both the ciphertext and the leaked information. Therefore in side-channel attacks, the adversary has actually oracle access to $\mathcal{E}^+_K(\cdot)$[2].

Security against Key Recovery Attacks. Key recovery attacks aim to recover the whole or partial key of the cryptographic algorithm. We define two kinds of security of symmetric encryption scheme against key recovery attacks

[2] In [9], the adversary has oracle access to $\mathcal{E}_K(\cdot)$ and $S^*_K(\cdot)$ in side-channel model, where the input to $S^*_K(\cdot)$ is the side-channel information and the output of $S^*_K(\cdot)$ is a key $K' \in \{0,1\}^* \cup \{\perp\}$. This formalization conceals where the side-channel information comes from, and brings about confusions in subsequent discussions.

both in black-box model and in side-channel model. In whole key recovery attacks, the adversary tries to recovery the full key.

Definition 1 (WKR and WKR-SCA). *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Consider following two advantages:*

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{WKR}}(\mathbf{A}) = \Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K(\cdot)} \Rightarrow K' : K = K'],$$

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{WKR\text{-}SCA}}(\mathbf{A}) = \Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K^+(\cdot)} \Rightarrow K' : K = K'].$$

We say that $\mathcal{SE}$ is secure against whole key recovery attacks in black-box model (in side-channel model), or WKR-*secure* (WKR-SCA-*secure*), *if the advantage* $\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{WKR}}(\mathbf{A})$ ($\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{WKR}}(\mathbf{A})$) *is negligible for any adversary* $\mathbf{A}$ *with feasible resources.*

In partial key recovery attacks, the adversary tries to recover any information of the key. We adopt a simulator-based definition, in which we use a function $f(K)$ to represent the targeted information of the key and define the security against partial key recovery attacks as whatever an adversary $\mathbf{A}$ with oracle $\mathcal{E}_K$ ($\mathcal{E}_K^+$) can do, a simulator $\mathbf{S}$ without oracle also can do.

Definition 2 (PKR and PKR-SCA). *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Consider following two advantages:*

$$\begin{aligned} &\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{PKR}}(\mathbf{A}, \mathbf{S}) \\ =&\Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K(\cdot)} \Rightarrow b : f(K) = b] - \Pr[K \leftarrow \mathcal{K}, \mathbf{S} \Rightarrow b : f(K) = b], \\ &\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{PKR\text{-}SCA}}(\mathbf{A}, \mathbf{S}) \\ =&\Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K^+(\cdot)} \Rightarrow b : f(K) = b] - \Pr[K \leftarrow \mathcal{K}, \mathbf{S} \Rightarrow b : f(K) = b], \end{aligned}$$

where $f : \{0,1\}^* \to \{0,1\}^*$ is a function. We say that $\mathcal{SE}$ is secure against partial key recovery attacks in black-box model (in side-channel model), or PKR-*secure* (PKR-SCA-*secure*), *if for any function f and any adversary* $\mathbf{A}$ *with feasible resources, there exists an algorithm* $\mathbf{S}$ *(we often call it a simulator) with feasible resources, such that the advantage* $\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{PKR}}(\mathbf{A}, \mathbf{S})$ ($\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{PKR\text{-}SCA}}(\mathbf{A}, \mathbf{S})$) *is negligible.*

Security against Distinguishing Attacks. We adopt the security definition of Real-Or-Random in [1] for symmetric encryptions, which define the security as indistinguishability of ciphertexts of required plaintexts and ciphertexts of random strings.

Definition 3 (IND and IND-SCA). *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Consider following two advantages:*

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{IND}}(\mathbf{A}) = \Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K(\$(\cdot))} \Rightarrow 1],$$

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{IND}}(\mathbf{A}) = \Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K^+(\cdot)} \Rightarrow 1] - \Pr[K \leftarrow \mathcal{K}, \mathbf{A}^{\mathcal{E}_K^+(\$(\cdot))} \Rightarrow 1],$$

where $\$(\cdot)$ *returns a random string with the same length of the input. We say that* $\mathcal{SE}$ *is secure against distinguishing attacks in black-box model* (*in side-channel model*), *or* IND-*secure* (IND-SCA-*secure*), *if the advantage* $\mathbf{Adv}^{\mathrm{IND}}_{\mathcal{SE}}(\mathbf{A})$ ($\mathbf{Adv}^{\mathrm{IND\text{-}SCA}}_{\mathcal{SE}}(\mathbf{A})$) *is negligible for any adversary* $\mathbf{A}$ *with feasible resources.*

3 KR vs. IND in Block-Box Model

In this section, we elaborate the implications or separations of the notions summarized in Figure 1.

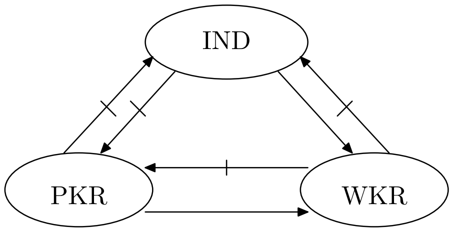


Fig. 1. Relations among PKR, WKR and IND

Theorem 1 (PKR → WKR). *Let* $\mathcal{SE}$ *be an encryption scheme. If* $\mathcal{SE}$ *is* PKR-*secure, then it is* WKR-*secure as well.*

Proof. If $\mathbf{A}$ is an adversary against WKR-security, we construct an adversary $\mathbf{B}$ against PKR-security: run $\mathbf{A}$ and get K', then return K'. We set $f = ID$ where ID is the identical transformation, then for any simulator $\mathbf{S}$, we have $\mathbf{Adv}^{\mathrm{PKR}}_{\mathcal{SE}}(\mathbf{B},\mathbf{S}) \geq \mathbf{Adv}^{\mathrm{PKR}}_{\mathcal{SE}}(\mathbf{A}) - 1/2^k$, where k is the length of the key. □

Theorem 2 (IND → WKR). *Let* $\mathcal{SE}$ *be a symmetric encryption scheme. If* $\mathcal{SE}$ *is* IND-*secure, then it is* WKR-*secure as well.*

Proof. If $\mathbf{A}$ is an adversary against WKR-security, we construct an adversary $\mathbf{B}$ against IND-security: run $\mathbf{A}$ and get K', then query $M \in \{0,1\}^n$ and get C, if $\mathcal{E}_{K'}(M) = C$, then return 1, else return 0. We have that $\mathbf{Adv}^{\mathrm{IND}}_{\mathcal{SE}}(\mathbf{B}) \geq \mathbf{Adv}^{\mathrm{WKR}}_{\mathcal{SE}}(\mathbf{A}) - 1/2^n$. □

Proposition 1 (PKR $\nrightarrow$ IND). *There exists a symmetric encryption scheme which is* PKR-*secure, but not* PKR-*secure.*

Proof. We construct a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where $\mathcal{E}_K(M) = M$, i.e. the encryption algorithm is an identical transformation and has nothing to do with the key K. Hence no matter how the adversary queries the encryption oracle, no information about K is obtained. More specifically, for any function f and adversary $\mathbf{A}$ against PKR-security, the simulator $\mathbf{S}$ just runs

$\mathbf{A}^{ID(\cdot)}$ and returns whatever $\mathbf{A}$ returns, where $ID(\cdot)$ is the identical transformation. Then $\mathbf{Adv}^{\text{PKR}}_{\mathcal{SE}}(\mathbf{A}, \mathbf{S}) = 0$.

Furthermore, the identical transformation reveals all the information about the plaintext. More specifically, the adversary $\mathbf{B}$ just queries $M \in \{0,1\}^n$, if the answer is M then return 1, else return 0. We have $\mathbf{Adv}^{\text{IND}}_{\mathcal{SE}}(\mathbf{B}) = 1 - 1/2^n$. $\square$

Proposition 2 (IND $\nrightarrow$ PKR). *Given a symmetric encryption scheme $\mathcal{SE}$ which is* IND*-secure, we can construct a symmetric encryption scheme $\mathcal{SE}'$ which is also* IND*-secure, but not* PKR*-secure.*

Proof. Suppose $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is IND-secure. We construct $\mathcal{SE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ as follows:

	$\mathcal{SE}$	$\mathcal{SE}'$
Key generation	$K \leftarrow \mathcal{K}$	$K_1 \leftarrow \mathcal{K}, K_2 \overset{\$}{\leftarrow} \{0,1\}^n$
Encryption	$\mathcal{E}_K(M)$	$\mathcal{E}'_{K_1 K_2}(M) = \mathcal{E}_{K_1}(M) \| K_2$
Decryption	$\mathcal{D}_K(C)$	$\mathcal{D}'_{K_1 K_2}(C) = \mathcal{D}_{K_1}(C)$

The new encryption scheme $\mathcal{SE}'$ generates an extra key $K_2 \in \{0,1\}^n$, but reveals it in the ciphertext. If $\mathcal{SE}$ is IND-secure, then $\mathcal{SE}'$ is also IND-secure. More specifically, for any adversary $\mathbf{A}$ attacking $\mathcal{SE}'$, we construct adversary $\mathbf{B}^{\mathcal{O}}$ attacking $\mathcal{SE}$: $K_2 \overset{\$}{\leftarrow} \{0,1\}^n$, run $\mathbf{A}$, when $\mathbf{A}$ queries M, answer it with $\mathcal{O}(M) \| K_2$, and return whatever $\mathbf{A}$ returns. We have $\mathbf{Adv}^{\text{IND}}_{\mathcal{SE}}(\mathbf{B}) = \mathbf{Adv}^{\text{IND}}_{\mathcal{SE}'}(\mathbf{A})$.

Furthermore, the ciphertext reveals the partial key of the $\mathcal{SE}'$, so it is not PKR-secure. More specifically, given the function $f(K_1 K_2) = K_2$ and the adversary $\mathbf{A}$ which returns K_2 after arbitrary one query, any simulator $\mathbf{S}$ has no information about K_2, therefore $\mathbf{Adv}^{\text{PKR}}_{\mathcal{SE}'}(\mathbf{A}, \mathbf{S}) \geq 1 - 1/2^n$. $\square$

Corollary 1 (WKR $\nrightarrow$ IND). *There exists a symmetric encryption scheme $\mathcal{SE}$ which is* WKR*-secure, but not* IND*-secure.*

Proof. We have PKR $\rightarrow$ WKR by Theorem 1. If WKR $\rightarrow$ IND, then PKR $\rightarrow$ IND. That contradicts Proposition 1. $\square$

Corollary 2 (WKR $\nrightarrow$ PKR). *There exists a symmetric encryption scheme $\mathcal{SE}$ which is* WKR*-secure, but not* PKR*-secure.*

Proof. We have IND $\rightarrow$ WKR by Theorem 2. If WKR $\rightarrow$ PKR, then IND $\rightarrow$ PKR. That contradicts Proposition 2. $\square$

4 KR vs. IND in Side-Channel Model

Attacks in black-box model can be regarded as special attacks in side-channel model when the leakage function returns nothing. Therefore the separations of the notions still hold in side-channel model. It is easy to verify the implications of the notions also hold in side-channel model. We summarize these results in Figure 2.

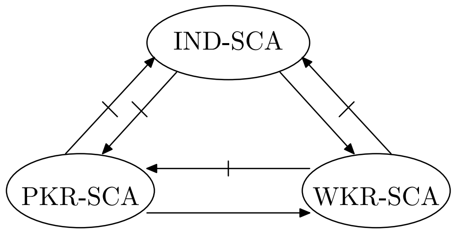


Fig. 2. Relations among PKR-SCA, WKR-SCA and IND-SCA

5 Failing Combination of Notions

From the above section, we know that the notion of WKR-SCA is much weaker than the notion of IND-SCA. Even if we combine the notion of WKR-SCA with that of IND, we can not get the notion of IND-SCA. Therefore we actually overturn the main result in [9].

Proposition 3 (WKR-SCA $\wedge$ IND $\nrightarrow$ IND-SCA). *Given a symmetric encryption scheme $\mathcal{SE}$ which is* IND*-secure, we can construct a symmetric encryption scheme $\mathcal{SE}'$ which is both* IND*-secure and* WKR-SCA*-secure for some leakage function, but not* IND-SCA*-secure for the same leakage function.*

Proof. Suppose $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ which is IND-secure. We construct $\mathcal{SE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ as follows:

	$\mathcal{SE}$	$\mathcal{SE}'$
Key generation	$K \leftarrow \mathcal{K}$	$K_1 \leftarrow \mathcal{K}, K_2 \overset{\$}{\leftarrow} \{0,1\}^n$
Encryption	$\mathcal{E}_K(M)$	$\mathcal{E}'_{K_1K_2}(M) = \mathcal{E}_{K_1}(M)$
Decryption	$\mathcal{D}_K(C)$	$\mathcal{D}'_{K_1K_2}(C) = \mathcal{D}_{K_1}(C)$

The new encryption scheme $\mathcal{SE}'$ generates an extra $K_2 \overset{\$}{\leftarrow} \{0,1\}^n$, but does not used in the encryption. The encryption algorithms of $\mathcal{SE}$ and $\mathcal{SE}'$ are the same, so $\mathcal{SE}'$ is also IND-secure.

Now we consider the security of $\mathcal{SE}'$ in side-channel model, given that the leakage function is $L(K_1K_2, M) = K_1$.

The encryption algorithm of $\mathcal{SE}'$ does not use the key K_2, which is also not revealed by the leakage function, so it is WKR-SCA-secure. More specifically, for any adversary $\mathbf{A}$, $\mathbf{Adv}^{\text{WKR-SCA}}_{\mathcal{SE}'}(\mathbf{A}) \leq 1/2^n$.

Furthermore, the key K_1 used in the encryption algorithm is revealed by the leakage function, so $\mathcal{SE}'$ is not IND-SCA-secure. More specifically, the adversary $\mathbf{B}$ makes arbitrary query and gets K_1 through the leakage function, and then queries $M \in \{0,1\}^n$, gets C. If $C = \mathcal{E}'_{K_1}(M)$, then return 1, else return 0. We have $\mathbf{Adv}^{\text{IND-SCA}}_{\mathcal{SE}}(\mathbf{A}) = 1 - 1/2^n$. $\square$

Moreover, we show that even the scheme is PKR-SCA-secure and IND-secure, the approach in [9] still does not hold.

Proposition 4 (PKR-SCA $\wedge$ IND $\nrightarrow$ IND-SCA). *Given a symmetric encryption scheme $\mathcal{SE}$ which is both* PKR-*secure and* IND-*secure, we can construct a symmetric encryption scheme $\mathcal{SE}'$ which is both* PKR-SCA-*secure and* IND-*secure for some leakage function, but not* IND-SCA-*secure for the same leakage function.*

Proof. Suppose $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is both PKR-secure and IND-secure.

Now we consider the security of $\mathcal{SE}$ in side-channel model, given that the leakage function is $L(K, M) = M$.

The leakage function does not reveal any information about the key K, so $\mathcal{SE}$ is PKR-SCA-secure.

The leakage function reveals the queried plaintext, so $\mathcal{SE}$ is not IND-SCA-secure. More specifically, the adversary $\mathbf{A}$ just queries $M \in \{0,1\}^n$ and gets (C, M'). If $M = M'$ then return 1, else return 0. We have $\mathbf{Adv}_{\mathcal{SE}}^{\text{IND-SCA}}(\mathbf{A}) = 1 - 1/2^n$. $\square$

6 Conclusion

This paper gives implications or separations among the notions of IND, PKR and WKR both in black-box model and side-channel model, which show that the notion of WKR is much weaker than the notion of IND. Then we construct a concrete scheme to show that the approach against side-channel attacks proposed in [9] is flawed. The security against key recovery attacks does not help much for a practically cryptographic requirement. We note that the results are not limited to the security of symmetric encryption scheme, as to the security of the other cryptographic algorithms, such as block ciphers or authenticated encryption schemes, the corresponding results still hold true.

Acknowledgment

This research is supported by the National Natural Science Foundation Of China (No. 60673083, 60873259), the National High-Tech Research and Development 863 Plan of China (No. 2007AA01Z470) and the National Grand Fundamental Research 973 Program of China(No.2007CB311202).

References

1. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. In: Proceedings of the 38th Symposium on Foundations of Computer Science (FOCS), pp. 394–403. IEEE Computer Society Press, Los Alamitos (1997)

2. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
3. Dobbertin, H., Knudsen, L., Robshaw, M.: The cryptanalysis of the AES - A brief survey. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 1–10. Springer, Heidelberg (2005)
4. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
5. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28, 270–299 (1984)
6. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
7. Kocher, P., Jaffe, J., Jun, B.: Introduction to differential power analysis and related attacks (1999), http://www.cryptography.com/dpa/technical/
8. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
9. Li, W., Gu, D.: An approach for symmetric encryption against side channel attacks in provable security. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 178–187. Springer, Heidelberg (2007)
10. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
11. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. Cryptology ePrint Archive, Report 2006/139 (2006), http://eprint.iacr.org/
12. Standaert, F.-X., Peeters, E., Archambeau, C., Quisquater, J.-J.: Towards security limits in side-channel attacks. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 30–45. Springer, Heidelberg (2006)

Ranking Attack Graphs with Graph Neural Networks

Liang Lu[1], Rei Safavi-Naini[2], Markus Hagenbuchner[1], Willy Susilo[1], Jeffrey Horton[1], Sweah Liang Yong[1], and Ah Chung Tsoi[3]

[1] University of Wollongong, Wollongong, Australia
{ll97,rei,markus,wsusilo}@uow.edu.au
[2] Department of Computer Science, University of Calgary, Canada
rei@ucalgary.ca
[3] Hong Kong Baptist University, Kowloon, Hong Kong
act@hkbu.edu.hk

Abstract. Network security analysis based on attack graphs has been applied extensively in recent years. The ranking of nodes in an attack graph is an important step towards analyzing network security. This paper proposes an alternative attack graph ranking scheme based on a recent approach to machine learning in a structured graph domain, namely, Graph Neural Networks (GNNs). Evidence is presented in this paper that the GNN is suitable for the task of ranking attack graphs by learning a ranking function from examples and generalizes the function to unseen possibly noisy data, thus showing that the GNN provides an effective alternative ranking method for attack graphs.

1 Introduction

A large computer system is built upon multiple platforms, runs different software packages and has complex connections to other systems. Despite the best efforts by system designers and architects, there will exist vulnerabilities resulting from bugs or design flaws, allowing an adversary (attacker) to gain a level of access to systems or information not desired by the system owners. The act of taking advantages of an individual vulnerability is referred to as an "atomic attack" or an "exploit". A vulnerability is often exploited by feeding a specially crafted chunk of data or sequence of commands to a piece of defective software resulting in unintended or unanticipated behaviour of the system such as gaining control of a computer system or allowing privilege escalation. Although an exploit may have only insignificant impact on the system by itself, an adversary may be able to construct a system *intrusion* that combines several atomic attacks, each taking the adversary from one system state to another, until attaining some state of interest, such as access to customer credit card data. To evaluate the security level of a large computer system, an administrator must not only take into account the effects of exploiting each individual vulnerability, but also considers global intrusion scenario where an adversary may combine several exploits possibly in a multi-stage attack to compromise the system.

In order to provide a global view on multi-stage attacks by combining several individual vulnerabilities, Sheyner *et al.* proposed the use of attack graphs [11]. An attack graph is a graph that consists of a set of nodes and edges, where each node represents a reachable system *state* and each edge represents an *atomic attack* that takes the system from one state to another. However, as the size and complexity of an attack graph usually greatly exceeds the human ability to visualize, understand and analyze, a scheme is required to identify important portions of attack graphs (which may be vulnerable to external attacks). Recently, Mehta *et al.* [15] proposed to rank states of an attack graph by their importance using PageRank [10], an algorithm used by the Google web search engine. This is possible since the class of attack graphs is a subset of a web graph with nodes representing web pages, and edges representing hyperlinks among the web pages. The PageRank algorithm would rank the importance of web pages based on its hyperlink structure only [10]. In other words, Mehta *et al.* demonstrated that the PageRank algorithm when applied to attack graphs allow the color coding of nodes in the graphs so as to highlight its important portions.

Questions such as "what happens if a particular vulnerability is patched" or "what happens if a firewall rule is altered, added or removed" are often raised in an organization before any planned network structural change takes place. Being able to answer such questions (without actually carrying the reconfiguration) is necessary for the network administrator to evaluate the results of the planned network changes. Clues to answering such questions can be obtained from analyzing the attack graph resulted from situation changes in the modeled system. However, the PageRank algorithm [10] is not of linear time complexity and thus it may be difficult to rank many attack graphs, each resulting from one of many possible changes in the modeled system.

In this paper, we consider an alternative scheme to estimate ranks of attack graphs based on the Graph Neural Network (GNN) [3], a new neural network model capable of processing graph structures. The applications of GNN have been successful in a number of areas where objects are naturally represented by a graph structure, the most notable example being the ranking of web pages [13]. In [13] it is shown that the GNN can be trained to simulate the PageRank function by using a relatively small set of training samples. Once trained, the GNN can rank large sets of web pages efficiently due to its ability to generalize over unseen data. In other words, the GNN learns the ranking function (whether explicit, as in the case of PageRank, or implicit) of web pages from a small number of training samples, and can then generalize over unseen examples, possibly contaminated by noise.

Because of the above stated properties, GNN may be considered suitable for the task of ranking attack graphs. Moreover, [12] provides a universal approximation theorem that the GNN is capable of learning to approximate any "reasonable" problem, reasonable in the sense that the problem is not a pathological problem, to any arbitrary degree of desired precision. The universal approximation theorem further justifies the suitability of ranking attach graphs using GNN. However on the other hand, some properties of attack graphs differ fundamentally from those of the web graph. The web graph is a single very large graph with a recursive link structure (web pages referring to themselves) whereas attack graphs can be numerous, are relatively small, and may be of strictly tree structured. This together with the observation that PageRank is a link based algorithm whereas the GNN is a node based approach in that it processes nodes more effectively than links, whether the GNN is suitable for the task of ranking attack graphs is unknown. One of the key questions that this paper wishes to answer is the suitability of

GNN for the purpose of ranking attack graphs. There were numerous prior approaches to the processing of attack graphs and similar poli-instantiation problems. Research in this area was particularly active in the 1980s and early 1990s (see for example [8]). Most of these work were of an automated proof nature, desiring to show if an attack graph is vulnerable or not. Such automated proof concept can be expressed in terms of graph structured data [5]. However, any attempts to use a machine learning approaches required the pre-processing of the graph structured data to "squash" the graph structure into a vectorial form as most popular machine learning approaches, e.g. multilayer perceptrons, self organizing maps [6], take vectorial inputs rather than graph structured inputs. Such pre-processing step can result in the loss of contextual information between atomic components of the data. The GNN is a relatively recent development of a supervised machine learning approach which allows the processing of generic graphs without first requiring the "squashing" of graph structured data into vectorial form. It is shown in [12] that the GNN algorithm is guaranteed to converge, and that it can model any set of graphs to any arbitrary degree of precision. Hence, to the best of our knowledge, this paper is the first reported work on ranking attack graphs (without pre-processing) using a machine learning approach.

Training a GNN can take a considerable period of time. It is shown in [3] that the computational burden of the training procedure can be quadratic. However, once trained, a GNN is able to produce output in linear time. This is an improvement over $O(N \log \frac{1}{\epsilon})$, the computational complexity of PageRank, where N is the number of links and ϵ is the expected precision [10]. Moreover, GNN is able to learn the ranking scheme from a small number of training examples and then generalize to other unseen attack graphs. For reasons stated above, using GNN may be more suitable than the PageRank algorithm in the case where it requires ranking many attack graphs, each resulting from one of many possible changes in the modeled system.

The contributions of this paper include: (a) an alternative attack graph ranking method based on a supervised machine learning approach, viz., GNN is proposed, and (b) both the PageRank-based and the GNN-based ranking approaches are implemented and their results are compared. This provides an insight into the suitability of applying GNN to rank attack graphs.

The paper is organized as follows. Section 2 provides a brief review of relevant background. The proposed alternative ranking scheme is presented in Section 3. Section 4 provides the experimental results to show the effectiveness of the proposed scheme, and Section 5 concludes the paper.

2 Background

To evaluate the security condition of a computer system built upon multiple platforms, runs different software packages and has complex connections to other systems, a network administrator needs to consider not only the damages that can be done by exploiting individual vulnerabilities on the system, but also investigates the global effect on what an intruder can achieve by combining several vulnerabilities possibly in multi-stages which by themselves only have insignificant impact on the system. This requires an appropriate modeling of the system that takes into account information such as vulnerabilities and connectivity of components in the system, and is able to model a global intrusion scenario where an intruder combines several vulnerabilities to attack the system.

Sheyner *et al.* first formally defined the notion of attack graph to provide such a modeling of the system [11]. An *attack graph* describes security related attributes of the attacker and the modeled system using graph representations. States of a system such as the attacker's privilege level on individual system components are encapsulated in graph nodes, and actions taken by the attacker which lead to a change in the state of the system are represented by transition between nodes, *i.e.* edges of the graph. The root node of an attack graph denotes the starting state of the system where the attacker has not been able to compromise the system but is only looking for an entry point to the system. Consider an attack on a computer network as an example: a state may encapsulate attributes such as the privilege level of the attacker, the services being provided, access privileges of users, network connectivity and trust relations. The transitions correspond to actions taken by the attacker such as exploiting vulnerabilities to attain elevated privileges. The negation of an attacker's goal against the modeled system can be used as *security properties* that the system must satisfy in a secure state. An example of a security property in computer networks would be "the intruder **cannot** log in onto the central server". Nodes in an attack graph where the *security properties* are not satisfied are referred to as *error states*. A path from the root node to an error state indicates how an intruder exploits several vulnerabilities, each corresponding to one of the edges on the path, to finally reach a state where the system is considered compromised. An attack graph as a whole represents all the intrusion scenarios where an intruder can compromise the system by combining multiple individual vulnerabilities. Following these descriptions, an *attack graph* is formally defined as follows

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

In other words, an attack graph is a tree structured graph where the root is the starting state, leaf nodes are final states denoting an intruder having reached a targeted system, and intermediate nodes are the intermediate states that a system can have during an attack. Typically, for mid- and large-sized organizations, an attack graph consists of several hundred or thousand nodes. Several attack graphs are formed to reflect the numerous various subnets or security configurations of a network.

Given a system and the related information such as vulnerabilities in the system, connectivity of components and security properties, model checking techniques can be used to generate attack graphs automatically [11].

2.1 Multi-layer Perceptron Neural Networks

The application of neural networks [6] for pattern recognition and data classification has gained acceptance in the past. In this paper, *supervised* neural networks based on *layered* structures are considered. Multilayer perceptrons (MLPs) are perhaps the most well known form of supervised neural networks [6]. MLPs gained considerable acceptance among practitioners from the fact that a single-hidden-layer MLP has a universal

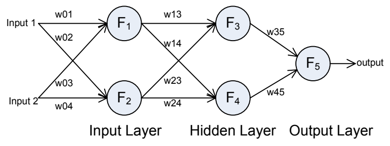


Fig. 1. An example of a multi-layered perceptron neural network, where F_1, and F_2 form the input layer, F_3 and F_4 form the hidden layer, while F_5 forms the output layer

approximation property [7], in that it can approximate a non-pathological nonlinear function to any arbitrary degree of precision.

The basic computation unit in a neural network is referred to as a *neuron* [6]. It generates the output value through a parameterized function called a *transfer function* f. An MLP can consist of several layers of neurons. The neuron layer that accepts external input is called the input layer. The dimension of the input layer is identical to the dimension of the data on which an MLP is trained. The layer that generates the output of the network is called the output layer. Its dimension is identical to the dimension of the target values. A layer between the input layer and the output layer is known as a hidden layer. There may be more than one hidden layer but here for simplicity we will only consider the case of a single hidden layer. Each layer is fully connected to the layer above or below[1]. The input to each hidden layer neuron and the output layer neuron is the weighted sum from the previous layer, parameterized by the connecting weights, known as synaptic weights. The multiple layered structure described here has given MLP its name.

Figure 1 illustrates an example of a single-hidden-layered MLP. The transfer function associated with each of the neurons is $F_j = f(\sum_{i=0}^{n} w_{ji}x_i)$, where n is the total number of neurons in the previous layer, $x_0 = 1$, x_i is the $i-$th input (either a sensory input or the output F_j from a previous layer), and w_{ji} is a real valued weight connecting neuron (or input) i with neuron j. The transfer function $f(\cdot)$ often takes the shape of a hyperbolic tangent function. The MLP processes data in a *forward* fashion starting from the input layer towards the output layer.

Training an MLP [6] is performed through a *backward* phase known as error back propagation, starting from the output layer. It is known that an MLP can approximate a given unknown continuously differentiable function[2] $g(x_1, \ldots x_n)$ by learning from a *training data set* $\{x_{i1}, \ldots x_{in}, t_i\}$, $1 \leq i \leq r$ where $t_i = g(x_{i1}, \ldots x_{in})$. It computes the output for each input $X_i = \{x_{i1}, x_{i2} \ldots x_{in}\}$ and compares the output with the target t_i. Weights are then adjusted using the gradient descent algorithm based on the error towards the best approximation of the target t_i. The forward and the backward phases are repeated a number of times until a given prescribed accuracy (stopping criterion) is met *e.g.* the output is "sufficiently" close to the target.

Once the stopping criterion is met, the learning stage is finished. The MLP is then said to emulate the unknown nonlinear function $g(\cdot, \ldots, \cdot)$. Given any input X_i with

[1] Fully connected layers are the most commonly studied and used architectures. There are some neural network architectures which are connected differently.

[2] The function does not need to be continuously differentiable. However for our purpose in this paper, we will assume this simplified case.

unknown target t_i, it will be able to produce an output from g or it is said to be able to generalize from the training set to the unseen input X_i.

The application of MLP networks has been successful in many areas [3,13,6]. The inputs to the MLP would need to be in vectorial form, and as such, it cannot be applied to graph structured inputs or inputs which relate to one another.

2.2 Graph Neural Network

In several applications including attack graphs and web page ranking, objects and their relations can be represented by a graph structure such as a tree or a directed acyclic graph. Each node in the graph structure represents an object in the problem domain, and each edge represents relations between objects. For example, an attack graph node represents a particular system state, and an edge represents that the attacker can reach one state from the other. Graph Neural Network (GNN) is a recently proposed neural network model for graphical-based learning environments [3]. It is capable of learning topological dependence of information representing an object such as its rank relative to its neighbors, *i.e.* information representing neighboring objects and the relation between the object and its neighboring objects. The use of GNN for determining information on objects represented as nodes in a graph structure has been successfully shown in several applications [13,4].

The GNN is a complex model. A detailed description of the model and its associated training method cannot be described in this paper due to page limits, and hence, this section only summarizes the key ideas behind the approach as far as it is necessary to understand the method. The interested reader is referred to [3].

In order to encapsulate information on objects, a vector $s_n \in \mathbb{R}^s$ referred to as *state*[3], which represents information on the object denoted by the node, is attached to each node n. This information is dependent on the information of neighboring nodes and the edges connecting to its neighbors. For example, in the case of page rank determinations, the *state* of a node (page) is the rank of the page. Similarly, a vector e_{ij} may also be attached to the edge between node n_i and n_j representing attributes of the edge [4]. The state of a node n is determined by states of its neighbors and the labels of edges that connect it to one of its neighbors. Consider the GNN shown in Figure 2, the state of node n_1 can be specified by Equation (1).

$$s_1 = f_w(s_2, s_3, s_5, e_{21}, e_{13}, e_{51}) \tag{1}$$

More generally, let $\mathbb{S}(n)$ denote the set of nodes connected to node n and $f_{\mathbf{w}_1}$ be a function parameterized by $\mathbf{w}_1$ that expresses the dependence of information representing a node on its neighborhood, then the state s_n is defined by Equation (2) where

[3] For historical reasons this is called a "state", which carries slightly different meaning to the meaning of "state" in the attack graph literature. However, there should not be any risk of confusion as this concept of "state" in GNN is used in the training of the GNN model.

[4] GNN can also accept labels on nodes; however this will not be used in this paper and is omitted here.

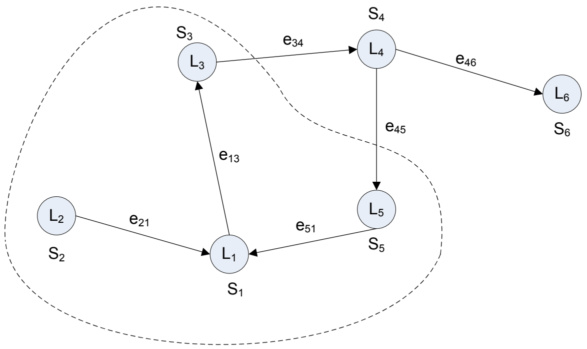


Fig. 2. The dependence of state s_1 on neighborhood information

$\mathbf{w}_1 \in \mathbb{R}$ is the parameter set of function f which is usually implemented as an MLP described in Section 2.1.

$$s_n = f_{\mathbf{w}_1}(s_u, e_{xy}), u \in \mathbb{S}(n), x = n \vee y = n \tag{2}$$

The state s_n is then put through a parameterized *output network* $g_{\mathbf{w}_2}$, which is usually also implemented as an MLP parameterized by $\mathbf{w}_2$, to produce an output (*e.g.* a rank) o_n

$$o_n = g_{\mathbf{w}_2}(s_n) \tag{3}$$

Equations (2) and (3) define a function $\varphi_w(G, n) = o_n$ parameterized by weights $\mathbf{w}_1$, and $\mathbf{w}_2$ that produces an output for the given graph G. The parameter sets $\mathbf{w}_1$ and $\mathbf{w}_2$ are adapted during the training stage such that the output φ is the best approximation of the data in the learning data set $\mathcal{L} = \{(n_i, o_i) | 1 \leq i \leq q\}$, where n_i is a node and t_i is the desired target for n_i.

2.3 Existing Attack Graph Ranking Schemes

The PageRank algorithm [10] is used by Google to determine the relative importance of web pages in the World Wide Web. The PageRank is based on the behavioural model of a random surfer in a web graph. It assumes that a random surfer starts at a random page and keeps navigating by following hyperlinks, but eventually gets bored and starts on another random page. To capture the notion that a random surfer might get bored and restart from another random page, a *damping factor* d is introduced, where $0 < d < 1$. The transition probability from a state is divided into two parts: d and $1-d$. The d mass is divided equally among the state's successors. Random transitions are added from that state to all other states with the residual probability $1-d$ equally divided amongst them, modeling that if a random surfer arrives at a dangling page where no links are available, he is assumed to pick another page at random and continues surfing from that page. The computed rank of a page is the probability of a random surfer reaching that page, *i.e.*, consider a web graph with N pages linked to each other by hyperlinks, the PageRank x_p of page (node) p can be represented by Equation (4) where $pa[p]$ denotes the set of nodes (pages) pointing to node (page) p

$$x_p = d \sum_{q \in pa[p]} \frac{x_q}{h_q} + \frac{1-d}{N} \tag{4}$$

When stacking all the x_p into a vector **x**, and using iterative expressions, this can be represented as

$$\mathbf{x}(t) = dW\mathbf{x}(t-1) + \frac{1}{N}(1-d)\Pi_N \tag{5}$$

The computation of the PageRank can be considered a Markov process, as can be observed from Eq. (5). It has been shown hat after multiple iterations, Eq. (5) will converge to a stationary state where each x_p represents the probability of the random surfer reaching page p [15]. Time complexity required by PageRank is $N\log\frac{1}{\epsilon}$ where N is the number of hyperlinks and ϵ is the expected precision [10].

Based on the PageRank algorithm, Mehta *et al.* proposed a ranking scheme for attack graphs as follows [15]: given an attack graph $G = (S, \tau, s_0, S_s, l)$, the transition probability from each state is divided into d and $1-d$, modeling respectively an attacker is discovered and isolated from the system, or that the attacker remains undetected and proceeds to the next state with the intrusion. Similar to PageRank, the rank of a state in an attack graph is defined as the probability of the system being taken to that state by a sequence of exploits. The ranks of all states are computed using Equation (5). Breadth first search starting from the initial system state s_0 is then performed for each atomic attack in τ to construct the transition matrix W. The adjustment from PageRank, where a transition from each state pointing to all other states with probability $1-d$ equally divided amongst all other states, is that a transition from each state pointing back to the initial state with probability $1-d$ is added to model the situation where an attacker is discovered and has to restart the intrusion from the initial state. The rationale behind the use of the PageRank algorithm to rank attack graphs is based on the possibility of a hacker pursuing the hacking. In other words, a hacker being successful in obtaining root status may not be fully satisfied until further (or all possible) routes to obtaining root status are explored, and hence, may restart at any system state for further attempts to find vulnerabilities leading to a root status. Note that PageRank helps to find nodes that can more easily be reached by chance from one of the starting states. This describes the blind probing behavior of attackers. In practice, a human attacker is more likely to have some intuition that, say, getting root access on the mail server is more preferable than getting root access on one user's machine inside the network. In other words, PageRank is most likely to lead to states which are easiest to reach, and least likely to reach states which are less likely to be reached. There is work which modifies the PageRank algorithm slightly to allow control of the importance of some states. In the literature, this is refered to as "personalization" achieved through replacing the 1-vector Π_N in Eq. (5) by a parameterized vector (e.g. [14]). It is shown in [13] that the GNN can successfully encode the personalized PageRank algorithm, and hence, this paper does not attempt to explicitly show this aspect in the context of ranking attack graphs.

3 Ranking Attack Graph Using GNNs

The use of GNN for determining information on objects represented as nodes in a graph structure has been shown in several applications such as ranking web pages. In this paper, we develop an implementation of the GNN model as described in Section 2.2 applied to ranking attack graphs. For attack graphs, nodes represent computer system states and edges represent the transition relations between system states. Hence, the *state* of a node is determined by states and outgoing links of its parent nodes. Symbolic values 0 or 1 are also assigned to edge labels to distinguish between incoming and outgoing links. The function $f_{\mathbf{w}_1}$ that expresses the dependence of the state s_n of node n on its neighbors can be represented by Equation (6) where $\mathbb{S}(n)$ denotes the set of nodes connected to node n,

$$s_n = f_{\mathbf{w}_1}(s_u, e_{uv}), u \in \mathbb{S}(n) \tag{6}$$

The output function $g_{\mathbf{w}_2}$ with input from the states produces the rank of node n.

$$o_n = g_{\mathbf{w}_2}(s_n) \tag{7}$$

The two computing units $f_{\mathbf{w}_1}$ and $g_{\mathbf{w}_2}$ are implemented by MLPs [1]. Inputs to the MLPs are the variables in Eqs. (6) and (7), *i.e.* states of neighboring nodes and labels of connecting edges. We use the *sigmoid* function [1] as the transfer function of $f_{\mathbf{w}_1}$. The *sigmoid* function is commonly used in back-propagation networks. The transfer function of the output network $g_{\mathbf{w}_2}$ is a linear transfer function that stretches the output from $f_{\mathbf{w}_1}$ to the desired target. $f_{\mathbf{w}_1}$ and $g_{\mathbf{w}_2}$ are thus parameterized by weight sets $\mathbf{w}_1$ and $\mathbf{w}_2$ of the underlying MLPs. The adaption of weights sets for the best approximation of training data is through a back-propagation process [1].

It is known that the computational complexity of the forward phase of MLP is O(n), where n is the number of inputs [2]. When considering that the functions $f_{\mathbf{w}_1}$ and $g_{\mathbf{w}_2}$ can be implemented as MLPs, it becomes clear that the computational complexity of the forward phase of GNN is also O(n). Note that a direct comparison of the computational complexity of PageRank is difficult since the computational complexity of PageRank depends on the number of links rather than on the number of nodes.

The GNN implemented is initialized by assigning random weights to the underlying MLPs. The training data is generated using Mehta *et al.*'s ranking scheme on a set of attack graphs. The trained GNN can take as input an attack graph and output through the trained network ranks of each node.

4 Experiments and Results

The objectives of the experiments are to verify the effectiveness of the proposed ranking approach: (1) if GNN can learn a ranking scheme applied to attack graphs, and (2) if it can generalize the results to unseen data. In particular we consider the PageRank scheme used by Mehta *et al.* to rank attack graphs.

It has been shown that GNN can learn PageRank on very large graphs [4,13]. Our objective is to ascertain if it can learn the ranking scheme when applied to attack graphs. We note that the WWW is a single very large graph and the aim of learning has been to generalize the results to this large graph by learning from selected small sub-graphs of WWW. However, in the case of ranking attack graphs, many small graphs of various sizes exist. For a small network with few vulnerabilities, it can be a small graph with few nodes and edges, while for more complex networks, it can have hundreds of nodes and edges. One question is: what type/size of graph should be used for training? It would be ideal to train GNN with a set of attack graphs generated from networks of various complexity. However, while attack graphs can be generated quite easily from real networks, generation of realistic attack graphs of artificial examples requires significant manual labor as well as computational effort [11]. It is, in general, difficult to generate a large number of real attack graphs on artificial examples as training samples. To solve this problem, we experiment with training GNN with a set of automatically generated pseudo attack graphs that have similar shapes and node connection patterns to those of manually generated real attack graphs. Details for pseudo attack graph generation are provided in Section 4.3.

Note that it is time consuming to generate *artificial* attack graphs which resemble the type of attack graphs one would encounter in the real world. Real world attack graphs can be easily generated (the process may even be automated). However, due to a lack of access to large scale distributed system we were unable to obtain real world attack graphs. Instead we use the time consuming task of generating artificial data. We did not attempt to simulate attack graphs for small systems since the resulting graphs would be rather small. Existing approaches to ranking attack graphs are sufficient for small scale systems whereas the advantage of the proposed approach is most pronounced on larger graphs.

A related question is how we determine the effectiveness of the proposed ranking approach. GNN output produces a set of ranks for nodes of an attack graph. There are many ways to define the "accuracy" of a set of ranks compared to the set of ranks that is calculated by another scheme. In particular one may use the average distances of the sets, or the maximum distance over all elements of the set. For our particular application we use *Relative Position Diagram (RPD)* and *Position Pair Coupling Error (PPCE)* as described in Section 4.1. The PPCE measure the agreement between the proposed approach and the ordering imposed on the nodes in an attack graph by PageRank. Hence, this ordering compares the proposed approach with an existing approach. No attempt is made to assess whether the PageRank order by itself makes sense.

4.1 Performance Measure

A ranking scheme can be denoted as a function that takes as input an attack graph (AG) and outputs the ranks, i.e. $f(AG) = R$ where R is a real vector of the same size as the number of nodes in AG. A naive performance measure is to compare the output ranks by GNN $f_G(AG) = R_G$ and that produced by Mehta's ranking scheme $f_M(AG) = R_M$. However, as ranks are used to identify important portions of an attack graph, ordering of ranks are considered more important than their actual numerical values. We therefore devised two performance measures to evaluate how well rank orderings are preserved.

`Relative Position Diagram (RPD)`: This visually illustrates how well rank orders are preserved, and is obtained by sorting R_M and R_G and plotting the order of each node in R_M against that in R_G. The X-axis and Y-axis represent the order of ranks in R_M and R_G respectively. Therefore, nodes that have the same rank orders by both schemes are plotted along the diagonal.

`Position Pair Coupling Error:` An RPD only intuitively shows how well rank orders are preserved. We also provide a quantitative measure on rank order preservation as follows. Let r_i^M and r_i^G denote the i-th element in R_M and R_G respectively. For each pair of nodes, if r_i^M and r_j^M are not in the same order as r_i^G and r_j^G, then a position-pair-coupling-error (PPCE) is found. Performance of the GNN ranking scheme can therefore be quantitatively measured by the PPCE rate.

4.2 Experimental Results

We found that the computational demand of the attack graph generation method in [11] is high. The generation of a single real world attack graph took several hours, and hence, we were not able to produce more than 14 graphs within a reasonable time frame for the experiments. The 14 attack graphs were generated using the same computer network example as in a previous work [9], with variations created by adding, removing, or varying vulnerabilities or network configurations. Eight of the attack graphs were randomly selected to form the training data set of the GNN. The remaining six attack graphs are used as the testing data set so as to examine the GNN's generalization ability in different situations. The GNN used for our experiments consists of two hidden layers, and each hidden layer has 5 neurons. The number of external inputs is also set to 5. Such a GNN has been shown to be successful when applied to ranking web pages [4,13], and it produces a sufficiently small network that allows for a fast processing of the data. However, this GNN was used successfully on a problem involving much larger number of nodes, and hence, it may be possible to use a somewhat smaller network when dealing with attack graphs. This is not attempted in this paper and may be considered as a topic for future work.

We varied the number of training epochs from 500 to 20,000 and repeated the experiment 6 times with randomized initial weights of the underlying MLPs. The PPCE rate results are plotted in Fig. 3. It can be observed that when trained for a sufficiently large number of epochs (around 10,000) the PPCE rate reduces asymptotically to an optimal value. Improvement by further training can be observed but is not significant. Hence the number of training epochs is fixed to be 10,000 in the rest of the experiments for an appropriate effort/time tradeoff. However when applied to ranking attack graphs in practice the training epoch can be set to 20,000 or larger for more accurate results.

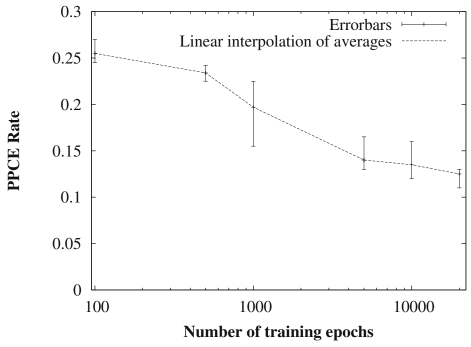


Fig. 3. Effect of number of training epochs

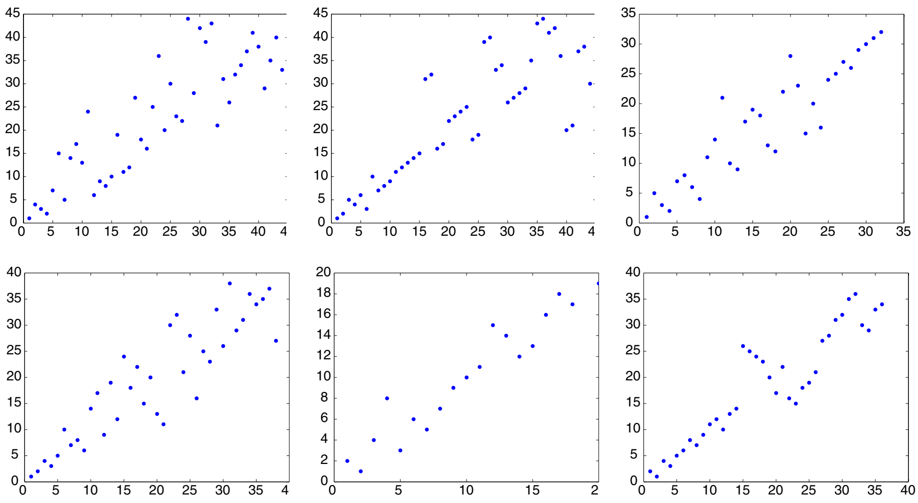


Fig. 4. Relative Position Diagram when trained on real-world attack graphs

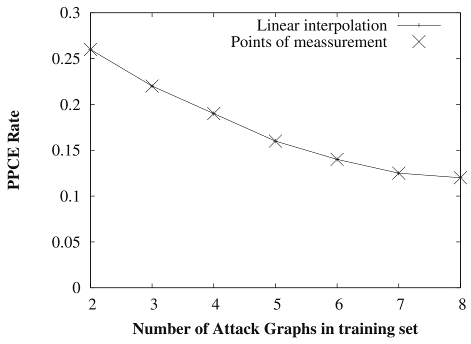


Fig. 5. Effect of number of attack graphs used in the training data set

From the above experimental results, it can be observed that the average PPCE rate can be reduced to around 10% when GNN is trained with sufficient number of training examples and training epochs. That being the case on an average the proposed GNN-based ranking scheme sacrificed around 10% accuracy in terms of relative position preservation. However, once trained, a GNN can produce output at an O(N) time complexity where N is the size of input. This may be an improvement compared with the O($N \log \frac{1}{\epsilon}$) computational complexity required by PageRank. Moreover, the generalization ability of GNN allows us to train GNN on only a small number of attack graphs and apply the results to unseen attack graphs.

Figure 4 plots the RPDs resulting from ranking graphs from the testing data set with GNN to visualize the effectiveness of relative position preservation. Each subgraph represents the resulting RPD for one of the 6 attack graphs in the testing data set, and hence it is of different scale to other subgraphs. It can be observed that the order of ranks are preserved to a reasonable degree. Although only a relatively small portion of nodes remains at exactly the same position as the PageRank scheme, most nodes remain centered around the diagonal and as a result important nodes remain important and unimportant nodes remain unimportant. The GNN-based ranking scheme therefore allows a system administrator to focus on most of the important nodes in an attack graph.

Finally, we experimented with reducing the number of training samples. The number of training attack graphs in the training data set is reduced from 8 to 2. The effect on reducing the number of attack graphs used in the training data set is plotted in Figure 5. It can be observed that the number of attack graphs in the training data set has a significant impact on the accuracy in terms of relative positions on the training results. Sufficient attack graphs must be provided in the training data set to simulate the ranking scheme effectively.

4.3 Training with Pseudo Attack Graphs

Generating a large number of attack graphs for the training data set is difficult due to significant manual labor and computational effort that it requires [11]. In the following, we provided an alternative scheme for automated training set generation, and train GNN with automatically generated pseudo attack graphs that have similar shapes and node connection patterns to those of manually generated realistic attack graphs.

That attack graphs are non-cyclic, tree shaped graphs resulting from a procedure as follows: at the initial system state, the intruder looks for an entry point and reaches the first layer of nodes in the attack graph by exploiting one of the vulnerabilities applicable to the initial state. With the escalated privilege by the initial exploit, the intruder obtains more intrusion options for the next intrusion step. This procedure repeats and keeps expanding the attack graph until the intruder can reach the intrusion goal. Leaf nodes in the attack graph are thus produced, and the attack graph begins to contract until all attack paths reach the intrusion goal. We generate pseudo attack graphs by

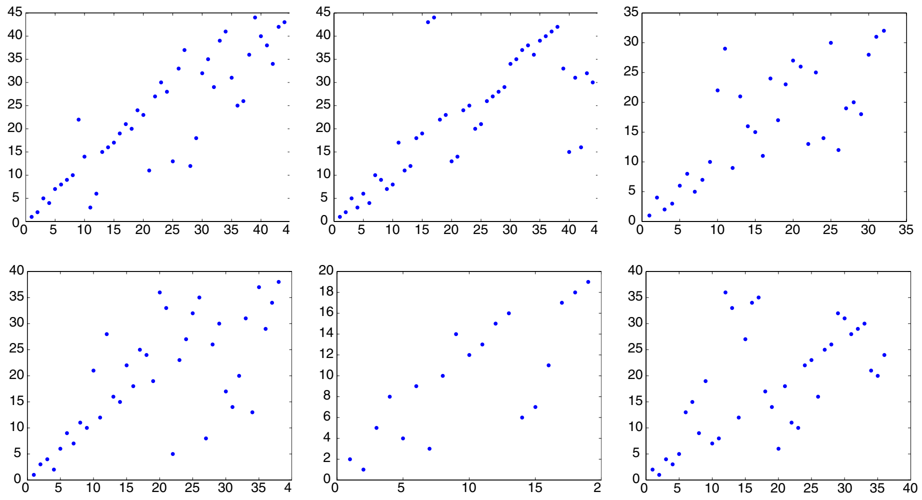


Fig. 6. Relative Position Diagram when training on pseudo attack graphs

Table 1. Position Pair Coupling Error

	AG-1	AG-2	AG-3	AG-4	AG-5	AG-6	Avg
Min PPCE	0.16	0.11	0.14	0.12	0.12	0.11	0.12
Max PPCE	0.21	0.15	0.19	0.17	0.15	0.15	0.16

simulating this procedure. A pseudo attack graph begins with the root node representing the initial state. It then expands with increasing number of nodes at each layer to simulate the expanding process of realistic attack graphs. This repeats until presumably the intruder begins to reach his goal. Then the pseudo attack graph contracts till all paths are terminated at a leaf node.

We repeat the experiments on the same 6 testing attack graphs but using pseudo attack graphs generated by the above procedure as the training data set. In total, 40 pseudo attack graphs are included in the training data set, and GNN is trained for 10,000 epochs for the asymptotic optimal result. The resulting Relative Position Diagrams are as shown in Figure 6. It can be observed that orders of ranks are preserved also to a reasonable degree, but these results are not as accurate as those obtained when using real attack graphs in the training data set. Table 1 lists maximum and minimum PPCE rate by repeating each experiment 3 times. The average PPCE is between 12% to 16%, indicating that on an average the proposed GNN-based ranking scheme sacrificed around 12% to 16% accuracy in terms of relative position preservation. Therefore, using pseudo attack graphs decreases the accuracy by around 2% to 4% compared with using real attack graphs for training, but on the other hand it significantly reduces the effort required to generate real attack graphs.

5 Conclusions

This paper produced evidence that the GNN model is suitable for the task of ranking attack graphs. GNN learns a ranking function by training on examples and generalizes the function to unseen data. It thus provides an alternative and time-efficient ranking scheme for attack graphs. The training stage for GNN, which is required only once, may take a relatively long period of time compared with *e.g.* PageRank scheme. However, once trained the GNN may compute ranks faster than that required by the PageRank scheme.

Another advantage of using a machine learning approach is their insensitivity to noise in a dataset. This is a known property of GNN which is based on the MLP architecture. Existing numeric approaches such as PageRank or rule based approaches are known to be sensitive to noise in the data.

The GNN provides much more flexible means for ranking attack graphs. A GNN can encode labels that may be attached to nodes or links, and hence, is able to consider additional features such as the cost of an attack or time required for an exploit. This is not possible with the PageRank scheme which is strictly limited to considerations of the topology features of a graph. We plan to investigate how the ability to encode additional information can help to rank attack graphs in the future.

Despite the proven ability of the GNN to optimally encode any useful graph structure [12], the result in this paper showed that the performance of the GNN has a potential for improvements. We suspect that this may be due to the choice of training parameters, or the chosen configuration of the GNN. A more careful analysis into this issue is left as a future task.

References

1. Bhadeshia, University Of Cambridge. Neural Networks in Materials Science
2. Mizutani, E., Dreyfus, S.E.: On complexity analysis of supervised MLP-learning for algorithmic comparisons. In: International Joint Conference on Neural Networks (IJCNN), vol. 1 (2001)
3. Scarselli, F., Tsoi, A.C., Gori, M., Hagenbuchner, M.: A new neural network model for graph processing. Technical Report DII 1/05, University of Siena (August 2005)
4. Scarselli, F., Yong, S.L., Hagenbuchner, M., Tsoi, A.C.: Adaptive Page Ranking with Neural Networks. In: WWW (Special interest tracks and posters), pp. 936–937 (2005)
5. Frasconi, P., Gori, M., Sperduti, A.: A general framework for adaptive processing of data structures 9(5), 768–786 (September 1998)
6. Haykin, S.: Neural Networks, A Comprehensive Foundation. Macmillan College Publishing Company, Inc., 866 Third Avenue, New York, 10022 (1994)
7. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359–366 (1989)
8. Kemmerer, R.A., Catherine, M., Millen, J.K.: Three system for cryptographic protocol analysis. Cryptology 7(2), 79–130 (1994)
9. Lu, L., Safavi-Naini, R., Horton, J., Susilo, W.: An adversary aware and intrusion detection aware attack model ranking scheme. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 65–86. Springer, Heidelberg (2007)
10. Bianchini, M., Gori, M., Scarselli, F.: Inside PageRank. ACM Transactions on Internet Technology 5(1), 92–118 (2001)
11. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated generation and analysis of attack graphs. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA (May 2002)
12. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: Computational capabilities of graph neural networks. IEEE Transactions on Neural Networks 20(1), 81–102 (2009)
13. Scarselli, F., Yong, S.L., Gori, M., Hagenbuchner, M., Tsoi, A.C., Maggini, M.: Graph neural networks for ranking web pages. In: Web Intelligence Conference, pp. 666–672 (2005)
14. Tsoi, A.C., Hagenbuchner, M., Scarselli, F.: Computing customized page ranks. ACM Transactions on Internet Technology 6(4), 381–414 (2006)
15. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.M.: Ranking attack graphs. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 127–144. Springer, Heidelberg (2006)

Implementing IDS Management on Lock-Keeper

Feng Cheng, Sebastian Roschke, and Christoph Meinel

Hasso Plattner Institute (HPI), University of Potsdam,
P.O.Box 900460, 14440, Potsdam, Germany
{feng.cheng,sebastian.roschke,christoph.meinel}@hpi.uni-potsdam.de

Abstract. Intrusion Detection System (IDS) management is an important component for most distributed IDS solutions. One of the main requirements is extensibility, which enables the integration of different types of IDS sensors as well as the deployment in different kinds of environments. Lock-Keeper is a simple implementation of the high level security idea, "Physical Separation". It works as a sluice to exchange data between two networks without having to establish a direct and physical connection. To enhance the security of the Lock-Keeper system itself, it is necessary to deploy IDS sensors on Lock-Keeper components. This paper proposes an extensible IDS management architecture, which can be easily integrated on the special hardware platform of Lock-Keeper. Unified interface and communication between different integrated IDS sensors are designed using the known IDS standard, IDMEF, and realized as several kinds of plugins, such as handlers, receivers, and senders. A prototype of implementation is presented and some practical experiments are carried out to show the extensibility and applicability of the proposed architecture.

1 Introduction

Intrusion detection systems (IDS) have been commonly used in practice for identifying malicious behaviors against protected hosts or network environments. An effective IDS should be capable of detecting various types of attacks and all the possible variants of a certain type of attack. Nowadays, a large number of commercial and open source IDS implementations have emerged, such as Snort [1], Samhain [2], Bro [3], F-Secure Linux Security [4] and Prelude [5]. Besides, many research works have been done or are being carried out around various topics, such as proposing new detection models for specific applications [6], accelerating efficiency of alert analysis [7], and improving precision of detection methods [8].

Based on the protected objective, IDS can be classified into host-based intrusion detection system (HIDS) and network-based intrusion detection system (NIDS) [9]. Another widely accepted classification is according to the used detection model, e.g., misuse detection [10] or anomaly detection [11]. Different detection models result in different design and implementation of IDS sensors. Therefore, it makes sense to provide integrated IDS solutions, which make it possible for users to simultaneously benefit from advantages of various kinds of IDS sensors. For this purpose, many distributed IDS (DIDS) solutions, which consists of multiple IDS sensors located in a loosely coupled environment, as well as hybrid IDS solutions, which contain both HIDS and NIDS, have been developed. The Intrusion Detection Message Exchange Format (IDMEF) [12], has been proposed as a standard to provide interoperability among different IDS approaches, including commercial, open source, and research systems. However, due to the highly heterogeneous architecture of these IDS implementations and poor acceptance of the IDMEF standard, integrating different types of IDS into a unified architecture remains to be a big challenge.

As an important implementation of the security concept, Physical Separation, Lock-Keeper has been offered as an efficient approach to separate private networks or sensitive hosts at any levels and permit secure data exchange simultaneously ([13], [14], and [15]). A Lock-Keeper system consists of three independent Single Board Computers (SBC): INNER, OUTER, and GATE, which are connected using a patented switch unit. The connection inside Lock-Keeper are restricted on the hardware layer by the switch unit that GATE can just be connected with only one partner, either INNER or OUTER. Since the OUTER component has to expose itself to the external world for providing Lock-Keeper supported services, the security of OUTER needs to be enhanced. Furthermore, normal operating system and application modules running on each Lock-Keeper component are another vulnerable points for being misused or even intentionally attacked. Deploying feasible IDS solutions on the main Lock-Keeper components is expected.

Therefore, we are motivated to combine the software based IDS solution with the hardware based Lock-Keeper system. An extensible IDS management architecture, which can easily aligned on the Lock-Keeper, is proposed in this paper. It includes several IDS sensors and a central management unit. A new design of the *Event Gatherer* for the management unit is provided through several *plugins*, i.e., *Sender*, *Receiver*, and *Handler*, which provide high flexibility in IDS deployment and support multiple types of communications, e.g., the file-based communication, network-based communication, etc. The IDMEF standard is implemented to represent and exchange the alarms. A standardized interface is designed to provide a unified view of alert reports. A prototype is implemented and practically deployed on the Lock-Keeper system.

The rest of the paper is organized as follows. Section 2 introduces some related works. Section 3 describes the proposed Lock-Keeper based IDS management architecture. Section 4 presents some experiments. Section 5 gives a short summary of our contributions and a brief outlook for the future work.

2 Related Works

In this section, we provide a short overview of existing works on IDS management. A practical IDS management system, called Prelude [5], is described as an example. The IDMEF standard is briefly introduced. Furthermore, a review of the high level security concept, Physical Separation, and its Lock-Keeper implementation is provided as well.

2.1 Overview of IDS Management

There are many previous works focusing on IDS management. In [16], a multi-agent based approach is described, which uses SOAP-based Web Services for communications. The system is based on a layered architecture and aims to increase the readiness of information as well as decrease the time of information access. The system described in [17] is proposed to integrate wireless sensors. It is capable of analyzing data with data mining techniques, such as rule-based classifiers, spatial data mining and clustering methods, as well as decision tree models. The decentralized approach described in [18] consists of a so-called distributed hash table architecture and focuses on identifying large scale distributed attacks, such as scanning, worms, or Denial-of-Service (DoS) attacks. TRINETR [19] is an intrusion detection alert management system, which provides alert aggregation, knowledge-based alert evaluation and correlation. The correlation procedure is supported by several knowledge bases (KB), such as vulnerability KB, network and host asset KB, etc.

Prelude is a distributed IDS product, which is capable of connecting other separate IDS sensors. It can be used to collect, normalize, sort, aggregate, correlate and report most of security-related events. Although *Prelude* is relatively robust and can ensure high performance, it reveals several problems concerning extensibility. The interface to the *Prelude* management system is built based on the *Prelude* library. The change of source code of the target IDS sensor is always required for any kinds of extensions. Additionally, the connection between the management system and the sensors requires a stable TCP connection. That means it is not suitable for deploying in loosely coupled environment, where connections are not so reliable, e.g., in wireless network.

As a new standard, Intrusion Detection Message Exchange Format (*IDMEF*) provides a unified format for communications between IDS sensors, response systems and management units. It specifies a data model to represent the exchange data in XML files. There are two basic message types defined in *IDMEF*: Alert and Heartbeat. The alert message is used to represent and exchange security related events, such as a detected attack. The heartbeat message is used to inform the management system that a sensor is still active, i.e., it is used for maintenance of the sensors. By offering such a convenient communication channel, *IDMEF* has been accepted by the IDS community. However, there are still several existing IDS implementations, which do not support this standard.

2.2 Physical Separation and Lock-Keeper

Based on the intuitive principle that "the ultimate method to secure a network is to disconnect it", the Lock-Keeper technology has been accepted as an efficient approach to guarantee the high-level security and prevent online network attacks by physically separating the protected hosts or networks ([13], [15]). Because of its simple idea and extensible architecture, the Lock-Keeper system can be easily and seamlessly integrated with any other security methods or solutions to provide thorough protection for most actual network-based applications. Currently,

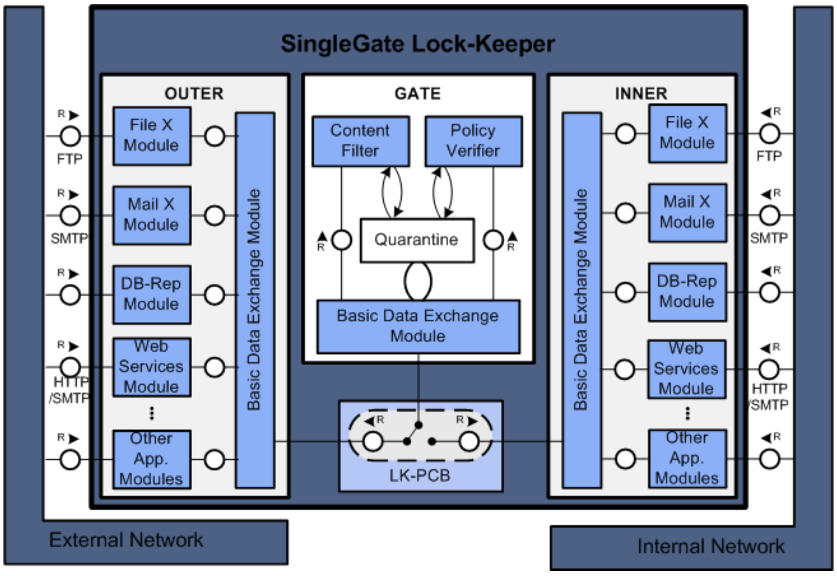


Fig. 1. Conceptual Architecture of the SingleGate Lock-Keeper

there are a lot of different "Physical Separation" implementations, such as Microsoft e-GAP-based Intelligent Application Gateway (IAG) [20], NRL Pump technology offered by U.S. Naval Research Laboratory [21], and DualDiode from Owl Computing Technologies [22], etc.

Lock-Keeper works as a sluice on the border of the protected network [15]. Because of such physical network separation, it can be guaranteed that hackers and malign data have no opportunities to break into the internal network by any means of online attacks. Currently, the commercial version of Lock-Keeper has already been developed and is now under the marketing extension by Siemens [14]. Here, we use a SingleGate Lock-Keeper system as an example to briefly explain what the Lock-Keeper is and how it works.

As shown in Fig. 1, a SingleGate Lock-Keeper system consists of three independent SBCs: INNER, OUTER and GATE. A patented switch unit, realized on a Printed Circuit Board (PCB), is used to autonomously control connections so that GATE can just be connected with only one partner, either INNER or OUTER. There are no ways to directly establish the connection between INNER and OUTER at any time. Besides these hardware components, there are also Lock-Keeper Secure Data Exchange (LK-SDE) software running in the Lock-Keeper system. LK-SDE software includes several application modules located on INNER and OUTER, which work as interfaces and provide popular network services to outside users, e.g., File eXchange (File-X) Module, Mail eXchange (Mail-X) Module, Database Replication (DB-Rep) Module and Web Services (WS) Module. Normal communication protocols, such as SMTP, HTTP, FTP, etc., are stopped and analyzed respectively by these application modules. Then, standard file-based Lock-Keeper Message Containers (LKMC) can be created to carry the data for respective services. These LKMCs will be transferred by "Basic Data Exchange Module" [15], which works based on the "Pull-and-Push" principle to avoid running any kinds of servers or server-similar programs on GATE so that there are no possibilities for outside hosts (even INNER and OUTER) to actively establish connection to GATE. In particular, because GATE is also a normal PC, it is possible to integrate Third-Party security software, e.g., virus scanning software, mail analysis tools, or content filtering methods, etc., into LK-SDE architecture, which help to check data traffic and prevent offline attacks, e.g., virus, malicious codes, etc.

3 Lock-Keeper IDS Management Architecture

To improve the accuracy of the intrusion detection effort, users need to use alerts from different audit sources. Therefore, it is necessary to propose an efficient management architecture, which can flexibly integrate different types of IDS sensors and automatically synthesize alerts. On the other hand, to enhance security of Lock-Keeper system, especially its main components, INNER and OUTER, a feasible distributed IDS solution is required. In this section, we present a new IDS Management architecture using the Lock-Keeper as the hardware base.

3.1 IDS Management: Architecture

As shown in Fig. 2, the proposed IDS Management Architecture consists of several *IDS Sensors* and an *IDS Management Unit*. The *IDS Management Unit* consists of four active components: *Event Database*, *Analysis Component*, *Remote Controller* and *Event Gatherer*.

IDS Sensors are responsible for detecting and reporting malicious behaviors. They are configured through the *Remote Controller*, which is responsible for

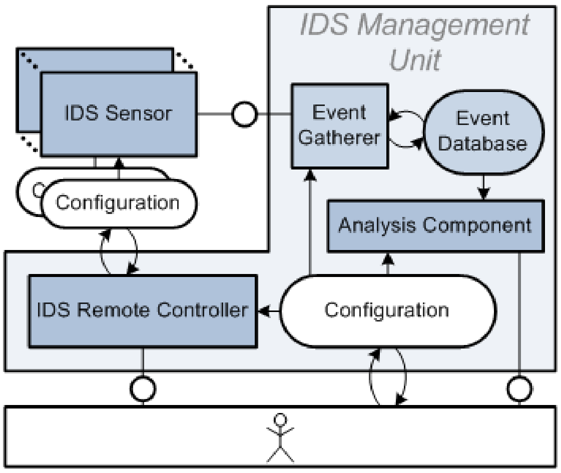


Fig. 2. IDS Management: Architecture

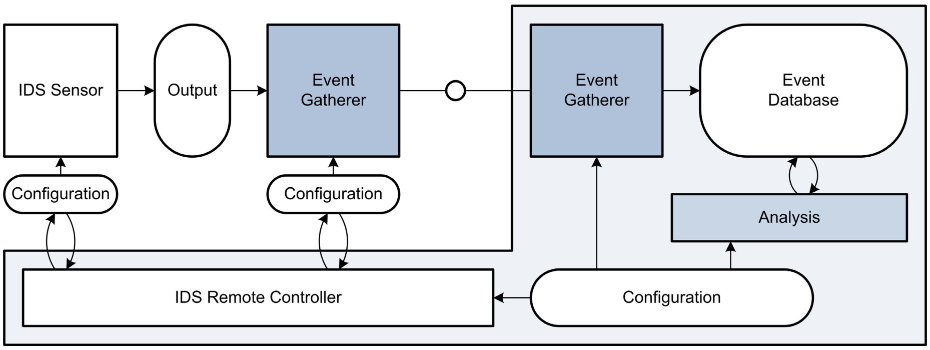


Fig. 3. IDS Management: Implementation

remotely configure and control all connected *IDS Sensors*. The *Remote Controller* can be a remote controlling and monitoring interface, where *Users* have possibilities to directly access the *IDS Management Unit*. The control can also be realized by changing configuration files of the core components. *Event Database* is a passive storage that holds information on all received events. All the events are made persistent in the *Event Database* storage and can be accessed through the *Analysis Component*, which is responsible for representing the gathered events as well as analyzing these events, e.g., correlating multiple events in complex attack scenarios. Such an analysis component might be the front end to the database within the IDS management unit. The *Event Gatherer* is an important component for collecting and unifying all events generated from connected *IDS Sensors*. Similarly, the *Event Gatherer* can be configured by *Users* through its configuration file.

Based on the proposed theoretical model of IDS management, we make a detailed design of the IDS management as shown in Fig. 3. A new *Event Gatherer* component is introduced on the *IDS Sensor* side. This gatherer is used to standardize the outputs from different sensors, which solves the problem of non-standardized sensor output. Furthermore, it realizes the logical communication between the sensor and the management component.

3.2 IDS Management: Implementation

Fig. 4 shows a detailed view on the architecture of the *Event Gatherer*. The gatherer is the core component of the event flow. It is a plugin engine capable of loading several *Event Receivers* and one *Event Sender*. A receiver processes a certain output, creates standardized events in IDMEF format and writes all events into a queue for further processing. Possible receivers are *Snort CSV Receiver*, *Samhain Syslog Receiver* or *IDMEF Tcp Receiver*. A sender reads standardized events from the queue and forwards them to a specific channel or storage, e.g., to a defined host and port via TCP connection or to a database with a certain schema. Possible senders are the *MySQL Database Sender* or the *IDMEF Tcp Sender*. To connect *Event Gatherers*, there are pairs of senders and

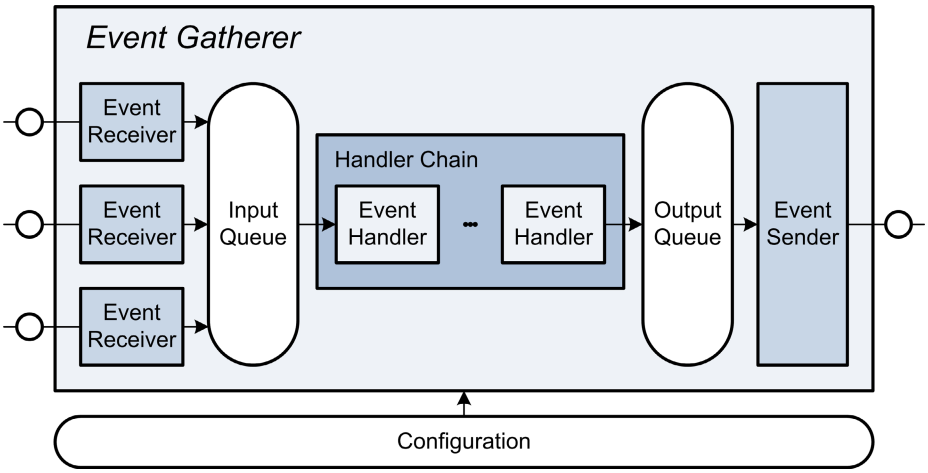


Fig. 4. Event Gatherer

receivers required, which can communicate with each other. This means that the sender creates output the receiver can process and understand. Possible pairs are the *IDMEF Tcp Sender and Receiver* or the *IDMEF File Sender and Receiver*. The senders and receivers are independent threads loaded at start up into the gatherer process. The gatherer can be configured to load several different receivers. This allows high flexibility in deployment and building intrusion detection scenarios.

3.3 A Lock-Keeper Based IDS Management Architecture

As shown in Figure 5, the proposed IDS management architecture is practically implemented on the Lock-Keeper system. We integrate Samhain and Snort on the OUTER and INNER component. On GATE, the Samhain and F-Secure Linux Security are deployed. The HIDS implementation Sambain is used to oversee the working state of each host. The NIDS snort is used to monitor network connections, which INNER/OUTER exposes to both sides. F-Secure Linux Security is used to do the application layer checking on the passing traffic, i.e., LKMC. Such known offline attacks as virus, worms, and other malicious codes, should be detected here.

There is an *Event Gatherer* running on each Lock-Keeper component to collect alerts from the connected sensors. The gatherer on OUTER includes the plugins: Samhain receiver, snort receiver, and IDMEF sender. The gatherer on GATE consists of the plugins: Samhain receiver, F-Secure receiver and IDMEF sender. The gatherer on INNER is running through the plugins: Samhain receiver, Snort receiver, IDMEF receiver, and SQL sender. The IDMEF sender on OUTER and GATE writes all alerts to a specified directory which is the interface to the Lock-Keeper SDE software. On INNER, the IDMEF receiver is listening on a specified directory where the alerts from OUTER and GATE will be put by the

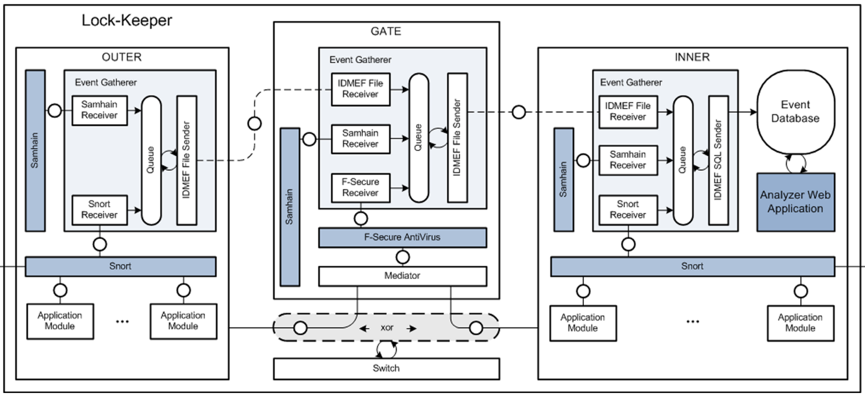


Fig. 5. Lock-Keeper based IDS Management Architecture

Lock-Keeper SDE software. The SQL sender is configured to write all the incoming alerts to a database for further processing.

As the host of the central IDS management unit, INNER has two additional important components: *Analyzer* and *Event Database*. The *analyzer* works on the event storage to provide simple event analysis and a unified view to end users. The database provides an easy way to store, query and process huge amounts of incoming alerts. Nevertheless, alerts can also be stored in other ways according to the practical requirements or environments, e.g., in a single XML file or in a specified directory with multiple alert files. A web application is built to connect *Analyzer* and *Event Database* as well as provide a user interface for displaying the unified view of events.

4 Experiment Results

As mentioned previously, we have two overall objectives: 1) improving the extensibility of IDS deployment and management, 2) enhancing the security of Lock-Keeper system. To evaluate the proposed Lock-Keeper based IDS management architecture and its implementation, we construct an experiment environment, as shown in Fig. 6. To test the functionality of each single integrated IDS sensor, e.g., Samhain, Snort, and F-Secure Linux Security, several simple attacks are executed within this environment.

A port scan towards the Lock-Keeper OUTER is executed from the *Attacker* by connecting to a port range of 1-1000. This port scan was performed using *Nmap* [23]. Similarly, this attack could also be carried out from inside towards INNER. The NIDS sensor, Snort, is supposed to detect this kind of attack.

Since Lock-Keeper also provides normal Email service, there are SMTP ports opened on OUTER. Therefore, attackers have the possibility to perform the SYN Flood DoS attack [24] towards OUTER. Snort on OUTER is responsible for recognizing this attack based on the preset threshold of the number of received SYN packets.

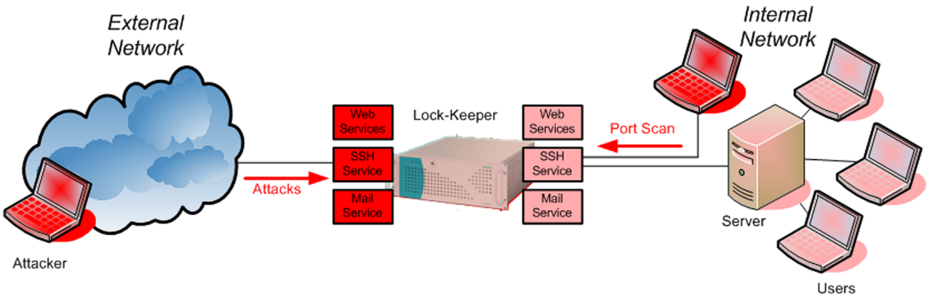
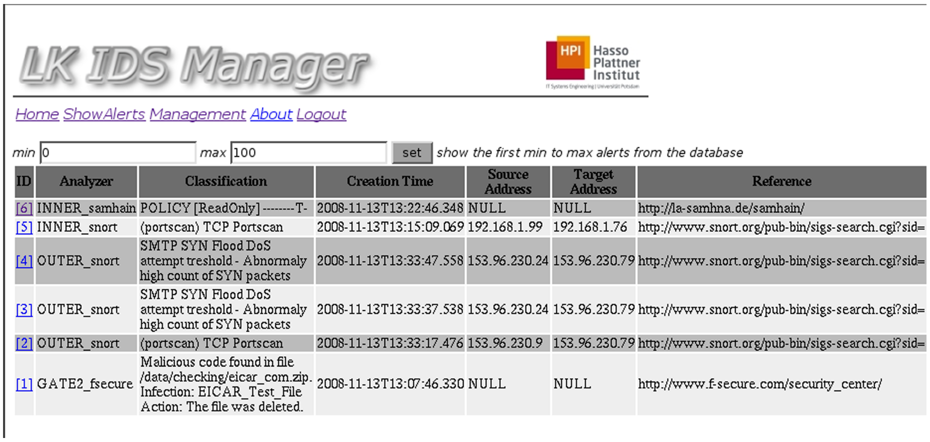


Fig. 6. Experiment: Testing Setup

To test the HIDS, we attempt to access and modify files in the sensitive directory on INNER, which is configured as "Read-Only". The deployed HIDS, Samhain, is used to identify such attacks.

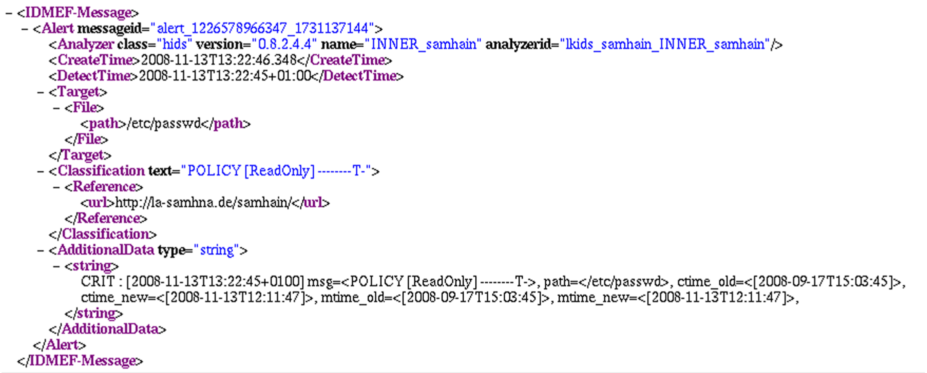
To test the functionality of the F-Secure Linux Server Security running on GATE, which is used as an IDS sensor to find and prevent application layer attacks, we create a scenario by sending an Email from the *Attacker* side to Lock-Keeper. Within this test Email, we attached the *eicar Anti-Virus Test File* [25]. The Lock-Keeper Mail-X module handles the Email and passes it to GATE. On GATE, the integrated F-Secure virus scanner is triggered by the malicious Email message. An alert is created in the format of IDMEF and later sent to the management system on INNER.

As shown on Fig. 7, F-Secure Linux Security on GATE detects the Email with the eicar file (see alert [1]), Snort on OUTER identifies the port scan and SYN Flooding (see alerts [2], [3] and [4]). The port scan is identified as well on INNER (see alert [5]). Besides, the Samhain on INNER also successfully detects the illegal access to the protected file *"/etc/passwd"* (see alert [6]). The


LK IDS Manager
HPI Hasso Plattner Institut
Home ShowAlerts Management About Logout
min 0 max 100 set show the first min to max alerts from the database

ID	Analyzer	Classification	Creation Time	Source Address	Target Address	Reference
[6]	INNER_samhain	POLICY [ReadOnly] --------T-	2008-11-13T13:22:46.348	NULL	NULL	http://la-samhna.de/samhain/
[5]	INNER_snort	(portscan) TCP Portscan	2008-11-13T13:15:09.069	192.168.1.99	192.168.1.76	http://www.snort.org/pub-bin/sigs-search.cgi?sid=
[4]	OUTER_snort	SMTP SYN Flood DoS attempt treshold - Abnormaly high count of SYN packets	2008-11-13T13:33:47.558	153.96.230.24	153.96.230.79	http://www.snort.org/pub-bin/sigs-search.cgi?sid=
[3]	OUTER_snort	SMTP SYN Flood DoS attempt treshold - Abnormaly high count of SYN packets	2008-11-13T13:33:37.538	153.96.230.24	153.96.230.79	http://www.snort.org/pub-bin/sigs-search.cgi?sid=
[2]	OUTER_snort	(portscan) TCP Portscan	2008-11-13T13:33:17.476	153.96.230.9	153.96.230.79	http://www.snort.org/pub-bin/sigs-search.cgi?sid=
[1]	GATE2_fsecure	Malicious code found in file /data/checking/eicar_com.zip. Infection: EICAR_Test_File Action: The file was deleted.	2008-11-13T13:07:46.330	NULL	NULL	http://www.f-secure.com/security_center/

Fig. 7. Experiment: Screenshot



```
- <IDMEF-Message>
  - <Alert messageid="alert_1226578966347_1731137144">
      <Analyzer class="hids" version="0.8.2.4.4" name="INNER_samhain" analyzerid="lkids_samhain_INNER_samhain"/>
      <CreateTime>2008-11-13T13:22:46.348</CreateTime>
      <DetectTime>2008-11-13T13:22:45+01:00</DetectTime>
    - <Target>
      - <File>
          <path>/etc/passwd</path>
        </File>
      </Target>
    - <Classification text="POLICY [ReadOnly] --------T-">
      - <Reference>
          <url>http://la-samhna.de/samhain/</url>
        </Reference>
      </Classification>
    - <AdditionalData type="string">
      - <string>
          CRIT : [2008-11-13T13:22:45+0100] msg=<POLICY [ReadOnly] --------T->, path=</etc/passwd>, ctime_old=<[2008-09-17T15:03:45]>,
          ctime_new=<[2008-11-13T12:11:47]>, mtime_old=<[2008-09-17T15:03:45]>, mtime_new=<[2008-11-13T12:11:47]>,
        </string>
      </AdditionalData>
    </Alert>
  </IDMEF-Message>
```

Fig. 8. Experiment: IDMEF Message of Alert Nr. 6

proposed *Event Gatherer* transforms the output of different IDS sensors into the standard IDMEF messages, e.g., the alert [6] shown in the Fig. 8. With the help of respective sender plugins and receiver plugins as well as the Lock-Keeper SDE software, alerts generated on OUTER and GATE are transferred to the central management unit on INNER and displayed on the end user interface.

The experiments have shown that our system works well, which exactly supports the claimed benefits on both improving extensibility of IDS management and enhancing Security of Lock-Keeper. Different types of IDS implementations, e.g., Snort, Samhain and F-Secure Linux Security, are seamlessly integrated into a loosely coupled environment with several distributed hosts. Standardized events are correctly created, transferred, analyzed and reported using the central IDS management unit. On the other hand, the deployed IDS sensors strongly increase the security level of Lock-Keeper, especially on detecting and preventing the network attacks on INNER/OUTER and application layer attacks towards the internal users.

5 Conclusion

The paper proposes an extensible architecture for deploying different types of IDS on the Lock-Keeper system. A central management unit for adding, configuring, operating and removing the different IDS sensors, is designed and deployed on the Lock-Keeper components. A high amount of security related events produced by all the integrated sensors could be synchronized, unified, analyzed and reported in the distributed platform. The known IDS standard, IDMEF, is realized in several kinds of plugins, e.g., *Handler*, *Sender* and *Receiver*, for storage and communication between the involved IDS sensors and the central management unit. The possibility to connect *Event Gatherers* in the architecture provides a high flexibility in deployment. A prototype implementation is presented and testified by experiments. The architecture is developed with focus on flexibility, extensibility and portability. However, there are still many open issues.

More assistant parts of the architecture need to be realized and integrated, e.g., an extended analysis of events. The analysis component should enable different modules and approaches for event correlation as well as extended management of events, such as sorting, filtering, tagging, etc. The number of available receivers and senders should be enriched. Performance is another important issue, which needs to be considered due to the complex integration and communication of heterogenous components. It is necessary to handle a certain amount of events within a tolerant time.

References

1. Snort IDS Website (1998-2009), http://www.snort.org/
2. Samhain IDS Website (2001-2009), http://www.la-samhna.de/samhain/
3. Bro IDS Website (2003-2009), http://www.bro-ids.org/
4. F-Secure Linux Security Website F-Secure Corporation (2006-2009), http://www.f-secure.com/linux-weblog/
5. Prelude IDS Website: PreludeIDS Technologies (2005-2009), http://www.prelude-ids.com/
6. Hallaraker, O., Vigna, G.: Detecting malicious javascript code in mozilla. In: Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2005, Washington, DC, USA, pp. 85–94 (2005)
7. Mahoney, M.V., Chan, P.K.: An analysis of the 1999 dARPA/Lincoln laboratory evaluation data for network anomaly detection. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 220–237. Springer, Heidelberg (2003)
8. Ramadas, M., Ostermann, S., Tjaden, B.C.: Detecting anomalous network traffic with self-organizing maps. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 36–54. Springer, Heidelberg (2003)
9. Northcutt, S., Novak, J.: Network Intrusion Detection: An Analyst's Handbook. New Riders Publishing, Thousand Oaks (2002)
10. Brumley, D., Newsome, J., Song, D., et al.: Towards automatic generation of vulnerability-based signatures. In: Proceedings of the, IEEE Symposium on Security and Privacy, SP 2006, Washington, DC, USA, pp. 2–16 (2006)
11. Leung, K., Leckie, C.: Unsupervised anomaly detection in network intrusion detection using clusters. In: Proceedings of the 28th Australasian conference on Computer Science, ACSC 2005, Darlinghurst, Australia, pp. 333–342 (2005)
12. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format, Internet Draft. Technical Report, IETF Intrusion Detection Exchange Format Working Group (July 2004)
13. Cheng, F., Meinel, C.: Research on the Lock-Keeper Technology: Architectures, Applications and Advancements. International Journal of Computer and Information Science 5(3), 236–245 (2004)
14. Lock-Keeper Website (2003-2009), http://www.lock-keeper.org/
15. Cheng, F., Meinel, C.: Lock-Keeper: A new implementation of physical separation technology. In: Paulus, S., Pohlmann, N., Reimer, H. (eds.) Securing Electronic Business Processes: Highligths of the Information Security Solutions Europe Conference, ISSE 2006, pp. 275–286. Friedrich Vieweg & Sohn Verlag (2006)

16. Claudino, E.C., Abdelouahab, Z., Teixeira, M.M.: Management and integration of information in intrusion detection system: Data integration system for IDS based multi-agent systems. In: Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology, WI-IATW 2006, Washington, DC, USA, pp. 49–52 (2006)
17. Derrick, E.J., Tibbs, R.W., Reynolds, L.L.: Investigating new approaches to data collection, management and analysis for network intrusion detection. In: Proceedings of the 45th Annual Southeast Regional Conference, SE 2007, New York, USA, pp. 283–287 (2007)
18. Zhou, C.V., Karunasekera, S., Leckie, C.: Evaluation of a decentralized architecture for large scale collaborative intrusion detection. In: Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management, IM 2007, Munich, Germany, pp. 80–89 (2007)
19. Yu, J., Reddy, Y.V.R., Selliah, S., et al.: TRINETR: An intrusion detection alert management system. In: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2004, Washington, DC, USA, pp. 235–240 (2004)
20. Intelligent Application Gateway (IAG) Website: Microsoft Corporation (2006-2009), http://www.microsoft.com/iag/
21. Kang, M.H., Moskowitz, I.S.: A pump for rapid, reliable, secure communication. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS 1993, New York, USA, pp. 119–129 (1993)
22. Menoher, J.: Owl computing product overview: Secure one-way data transfer systems. White Paper, Owl Computing Technologies, Inc. (2008)
23. Nmap Security Scanner Website (1997-2008), http://www.nmap.org/
24. Moore, D., Shannon, C., Brown, D.J., et al.: Inferring internet denial-of-service activity. ACM Transactions on Computer Systems (TOCS) 24(2), 115–139 (2006)
25. The Anti-Virus or Anti-Malware Test File: European Institute for Computer Antivirus Research (EICAR) (2008), http://www.eicar.org/

Ensuring Dual Security Modes in RFID-Enabled Supply Chain Systems

Shaoying Cai[1], Tieyan Li[2], Yingjiu Li[1], and Robert H. Deng[1]

[1] Singapore Management University, 80 Stamford Road, Singapore 178902
[2] Institute for Infocomm Research, 1 Fusionopolis Way, Singapore 138632
sycai@smu.edu.sg, litieyan@i2r.a-star.edu.sg,
yjli@smu.edu.sg, robertdeng@smu.eud.sg

Abstract. While RFID technology has greatly facilitated the supply chain management, designing a secure, visible, and efficient RFID-enabled supply chain system is still a challenge since the three equally important requirements (i.e., security, visibility, and efficiency) may conflict to each other. Few research works have been conducted to address these issues simultaneously. In this paper, we observe the different security requirements in RFID-enabled supply chain environments and differentiate the simplified model into two security levels. Accordingly, dual security modes are properly defined in our RFID setting. In the relatively secure environment, our system is set to the *weak security mode*, the tagged products can be processed in a highly efficient way. When in the *strong security mode*, our system guarantees a high level of security, while its efficiency is lower than that in the *weak security mode*. A set of RFID tag/reader protocols to facilitate the duel security modes are presented. Their security, visibility and efficiency are analyzed and compared with the relevant works.

1 Introduction

RFID systems consist of two main components: tags and readers. Tags are radio transponders attached to physical objects, while radio transceivers, or readers, query these tags for identifying information about the objects to which tags are attached. RFID technology, when combined with internet and networking technology, enables product information to be collected, integrated, shared, and queried at various levels (e.g., item, pallet, case, and container) in real time in a supply chain. RFID technology has been widely envisioned to have significant impact on the economy world-wide as an inevitable replacement of barcodes in the near future, which may facilitate the creation of secure, visible, and efficient supply chains. As a result, EPCglobal Network [2] is being formed to provide an open and standard interface to process RFID information in supply chain management.

The current EPCglobal Network standards depend entirely on honest supply chain partners to realize supply chain visibility. Few security mechanisms have been developed to ensure that the tracking services are confidential, verifiable, and accountable in the presence of realistic and malicious attacks, especially those coming from corrupted supply chain partners. Most of RFID technologies have focused on protecting a single RFID channel [5,6,7,8,9,10,11] without considering the relationship among supply chain parties. Such techniques cannot be directly used to protect the sharing of information among supply chain parties.

We develop a new security solution for RFID-enabled supply chain systems. In our solution, even a valid supply chain party, if not authorized, cannot track RFID tags after the ownership handover of the tags to other supply chain parties. On the other hand, an authorized party can have full supply chain visibility to track a tagged item in a supply chain. A root secret, shared between a trusted authority and each tag, is used to guarantee the anti-track and track/visibility properties. To achieve better efficiency, we classify the supply chain environments into two security levels. In an environment where insiders or outsiders can actively interact with a tag for the purpose of tracking, we set the system to *strong security mode* so as to maintain high security. In an environment where active attacks are not possible (e.g., within the territory of a supply chain party), we can set the system to *weak security mode* to achieve high processing speed. We use a binary switch on the tag to control the security modes. A set of RFID tag/reader protocols like tag reading protocol, security mode switching protocol and secret updating protocol, to facilitate the duel security modes are presented. We analyze the schemes in terms of security, visibility and efficiency. At last, our solution is compared with the relevant works.

The rest of this paper is organized as follows. In Section 2, we introduce a simplified system model and architecture for RFID-enabled supply chains. In Section 3, we present our protocols to protect RFID-enabled supply chain systems. In Section 4, we analyze our protocols in terms of security, visibility, and efficiency. In Section 5, we discuss the related works. Finally, Section 6 concludes this paper.

2 Model

Assumptions. In this paper, we focus on the attacks conducted on the wireless communications between RFID readers and tags. The adversaries can be either supply chain outsiders or insiders (i.e., dishonest supply chain parties). The adversaries are assumed to have the power to listen in communication channels, counterfeit as a valid supply chain party, to initiate, delete, modify, or transfer messages between RFID tags and readers. We do not consider the physical attacks, denial of service attacks, and side-channel attacks. We further assume that the communications between a supply chain partner and its RFID readers, and the communications between supply chain parties are secure, which can be protected by standard security techniques without limitation of resources.

Requirements. As discussed in existing RFID literatures, common security requirements of RFID tags include unlinkability, confidentiality, etc, nonetheless, applying RFID tags in supply chain environments introduces some unique needs such as supply chain visibility and extra efficiency. In the following, we list the details of the requirements for RFID system when deployed in the supply chain.

The security requirements of RFID-enabled supply chain are summarized in [12]. We increment the requirements to include forward and backward secrecy and de-synchronization resilience. The list of security requirements is given below: (i) Authoritative access: Only legitimate readers of an authorized party are allowed to identify and update a tag. (ii) Authenticity of tags: In a supply chain link, only legitimate RFID tags delivered by previous party will be accepted by the next party. (iii) Unlinkability: Weak unlinkability and strong unlinkability can be used to describe the security level of anti-tracking. Weak unlinkability requires that an unauthorized reader cannot link the responses of a tag interrogated before and after it is processed by an authorized party. Strong unlinkability requires that an unauthorized reader cannot link any two replies to the same tag. (iv) Forward and backward secrecy: If the communication between a tag and a party is compromised, it will not affect the security of the communication between the tag and any other party in the supply chain. (v) De-synchronization resilience: RFID communication protocol is resilient to the attacks that are targeted towards de-synchronizing a tag and a reader.

Besides security, supply chain visibility must be maintained in supply chain management. It means that the manager of the supply chain or any authorized party should be able to track the movement of RFID tags. The enhancement on supply chain visibility is the most attractive feature that RFID technology brings to the traditional supply chain management. It allows companies to track and monitor the progress of material flow without inefficient bar code scanning.

Due to mass product exchanging, the efficiency of RFID technology is crucial in supply chain management. Without incorporating security and visibility features, hundreds of read operations can be performed per second between a reader and tags. The processing speed should not be delayed by adding security and visibility features.

We stress that the three requirements (i.e., security, visibility, and efficiency) are equally important in supply chain environments. Our goal is to propose a practical solution for RFID-based supply chain systems under the three requirements with the lowest possible cost.

Architecture. Our solution involves four types of entities as shown in Figure 1: (i) a supply chain manager which is a trusted authority, denoted by TA; (ii) independent supply chain parties denoted by P_i; (iii) RFID readers inside a partner, which are collectively referred to as R_i controlled by its corresponding party P_i, and a back-end database referred to as D_i; (iv) RFID tags denoted by T_j.

The architecture we proposed is suitable for various types of supply chain structures [3] and compatible with contemporary EPCglobal network architecture [2]. In a third-party logistics (3PL) supply chain, TA can be the shipping company, which is specialized in handling the shipping issues in the supply chain. In the vendor managed inventory (VMI) supply chains, the vendor manages all the delivering of products; thus, it is straightforward for the vendor to take the role of TA. For the collaborative planning, forecasting, and replenishment

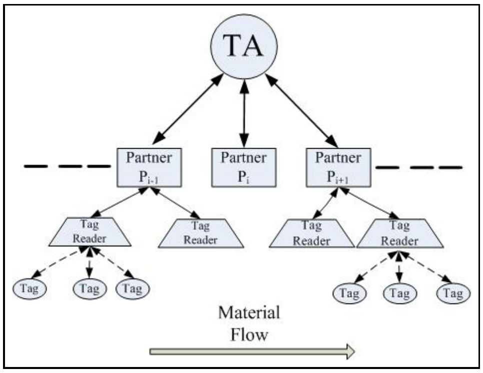


Fig. 1. A simplified RFID system architecture

(CPFR) supply chains, a supply chain hub is TA, which coordinates real-time sharing of supply chain information among supply chain parties. Finally, in supply network (SN), the situation is similar but more complex than in CPFR supply chains, where TA can be an existing supply chain hub or a dominant supply chain party.

3 Protocols

Since RFID tags will be used in vast numbers in supply chains, it is desired that the security design of the tags should be as cheap as possible. To achieve this goal, our protocols are designed to use passive tags that are equipped with pseudo-random number generators, standard XOR $\oplus$ and hash $H(\cdot)$ calculations.

A database is initialized by TA and sent to each supply chain party before the party can identify a batch of tags. With the help of the database, a supply chain party can switch security modes of RFID tags multiple times. Before a supply chain party sends a batch of tags to the next party in ownership handover, the current party needs to update the secret information in each tag. After ownership handover, the party can no longer identify the tags or track their movement.

3.1 Initialization

Tag initialization. Before the first supply chain party P_1 starts processing tagged products, TA will initiate three values (α_j, $\beta_{1\leftrightarrow j}$, $switch$) and embed them in each tag T_j in a secure manner.

- α_j is the tag root secret of length ℓ, which is fixed and shared between TA and the tag. The root secret is used to identify the tag uniquely.
- $\beta_{i\leftrightarrow j}$ is a temporary secret of length ℓ, which is shared between supply chain party P_i and tag T_j. The two parameters i and j of $\beta_{i\leftrightarrow j}$ denote the identity number of the supply chain party and the tag separately. This secret is initiated by TA to be $\beta_{1\leftrightarrow j}$. The temporal secret $\beta_{i\leftrightarrow j}$ will be updated by P_i to $\beta_{i+1\leftrightarrow j}$ before ownership handover to P_{i+1}.

- *switch* is a binary value used to indicate the security mode of a tag. This value is initiated to be 'on' for a strong security mode and it can be subsequently switched to 'off' for a weak security mode.

Database Initialization. Each party P_i maintains a database D_i in its local storage where each tuple in the database corresponds to a tag. D_i contains all RFID information with respect to a batch of tags except for tag root secrets α. D_i consists of five attributes $(\beta, x, p, s, switch)$, where $(\beta, switch)$ are defined the same as in tag initialization, (x, p, s) are defined below.

- Tag response x: Tag response to be received from the corresponding RFID tag (in weak security mode). When the tag is on strong security mode, the value of this attribute is set to NULL.
- Pointer p: An octet string containing an address where the business information relevant to the tag is stored. An alternative approach is to store information in this field directly. Obviously, it trades the storage cost for communication efficiency.
- Status s: Binary bit; $s = 1$ means that the corresponding RFID tag has been processed; otherwise not. We call an entry is unmarked if its value is zero.

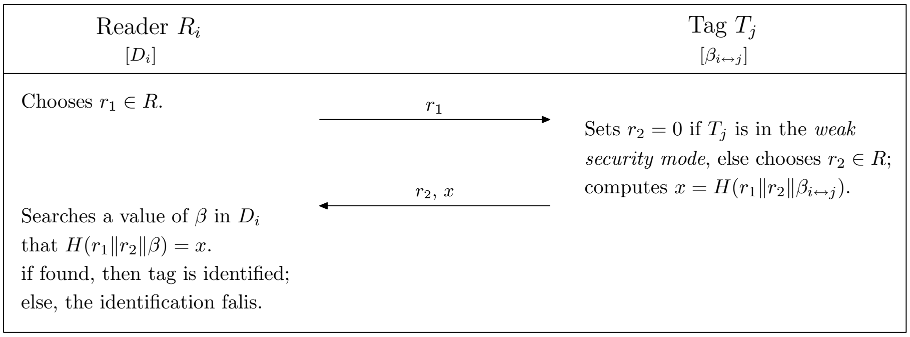
Reader initialization. When the P_i is to handover a batch of tagged products to P_{i+1}, it updates the tag temporal secrets and informs TA that P_{i+1} is the next party. Then TA will distribute the database D_{i+1} to P_{i+1} through a secure channel (e.g., SSL).

3.2 Tag Reading

Upon receiving D_i from TA and the tagged products from P_{i-1}, supply chain partner P_i can read any tag T_j in either the *strong security mode* if *switch* is on or in the *weak security mode* if *switch* is off.

1. $R_i \rightarrow T_j$: r_1, where r_1 is a random number of length ℓ generated by the reader.
2. $T_j \rightarrow R_i$: On receiving r_1, the tag sends the reply (r_2, x) to the reader, where $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$ and r_2 is a number of length ℓ. If the tag is in strong security mode, r_2 is a fresh random number generated by the tag; otherwise, $r_2 = 0$.

The tag reading protocol is illustrated in Figure 2. In the weak security mode, R_i can pre-compute the response of each tag x_j and store them in D_i. On receiving a response x , R_i identifies the tag if it can find a record $d_j = \langle \beta_{i\leftrightarrow j}, x_j, s_j, switch_j \rangle$ in D_i such that $x_j = x$ and $s_j = 0$. In the strong security mode, however, the response of each tag cannot be pre-computed due to the use of fresh tag random number; the value of x is set to NULL in each tuple of D_i in this case. Given a response (r_2, x), the reader identifies a tag by searching all of the unmarked$(s = 0)$ tuples in D_i until it finds a tuple d_j



Chooses $r_1 \in R$.

Sets $r_2 = 0$ if T_j is in the *weak security mode*, else chooses $r_2 \in R$; computes $x = H(r_1 \| r_2 \| \beta_{i\leftrightarrow j})$.

Searches a value of β in D_i that $H(r_1 \| r_2 \| \beta) = x$.
if found, then tag is identified;
else, the identification falis.

Fig. 2. Tag reading protocol

which satisfies $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$. For each identified tag T_j, the reader sets its status $s_j = 1$. If p_j is not empty, the reader can also obtain relevant product information following the pointer p_j.

The above reading process can be performed multiple times by supply chain partner P_i if necessary.

3.3 Security Mode Switching

Once the party P_i receives D_i from TA, it can change the security mode of its tags in different environments. Although the strong security mode is secure against both active attacks and passive attacks, the RFID tags can be processed more efficiently in the weak security mode in an environment where the active attacks are impossible. We design a security mode switching protocol below.

1. $R_i \rightarrow T_j$: To update the security mode of tag T_j, the reader chooses a fresh random number r_3 and computes $a = \beta_{i\leftrightarrow j} \oplus r_3$, and $b = H(switch_0||a||r_3)$, where $switch_0$ is the new value of $switch$. The reader then sends the triple $(switch_0, a, b)$ to the tag.
2. $T_j \rightarrow R_i$: When tag T_j receives $(switch_0, a, b)$, it computes $r_3 = \beta_{i\leftrightarrow j} \oplus a$, and checks whether $b = H(switch_0||a||r_3)$ holds; if so, it updates $switch = switch_0$. After update of switch value, the tag T_j will send a confirmation (r_2, x) back to the reader, where r_2 is generated based on the switch value and $x = H(r_2||r_3||\beta_{i\leftrightarrow j})$.
3. R_i: Upon receiving (r_2, x), the reader R_i confirms the update of $switch$ by checking whether $x = H(r_2||r_3||\beta_{i\leftrightarrow j})$.

The protocol is also illustrated in Figure 3. Since a tag will send a confirmation to the reader after security mode update, any failure can be detected by the reader.

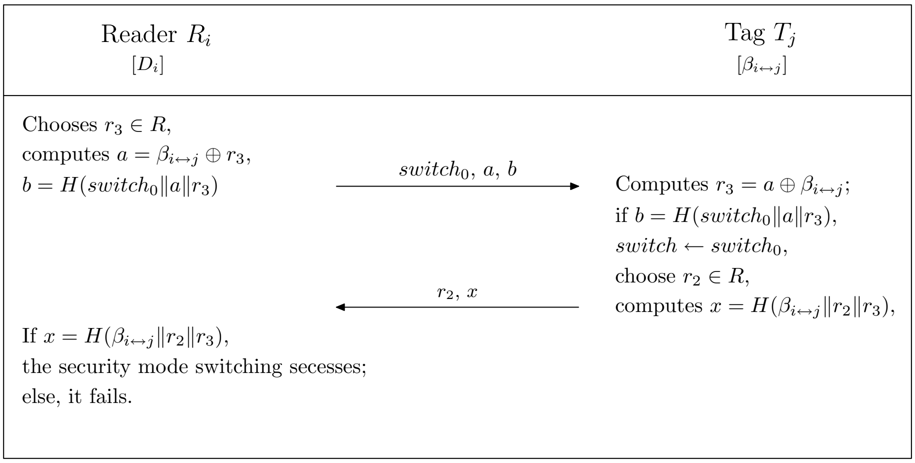

Reader R_i
$[D_i]$
Tag T_j
$[\beta_{i\leftrightarrow j}]$
Chooses $r_3 \in R$,
computes $a = \beta_{i\leftrightarrow j} \oplus r_3$,
$b = H(switch_0 \| a \| r_3)$
$switch_0$, a, b
Computes $r_3 = a \oplus \beta_{i\leftrightarrow j}$;
if $b = H(switch_0 \| a \| r_3)$,
$switch \leftarrow switch_0$,
choose $r_2 \in R$,
computes $x = H(\beta_{i\leftrightarrow j} \| r_2 \| r_3)$,
r_2, x
If $x = H(\beta_{i\leftrightarrow j} \| r_2 \| r_3)$,
the security mode switching secesses;
else, it fails.

Fig. 3. Security mode switching protocol

3.4 Ownership Handover

Ownership handover is performed between two supply chain parties P_i and P_{i+1} with RFID tags in weak security mode without TA's active involvement. Before the handover, P_i will update the temporal secrets of its tags and informs TA, who will send D_{i+1} to P_{i+1} in a secure manner. Note that the database D_{i+1} is not needed during the handover process.

In order to prevent the tagged products from being tracked by party P_i after ownership handover, the tag's temporary secret must be updated from $\beta_{i\leftrightarrow j}$ to $\beta_{i+1\leftrightarrow j}$. This updating process is performed by P_i before handover. Without being appropriately updated, a tag will not be accepted by P_{i+1} in the handover process (see below). The update of tag temporal secrets guarantees that only P_{i+1} can access the tags although the update is conducted by P_i. This is under the assumption that P_i cannot get access to tag root secrets nor the new database D_{i+1}. The temporary secret updating protocol is shown in Figure 4. During ownership handover, both parties need to agree upon a list of the tagged products and report the agreed list to TA for supply chain visibility.

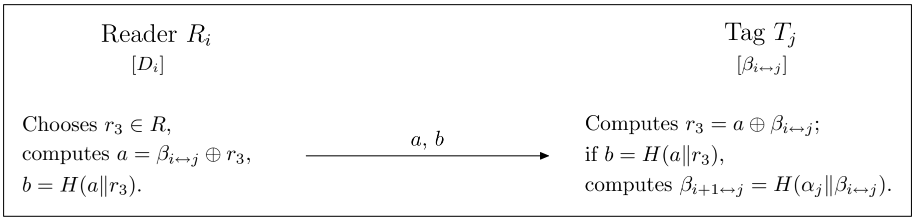

Reader R_i
$[D_i]$
Tag T_j
$[\beta_{i\leftrightarrow j}]$
Chooses $r_3 \in R$,
computes $a = \beta_{i\leftrightarrow j} \oplus r_3$,
$b = H(a \| r_3)$.
a, b
Computes $r_3 = a \oplus \beta_{i\leftrightarrow j}$;
if $b = H(a \| r_3)$,
computes $\beta_{i+1\leftrightarrow j} = H(\alpha_j \| \beta_{i\leftrightarrow j})$.

Fig. 4. Temporary secret updating protocol

During ownership handover, both parties need to agree upon a list of the tagged products and report the agreed list to TA for supply chain visibility.

1. For a batch of tagged products to be handed over to P_{i+1}, P_i performs tag reading protocol with the same random number r_1 of length ℓ and records a list L of responses x_j from all tags in the batch, where $x_j = H(r_1||r_2||\beta_{i+1\leftrightarrow j})$, $r_2 = 0$. Then, P_i sends r_1 to P_{i+1}.
2. Upon receiving r_1, P_{i+1} performs the reading protocol with the same random number r_1 and records a list L' of all responses x'_j from all tags in the batch, where $x'_j = H(r_1||r_2||\beta_{i+1\leftrightarrow j})$.
3. P_i and P_{i+1} compares the two list L and L'. If the lists match, then both sign on the matched list with the current time-stamp and keep a copy of the signed list. Party P_i sends the signed list to the TA for supply chain visibility. If the two lists do not match, the two parties will settle the disagreement till they reach an agreement. After the handover process, P_{i+1} should switch the tags into the strong security mode if P_i is still around.

The ownership handover process is illustrated in Figure 5. Note that P_i is responsible to report to TA since it is for P_i's interest to finalize the handover process as early as possible. TA is responsible to coordinate the handover process and manage the supply chain visibility accordingly. Our system remains secure even the tags are set to the weak security mode in the ownership handover process. The reason is that the tagged products remain static in this process; there is no point to track tags while they are not moving. After ownership handover, the tags are in P_{i+1}'s control, who will keep the tags secure by switching to appropriate security modes. If P_i has not updated some tags appropriately before ownership handover due to de-synchronization attacks or communication errors, both parties will detect the mismatch; P_i can re-update the tags to facilitate the handover. Therefore, our solution has the de-synchronization resilience property.

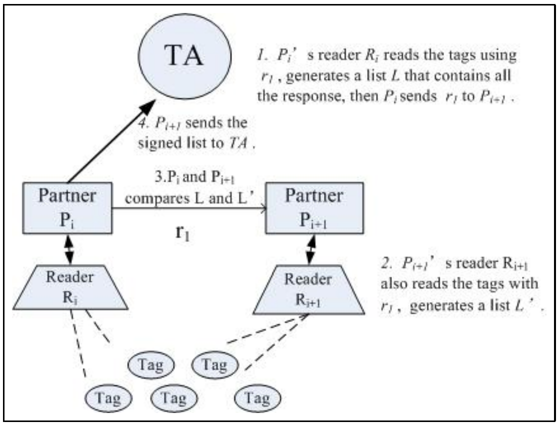

TA
1. P_i' s reader R_i reads the tags using r_1, generates a list L that contains all the response, then P_i sends r_1 to P_{i+1}.
4. P_{i+1} sends the signed list to TA.
3.P_i and P_{i+1} compares L and L'
Partner P_i
Partner P_{i+1}
r_1
Reader R_i
Reader R_{i+1}
2. P_{i+1}' s reader R_{i+1} also reads the tags with r_1, generates a list L'.
Tag
Tag
Tag
Tag
Tag

Fig. 5. Ownership handover process between P_i and P_{i+1}

4 Analysis

In this section, we analyze our protocols with respect to the requirements of security, visibility, and efficiency. We summarize our analysis results in the form of statements. Proofs and explanations of the statements will be given in the full version of this paper.

STATEMENT 4.1 (Authoritative access to RFID tags). *Only a valid reader with a tag's temporary secret authorized by TA is able to conduct reading and updating on the tag successfully.*

STATEMENT 4.2 (Authenticity of tags). *Given two numbers r_1 and r_2, the probability for an adversary without the knowledge of $\beta_{i\leftrightarrow j}$ to find a valid response x such that $x = H(r_1||r_2||\beta_{i\leftrightarrow j})$ holds is $\frac{1}{2^\ell}$, where ℓ is the length of $\beta_{i\leftrightarrow j}$.*

STATEMENT 4.3 (Weak unlinkability). *Given a response x_1 from a tag prior to being processed by party P_i and a response x_2 from a tag after being processed by P_i, it is computationally infeasible for a rogue reader to determine whether x_1 and x_2 is from the same tag.*

STATEMENT 4.4 (Strong unlinkability). *A tag is strong unlinkable in the strong security mode. It is also strong unlinkable in the weak security mode in an environment of no active attacks.*

STATEMENT 4.5 (Forward and backward secrecy). *If the protocol communication between a tag and a reader is compromised in certain party, it will not affect the security of the protocol communication between the tag and the reader in any other parties.*

STATEMENT 4.6 (De-synchronization resilience). *The temporary secret updating protocol and the security mode switching protocol in our security solution are resilient to de-synchronization attacks.*

STATEMENT 4.7 (Visibility). *While unauthorized entities are prevented from tracking the movement of material flow, authorized entities have access to the information about where and when a tag is processed.*

STATEMENT 4.8 (Efficiency). *In the weak security mode, the time complexity for an authorized reader to identify a batch of n tags is $O(n \log n)$. In the strong security mode, the time complexity is $O(n^2)$.*

The bottleneck of most RFID-enabled supply chain systems including ours is the process of identifying a large number of tags by each reader. According to [16,17], we roughly estimate that it takes about 210 CPU cycles of a Pentium CPU to perform a hash function (e.g., SHA-1) for digesting a 128-bit message and about 40 CPU cycles per sort operation for merge-sort or quick-sort algorithms. We assume that there are 2^{20} tags in each batch, and a 1-GHz Pentium machine is used in each reader's servant computer. In the weak security mode, it requires about 800ns in database search for identifying each tag. In the strong security mode, however, the batch size is better below 10^4 so that a reader can identify about 500 tags per second. Since an RFID reader usually can perform about 100 times reading, the speed of searching tags in database is sufficiently fast enough as it is higher than 100 tags per second.

5 Related Work

A bundle of research papers addressing RFID security and privacy problems have been published (refer to [4] for a detailed literature survey) in recent years. Among them, tens of (privacy enhanced) tag/reader mutual authentication protocols have been proposed in the literature such as the HB family of RFID protocols [5,6], symmetric cipher based protocols [7,8,9], and lightweight primitive based protocols [10, 11]. Unfortunately, all of these RFID protocols have focused on protecting tag to reader communication in a single domain, where no cross-domain relationship is considered among various supply chain parties. Therefore, their techniques cannot be directly used to ensure the information sharing protocols interacting multiple supply chain parties.

In this section, we compare our solution with the most related works [12,14,13] on RFID authentication protocols in supply chain environments. We realize that these works might have been attacked in some way [15], their original ideas are still meritable and worth being reviewed. A summary of our comparison is given in Table 1.

In [12], Li and Ding proposed a de-centralized solution for secure RFID communications in supply chains. In their solution, an access key is shared between each tag and each supply chain party. The access key of a tag can be updated by the current supply chain party before the tag is handed over to the next supply chain party. Since the updated access key is shared between the current supply chain party and the next supply chain party, their solution is vulnerable to insider attacks without backward or forward secrecy. Their solution is similar to our solution in the weak security mode in a sense that only weak unlinkability is provided. The time complexity of their solution is also similar to our solution in the weak security mode, which is $O(n \log n)$ for processing each batch of tags. Since there is no trusted authority involved in their solution, the supply chain visibility should be maintained by each party's database in a distributed manner.

Juels, Pappu, and Parno proposed an interesting solution for secure RFID-enabled supply chains [13]. In their work, a secret sharing method is used to break a secret key to multiple shares, with each share stored in a single tag along with the cipher of the tag id encrypted with the secret key. An authorized supply chain party, which is supposed to get access to a large number of tags can collect enough shares to recover the secret key, and thus decrypt the tags' IDs. An adversary is assumed to have limited access to the tags; thus, he or she cannot recover the secret key nor decrypt any tags' IDs. It is clear that this solution does not have any unlinkability feature. Anyone can track the movement of a tag even if it is encrypted. The advantage of this solution is that it can be directly used with the current EPC Gen2 tags [1] without any cryptographic extensions; therefore, the cost of tags is apparently lower than other solutions which have to incorporate hash and random number generation computations in tags.

Song proposed an RFID ownership transfer protocol recently [14]. In her solution, a supply chain party and a tag share a couple of secrets (t, s), where $t = h(s)$. The tag stores the value t, and the reader authorizes itself to the tag

Table 1. Comparison of our solution with three other solutions

	Unlinkability (anti-tracking)	Visibility (handover)	Efficiency (tag search)	Cost (tag)
[12]	Weak	Distributed	Batch process	Moderate
[13]	Null	Distributed	Decryption	Low
[14]	Strong	Distributed	Tag by tag	Moderate
Our solution	Strong	Centralized	Switch	Moderate

by proving its possession of s. In ownership handover, the current party must be online to help the next party to identify a tag; then, the current party will send the tag secret to the next party, which will use the secret information to update the tag secret to a new value. To provide strong unlinkability, each tag will generate its reply based on a fresh random number, which is similar to our solution in the strong security mode. The weakness of this solution is its low efficiency, especially in the handover process which takes $O(n^2)$ time for processing n tags one by one.

Comparing to the above works, our proposal solution is the only solution that involves a trusted authority. Therefore the supply chain visibility can be easily maintained in a centralized manner. On the one hand, our solution provides strong unlinkability under the assumption that the weak security mode is used in a relative secure environment of no active attacks. On the other hand, it can switch to the weak security mode for higher efficiency in tag reading. It thus provides higher efficiency in certain environment without downgrading the security features. In terms of tag cost, our solution is similar to [12, 14] as it involves random number generation and hash computations in tags. Note that our solution is suitable for the RFID tags of cost around US$0.5 and RFID reader of cost around US$1000. Such RFID readers and tags are currently available in the market and their costs are affordable in supply chain management at container, pallet, or case level (probably not at the item level).

6 Conclusion

In this paper, we investigate the security, visibility, and efficiency issues for RFID-enabled supply chain systems. High efficiency is particularly desirable in RFID-enabled supply chains since a large quantity of tagged products are routinely processed and exchanged among multiple supply chain parties such as suppliers, manufacturers, distributors, and retailers. In order to enhance the efficiency of a RFID-enabled supply chain system without sacrificing its security, we distinguish the environments into two secure levels. In a relatively secure environment with no active attacks, our RFID system can be set to the weak security mode so as to provide high processing speed. While in a relatively less secure environment that is exposed to active attacks, our RFID system can be switched to the strong security mode so as to maintain strong unlinkability. In the future, we are interested in investigating the scalability of our solution and verifying its practicality in real EPCglobal network or simulated environments.

Acknowledgment. This work is partly supported by A*Star SERC Grant No. 082 101 0022 in Singapore.

References

1. EPCglobal Inc., E.P.C. Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.1.0. EPCglobal Standards (October 2007)
2. EPCglobal Inc., Architecture Framework Standard v1.0
3. Liu, E., Kumar, A.: Leveraging information sharing to increase supply chain configurability. In: Twenty-Fourth International Conference on Information Systems, pp. 523–537 (2003)
4. Juels, A.: RFID Security and Privacy: A Research Survey. IEEE Journal on Selected Areas in Communications 24(2), 381–394 (2006)
5. Juels, A., Weis, S.A.: Authenticating pervasive devices with human protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
6. Bringer, J., Chabanne, H., Emmanuelle, D.: HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In: IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006, Lyon, France. IEEE Computer Society Press, Los Alamitos (2006)
7. Sarma, S., Weis, S., Engels, D.: RFID systems and security and privacy implications. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 454–469. Springer, Heidelberg (2003)
8. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic approach to privacy-friendly tags. In: Proc. of RFID Privacy Workshop (2003)
9. Aigner, M., Feldhofer, M.: Secure Symmetric Authentication for RFID Tags. Telecommunication and Mobile Computing (March 2005)
10. Vajda, I., Buttyan, L.: Lightweight authentication protocols for low-cost RFID tags. In: Proc. of UBICOMP 2003 (2003)
11. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: LMAP: A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags. In: Proc. of 2nd Workshop on RFID Security (July 2006)
12. Li, Y., Ding, X.: Protecting RFID Communications in Supply Chains. In: ASIACCS 2007: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, pp. 234–241. ACM, Singapore (2007)
13. Juels, A., Pappu, R., Parno, B.: Unidirectional key distribution across time and space with applications to rfid security. In: 17th USENIX Security Symposium, pp. 75–90 (2008)
14. Song, B.: RFID Tag Ownership Transfer. In: Conference on RFID Security (RFIDsec 2008), Budapest, Hungary (July 2008)
15. van Deursen, T., Radomirovic, S.: Attacks on RFID Protocols. Cryptology ePrint Archive: Report 2008/310 (2008)
16. Menascé, D.: Security performance. IEEE Internet Computing 7(03), 84–87 (2003)
17. Li, X., Zhang, X., Kubricht, S.: Improving memory performance of sorting algorithms. J. Exp. Algorithmics 5, 3 (2000)

Achieving Better Privacy Protection in Wireless Sensor Networks Using Trusted Computing

Yanjiang Yang[1], Robert H. Deng[2], Jianying Zhou[1], and Ying Qiu[1]

[1] Institute for Infocomm Research, Singapore 119613
[2] School of Information Systems, Singapore Management University

Abstract. A wireless sensor network (WSN) is an ad-hoc wireless network composed of small sensor nodes deployed in large numbers. Sensor nodes are usually severely resource limited and power constrained. Security enforcement in WSNs is thus a challenging task. In this paper we propose a clustered heterogeneous architecture for WSNs, where high-end cluster heads are incorporated, and they are further equipped with trusted computing technology (TC). As such, the cluster heads act as trusted parties, and are expected to help effectively address privacy issues in WSNs. As concrete examples, we discuss in details how user query privacy and source location privacy can be better protected.

1 Introduction

A WSN consists of a large number of sensor nodes collecting environmental data. The nodes are usually severely constrained in computation, storage, communication and power resources. When deployed in critical applications, mechanisms must be in place to secure a WSN. Security issues associated with WSNs can be categorized into two broad classes [24]: content-related security, and contextual security. *Content-related security* deals with security issues related to the content of data traversing the sensor network such as data secrecy, integrity, and key exchange. Numerous efforts have recently been dedicated to content-related security issues, such as secure routing [16,17,22,32], key management and establishment [5,8,10,25,26,34,37], and data aggregation [6,15,27,33]. In many cases, it does not suffice to just address the content-related security issues. Suppose a sensitive event triggers a packet being sent over the network; while the content of the packet is encrypted, knowing which node sends the packet reveals the location where the event occurs. *Contextual security* is thus concerned with protecting such contextual information associated with data collection and transmission.

It is commonly acknowledged that the resource-constrained nature of sensor nodes makes security enforcement in WSNs a challenging task. The majority of the above mentioned efforts attempted to solve security issues in *homogeneous* WSNs where all sensor nodes have the same capabilities. However, both theoretical and empirical studies have concluded that homogeneous WSNs are not scalable. Through a theoretical analysis it is shown in [12] that the throughput of each sensor node decreases rapidly as the numbers of nodes increases; in addition, it is demonstrated via simulations in [9] that, as the traffic becomes heavy, the overhead due to routing and control consumes a large portion of the available bandwidth.

Our Contribution. To improve the effectiveness of security enforcement, we present a trusted computing (TC)-enabled clustered heterogeneous WSN architecture, composed of not only resource constrained sensor nodes, but also a number of more powerful high-end devices acting as *cluster heads*. Compared to sensor nodes, a high-end cluster head has higher computation capability, larger storage, longer power supply, and longer radio transmission range, and it thus does not suffer from the resource scarceness problem as much as a sensor node does. A distinct feature of our heterogeneous architecture is that cluster heads are equipped with trusted computing (TC) technology, and in particular a TCG-compliant TPM (Trusted Platform Module) [35] is embedded into each cluster head. The TC-enabled cluster heads act as online *trusted parties*; security enforcement is thus expected to be substantially simplified and improved. We further substantiate the above assertion by demonstrating how trusted cluster heads help to provide elegant solutions for two important contextual security problems, user query privacy [7] and source location privacy [24,28,29,31].

2 TC-Enabled Heterogeneous Architecture for WSNs

We partition a WSN into a number of *clusters*. A high-end device is placed into each cluster, acting as the *cluster head*. In contrast to sensor nodes, high-end cluster heads have relatively higher computation capability, larger storage size, and longer radio range. They also have longer power supply, and in some circumstances they can even be line-powered. Therefore unlike sensor nodes, cluster heads do not drastically suffer from the resource scarceness problem. Depending on applications, hardware capabilities of cluster head may vary from that comparable to a bluetooth device to that of a high end PDA. The introduction of high-end cluster heads into a WSN makes the once homogeneous network *heterogeneous*. The general heterogeneous architecture is depicted in Figure 1.

Downlink communication (from base station to sensor nodes) and uplink communication (from senor nodes to base station) in the architecture are asymmetric. Messages broadcast by the base station can directly reach sensor nodes, whereas messages sent by a sensor node need to be forwarded by its corresponding cluster head. As a result, uplink communication follows a hierarchical manner and consists of intra-cluster and inter-cluster communications, respectively. At the level of *intra-cluster communication*, a cluster head acts as a gateway for the sensor nodes within the cluster; a sensor node can reach the cluster head directly, or through a *short* multi-hop channel. *Inter-cluster communication* is concerned with communication among cluster heads and the base station. Since inter-cluster communication does not suffer from the limits upon sensor nodes, it can utilize more advanced communication infrastructure, e.g. 802.11. Consequently, the heterogeneous architecture is expected to enormously improves the overall system performance and lifetime of the network.

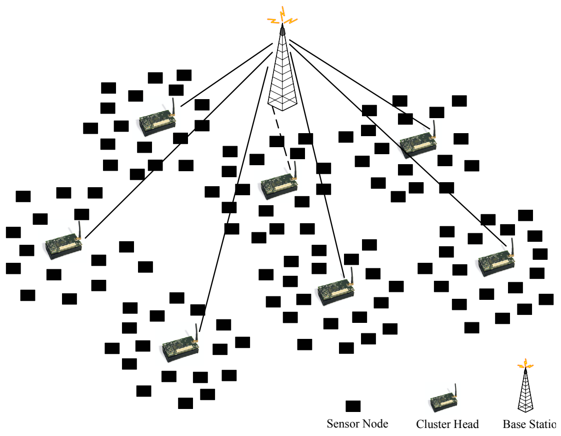


Fig. 1. Heterogeneous Wireless Sensor Network

A feature distinguishing our architecture from other similar heterogeneous ones (e.g., [18,19,36]) is that we equip each cluster head with a TPM, so as to simplify and improve security enforcement in WSNs. The rationale is that under the auspices of TPM, cluster heads can act as online *trusted parties* in enforcing security mechanisms. Relevant entities, e.g., the base station or users who query the network (see Section 4.1), can ascertain the trustfulness of cluster heads by means of attestation. We notice that the authors of [13,21] also discussed the idea of applying trusted computing technology to equipping more powerful cluster heads within WSNs, and there should be no distinction in the network architecture between ours and theirs. However, in their proposals, sensor nodes are responsible for verifying the platform state of the cluster heads through attestation, whereas our proposal is that sensor nodes never challenge their cluster heads for attestation (we believe they actually do not afford to do so), and they simply trust their respective cluster heads. Our argument relies on the fact that it is the end users (or the owner) of the network who should be concerned with the trustfulness of the cluster heads and the network.

3 Addressing Privacy Issues in WSNs

To show the effect of the trusted cluster heads on security enforcement, we present solutions to two important contextual security problems in WSNs: user query privacy and source location privacy. Compared to existing solutions in [7,24,28,29,31], our schemes achieve better privacy and higher efficiency.

Adversary Model. For the schemes presented below, we assume a *global eavesdropper* who eavesdrops on the entire network. In particular, the adversary is able to watch all the traffic traversing the sensor network. An adversary of this nature poses a great threat, especially to contextual security in WSNs. The global adversary in our consideration is much stronger than the *local eavesdropper* assumed by other work such as [7,24,29,31]. A local eavesdropper only has knowledge on the sensor node it stays with, and it can only trace communication hop by hop. We must stress that the privacy offered by the schemes in [7,24,29,31] is immediately broken under the global adversary, because through global eavesdropping the adversary can always know which node(s) initiates the targeted data transmission.

3.1 Achieving User Query Privacy

Problem Statement. WSNs are often deployed to provide services to other users than the network owner [7]. Users are allowed to query a network to get sensed data from particular areas. In such a scenario, a user may wish to protect her "areas of interest" from being disclosed to other users or even the network owner. User query privacy is thus concerned with the following problem: suppose a user queries the network, intending to get the sensed data in cluster c_i, a user query privacy scheme ensures that the user ends up getting the desired data, but the adversary does not learn c_i by observing the communication.

Our Algorithm

Network Model. We support roaming users querying a wireless sensor network. The network follows the heterogeneous architecture proposed earlier: the whole network is partitioned into a set of n clusters, c_1, c_2, ..., c_n, where c_i is the identifier of the ith cluster; each cluster is grouped around a TC-enabled cluster head and we denote ch_i the cluster head in c_i. A user who desires to query the network first contacts the nearest cluster head within her proximity, through which she will issue queries. This cluster head is called *access point*. Taking Figure 2 as an example, ch_1 of c_1 is the access point for the user. Once the access point is determined, all the other cluster heads need to dynamically establish routing pathes to the access point. The single-path routing in [24] can achieve this objective. The single-path routing uses a greedy algorithm where each node

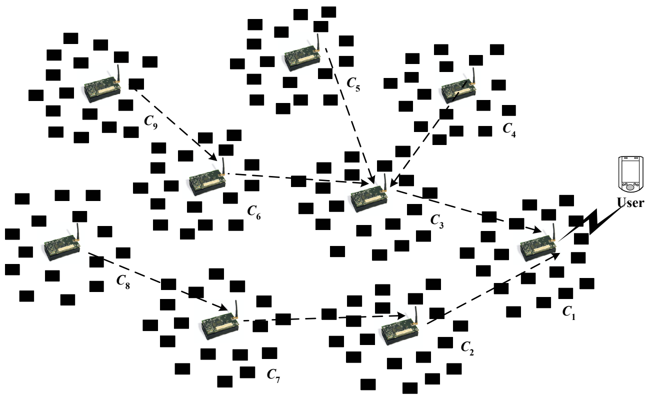


Fig. 2. User Querying Scenario

chooses one of its neighboring nodes that is nearest to the sink (i.e., the access point in our case) as its *forwarding node* in data transmission, and the node itself is a *dependent node* of the forwarding node. As a result, the routing pathes form a tree rooted at the access point. In Figure 2, the routing pathes are shown in dotted lines. After the routing pathes are formed, every cluster head knows which node is its forwarding node, and which nodes are its dependent nodes.

Assumption. To obtain services from a WSN, a user is assumed to have a certain means to authenticate to the WSN. Also, a TC-enabled cluster head can authenticate to users using the AIK of its embedded TPM/MTM. Therefore, the access point and the querying user can accomplish mutual authentication, based on which we assume the two entities share a secret key for semantic secure symmetric encryption. Further, it is also easy for each cluster head to get this secret key from the access point, since they can clearly authenticate each other with the help of their respective AIKs. We denote $\mathcal{E}_{ch_i}(.)$ and $\mathcal{D}_{ch_i}()$ the encryption and decryption, respectively, by ch_i using this secret key. We also assume each cluster head shares a secret *cluster key* (for semantic secure symmetric encryption) with all sensor nodes in its cluster. Several well studied key exchange schemes [26] can achieve this objective. $\mathcal{CE}_{c_i}(.)$ and $\mathcal{CD}_{c_i}()$ denote the encryption and decryption, respectively, using the cluster key of c_i.

Algorithm Overview. A straightforward way to achieve query privacy is that every cluster head sends encrypted data to the access point, who then forwards only the data desired by the user. This however unnecessarily wastes communication bandwidth among cluster heads. Even for this straightforward method, we should still be very cautious not to leak information about the queried cluster from the size of the data returned to the user. More specifically, clusters normally have different number of sensor nodes, so data from different clusters are likely to have different lengthes. Let us suppose the data from each sensor node forms a packet for simplicity. The total number of packets from a cluster equals the number of sensor nodes. Without privacy treatment, the number of packets eventually returned to the user by the access point would clearly indicate the cluster from which the data originates. A method to fix this problem is that regardless of which cluster is queried, the access point returns a fixed-number of packets, corresponding to the biggest cluster size. We use l to denote this number thereafter. For a cluster whose size is smaller than l, *dummy* packets are generated.

The basic idea of our approach is to only transmit packets from the queried cluster to the access point. However, to hide the target cluster, all the remaining clusters have to generate *fake* data transmission of the same pattern as that of the queried cluster. To better convey our idea, let us continue to use Figure 2 as an example. Suppose c_6 is the target queried cluster. The data transmission starts with ch_9, ch_8, ch_5 and ch_4, who are the tails of the respective routing pathes. Let us however only explain the transmission involving the target cluster c_6, as others follow the same manner. The cluster head ch_9 generates and sends l dummy packets $\mathcal{E}^l_{ch_9}(dummy\_data)$ to ch_6. We use $\mathcal{E}^l_{c_i}(data)$ to denote l packets of encryption of *data* by ch_i. ch_6 knows that its cluster is being queried, so he discards the data from ch_9, and generates and passes $\mathcal{E}^l_{ch_6}(sensed\_data)$ to ch_3. At this point of time, ch_3 may have already received $\mathcal{E}^l_{ch_5}(dummy\_data)$ and $\mathcal{E}^l_{ch_4}(dummy\_data)$ from its dependent nodes ch_5 and ch_4, respectively. If not, ch_3 waits until it gets the data from all its dependent nodes. The reason why a node does not proceed until receives from all its dependent nodes is to avoid disclosure of information from timing patterns. Then ch_3 decrypts $\mathcal{E}^l_{ch_6}(sensed\_data)$ and re-encrypts *sensed_date* to yield $\mathcal{E}^l_{ch_3}(sensed\_data)$, it then passes the data to the access point ch_1. The access point ch_1 waits until it gets data from all other routing pathes, and then decrypts $\mathcal{E}^l_{ch_3}(sensed\_data)$ and re-encrypts *sensed_date* to yield $\mathcal{E}^l_{ch_1}(sensed\_data)$. Finally, ch_1 sends $\mathcal{E}^l_{ch_1}$ $(sensed\_data)$ to the querying user as the query result. In our approach, every cluster head sends out l packets. Due to the semantic security of encryption, re-encryptions of the same data are not distinguishable. Therefore, the adversary watching the network cannot tell if the l packets sent out by a cluster head originate from the cluster head itself or from its dependent nodes.

Algorithm Details. A complete description of the algorithm in pseudo codes is shown in Algorithm 1, where u denotes the querying user and *ap* denotes the access point.

To start, the user contacts the access point by sending a *hello* message, including a *nounce* that will be used in the ensuing attestation process (Step 1). The access point then informs all the other cluster heads to form routing pathes using the method described earlier (Step 2). Before sending a query, the user must have assurance of the trustfulness of the cluster heads. This is achieved by means of attestation (Step 3). Note that it is unnecessary for the user to check the status of all cluster heads, which is quite expensive; it suffices to adopt the strategy of "chained attestation" along the established routing pathes. In particular, referring to Figure 2, u only verifies the access point: ch_9 is verified by ch_6; ch_4, ch_5 and ch_6 are verified by ch_3; ch_8 is verified by ch_7 who is in return verified by ch_2; ch_2 and ch_3 are verified by the access point. Once attestation is successful, the user sends to the access point the query e_u, which is the encryption of the identifier c of the target cluster using the shared secret key (Step 4). The access point broadcasts the query to all other cluster heads (Step 5), each decrypting the query and knowing which cluster the user is querying (Step 7). Each cluster head then collects sensed data (encrypted using the cluster key) from the sensor nodes of its cluster (Step 8).

Before sending out l packets of data to its forwarding node (Step 29), a cluster head must wait until it receives packets from all its dependent nodes (Step 9). Afterwards, if the cluster itself is the target cluster (Step 10-13), the cluster head simply ignores the packets from its dependent nodes, and encrypts the sensed data from its cluster. Note that every set of l packets consists of *head* and *content*, where the *head* is used to inform cluster heads enroute the origin of the l packets while without decrypting the *content*. For a cluster that is not the target one (Step 14-27), the cluster head checks whether one of its dependent nodes

Algorithm 1. Achieving User Query Privacy.

```
1: $u \rightarrow ap$: $hello$
2: $ap \leftrightarrow \{ch_i\}_i$: establish routing pathes.
3: $u \leftrightarrow ap$: attestation
4: $u \rightarrow ap$: $e_u = \mathcal{E}_u(c)$ /*$c$ is the identifier of the target cluster*/
5: $ap \rightarrow \{ch_i\}_i$: $e_u$
6: for EACH $ch_i$ do
7:    $c = \mathcal{D}_{ch_i}(e_u)$
8:    $\{sensor\} \rightarrow ch_i$: $data_{c_i} = \{\mathcal{CE}_{c_i}(sensed\_data)\}$
9:    $\{dependent\_node\} \rightarrow ch_i$: $\{packets\}$ /*$ch_i$ waits until gets packets from all its
      dependent nodes*/
10:   if $ch_i \in c$ then
11:      /*if $ch_i$ is cluster head of the target cluster $c$ */
12:      $packets = \mathcal{CD}_{c_i}(data_{c_i})$
13:      $packets = head||content = \mathcal{E}_c(c)||\mathcal{E}^l_{ch_i}(packets)$
14:   else
15:      $foundqueriedcluster$ = FALSE
16:      for $packets$ from EACH $dependent\_node$ do
17:         $c' = \mathcal{D}_{ch_i}(head)$
18:         if $c' == c$ then
19:            $content' = \mathcal{D}_{ch_i}(content)$
20:            $packets = head||content = \mathcal{E}_{ch_i}(c)||\mathcal{E}^l_{ch_i}(content')$
21:            $foundqueriedcluster$ = TRUE
22:            break
23:         end if
24:      end for
25:      if $foundqueriedcluster$ == FALSE then
26:         $packets = head||content = \mathcal{E}_{ch_i}(c_i)||\mathcal{E}^l_{ch_i}(dummy\_date)$
27:      end if
28:   end if
29:   $ch_i \rightarrow forward\_node$: $packets$
30: end for
31: $ap \rightarrow u$: $packets$
```

sends in the data of the target cluster. If yes, the cluster head re-encrypts the data (Step 16-22); otherwise, the cluster head generates l dummy packets (Step 25-27). Eventually, the access point passes the l packets of the target cluster to the querying user (Step 31).

Improvement. In the above scheme, to answer a user query all the sensor nodes are asked to send data to their respective cluster heads. This may shorten the lifetime of the network because of excess energy consumption. To mitigate this problem, We can alternatively trade off data freshness for energy efficiency, especially when queries come in at a high rate. In particular, sensor nodes *periodically* provide the sensed data to their respective cluster heads, who cache the data. The cluster heads then handle user queries using the cached data rather than collecting realtime data from the sensor nodes.

Table 1. Comparison Results

	Adversary model	Sensor storage	Sensor computation	Sensor communication	User registration
Our scheme	*Global eavesdropper*	*Constant*	*Constant*	*Constant*	*No*
Scheme in [7]	*Local eavesdropper*	*Linear to the total number of users*	*Related to routing path*	*Related to routing path*	*yes*

Comparison. The only earlier work we are aware of studying user query privacy is [7]. The two-server approach in [7] uses a routing scheme for data transmission similar to onion routing [14]. The routing path is constructed dynamically among the sensor nodes for each query. We list the comparison results between our approach and that in [7] in Table 1. To be more specific, our scheme assumes a global eavesdropper, so achieving better privacy. A sensor node in our scheme only needs to store a secret cluster key, while each sensor in [7] is required to store secret data linear to the total number of users authorized to query the network. On computation and communication, each sensor node in our scheme needs to encrypt its sensed data and sends to its cluster head, or is required to relay data from other nodes in the same cluster; as the routing pathes must be short within a cluster, the computation and communication are thus constant. In contrast, computation and communication for a sensor node in [7] depend on the routing path from the target cluster to the querying user. Finally, in [7] every authorized user is required to register to the servers, which in turn store secret data for the user's virtual name on each sensor node; sensor nodes in our scheme on the contrary have nothing to do with user registration. To conclude, the comparison results show that our scheme outperforms [7] in all aspects.

3.2 Achieving Source Location Privacy

Source location privacy is concerned with *not letting the adversary know which sensor node sends data to a sink (the base station)*. Most existing approaches, e.g., [24,29,31], consider local eavesdroppers who can only trace transmission hop by hop. An exception is [28], which assumes a global eavesdropper. The commonly used methods for achieving source location privacy are *random walk*, *source simulation*, or a combination of the two. The random walk method generates a random routing path from the source node to the sink for each transmission, while the source simulation method generates a number of fake source nodes simulating the actual source sensor, so as to confuse the adversary.

It appears that the random walk method is not effective under the global adversary overseeing the entire network, since the adversary always knows from which node packets originated. Our heterogeneous network architecture facilitates a much more efficient implementation of source simulation. In particular, to simulate the actual source node sending a packet to its cluster head, every other cluster head randomly chooses a sensor node in its cluster to send a fake packet.

Afterwards, cluster heads send their packets to the sink. Intuitively, source location privacy can be considered as a special case of user query privacy in our architecture. In order to keep the paper compact, we omit the details of the scheme, but it should not be difficult to specify, given the earlier scheme for user query privacy.

Source simulation in our architecture is more efficient than that in [28], due to the shortened routing pathes and reduced control messages resulting from the use of high-end cluster heads.

4 Related Work

Partitioning a WSN into clusters were proposed by several authors for achieving scalability and better performance [2,3,11,20]. In these schemes, one or more sensor nodes within a cluster are chosen as cluster head (i.e., homogeneous clustered WSNs). On the other hand, considering the limited capabilities of sensor nodes, studies from both the research community and the industry sector have tried to enhance network performance by incorporating a number of more powerful nodes in WSNs (i.e., heterogeneous clustered WSNs). A detailed theoretical analysis on the effect of adding powerful nodes to WSNs was given in [36]. It is concluded that only a modest number of reliable, long-range backhaul links and line-powered nodes are required to have a significant effect, and if properly deployed, heterogeneity can triple the average delivery rate and a 5-fold increase in the lifetime of a large battery-powered sensor networks. Intel has an on-going experimental effort [19] to incorporate Intel XScale® based nodes into WSNs. The experiment indicated that data traversing across a network are routed biased towards the XScale® nodes over simple sensor nodes, thereby indeed enhancing the overall system performance. [30,36] are among the work to study security issues in the homogeneous or heterogeneous clustered WSNs.

Our study of TC-enabled WSNs is motivated by the above idea of network clustering and heterogeneity, and TCG's trusted computing technology. The heterogeneous architecture we proposed differs from the existing heterogeneous networks such as [4,19,30,36] in that the high-end cluster heads in our WSNs are TC-enabled. This makes our WSNs not only have improved network performance, but also greatly facilitate security enforcement, as we have demonstrated earlier.

We realized that we are not the first to propose the idea of equipping some more powerful cluster heads in a WSN with trusted computing technology. [13,21] also discussed similar ideas. More specifically, [21] proposed lightweight attestation techniques that can help regular sensor nodes to check the status of a TC-enabled cluster head's platform, and [13] studied key establishment and management between sensor nodes and the base station, with the help of TC-enabled cluster heads. There seems no essential difference in the network architecture between our proposal and that of [13,21]. However, the techniques proposed in [13,21] are intended for the sensor nodes to perform attestation so as to check the platform state of the cluster heads. In contrast, in our architecture the sensor nodes never attempt to check the trustfulness of the cluster heads, and they simply trust their respective cluster heads. Our justification for this is that it is the responsibility of the end users of the network to concern about the trustfulness of the cluster heads. Other difference between our work and [13,21] is that different security issues in WSNs were addressed.

In this work, we focused on privacy protection in WSNs, and provided better solutions to two important privacy issues than existing proposals [7,24,28,29,31]. Another privacy issue in WSNs discussed in the literature is temporal privacy [23], which is concerned with the fact that an adversary, by simply monitoring the arrival of packets at the sink, can infer the temporal patterns of interested events. While we did not address the temporal privacy issue for lack of space, we believe our TC-enabled heterogeneous architecture is in a better position to solve the problem, because the *trusted* cluster heads should be able to perform timing synchronization more reliably and easily.

5 Conclusion and Future Work

Due to stringent resource limitations of sensor nodes, security enforcement is extremely challenging in wireless sensor networks. To solve this problem, we proposed to render a wireless sensor network heterogeneous, by incorporating TC-equipped high-end devices into clusters of the network, acting as cluster heads. We demonstrated how the TC-enabled cluster heads can effectively address privacy issues in WSNs.

This study is still in the preliminary stage. We are preparing to implement proof-of-the-concept TC-enabled WSN architecture, and further experiment with the architecture in certain real world wireless sensor network settings.

Acknowledgement

This work is partially funded by the A*STAR project SEDS-0721330047, and also supported in part by the Office of Research, Singapore Management University.

References

1. Arnold, T., Doorn, L.V.: The IBM PCIXCC: A New Cryptographic Coprocessor for the IBM EServer. IBM Journal of Research and Development 48 (May 2004)
2. Banerjee, S., Khuller, S.: A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks. In: Proc. IEEE INFOCOM 2001 (2001)
3. Basagni, S.: Distributed Clustering Algorithm for Ad-Hoc Networks. In: Proc. International Symposium on Parallel Architectures, Algorithms, and Networks (1999)
4. Bohge, M., Trappe, W.: An Authentication Framework for Hierarchical Ad Hoc Sensor Networks. In: Proc. ACM workshop on Wireless security, WiSE 2003, pp. 79–87 (2003)

5. Chan, H., Perrig, A., Song, D.: Random Key Pre-distribution Schemes for Sensor Networks. In: Proc. IEEE Symposium on Security and Privacy, pp. 197–213 (2003)
6. Chan, H., Perrig, A., Song, D.: Secure Hierarchical In-Network Aggregation in Sensor Networks. In: Proc. ACM Conference on Computer and Communications Security, CCS 2006 (2006)
7. Carbunar, B., Yu, Y., Shi, L., Pearce, M., Vasudevan, V.: Query Privacy in Wireless Sensor Networks. In: Proc. 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2007, pp. 203–212 (2007)
8. Du, W., Deng, J., Han, Y.S., Varshney, P.K.: A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In: Proc. ACM Conference on Computer and Communication Security, CCS 2003, pp. 42–51 (2003)
9. Das, S., Perkins, C., Royer, E.: Performance Comparison of Two On-demand Routing Procotols for Ad Hoc Networks. In: Proc. of IEEE INFOCOM 2000, vol. 1, pp. 3–12. IEEE Press, Los Alamitos (2000)
10. Eschenauer, L., Gligor, V.D.: A Key-Management Scheme for Distributed Sensor Networks. In: Proc. ACM Conference on Computer and Communication Security, CCS 2002 (2002)
11. Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next Century Challenges: Scalable Coordination in Sensor Networks. In: Proc. ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM 1999 (1999)
12. Gupta, P., Kumar, P.: The Capacity of Wireless Networks. IEEE Transactions on Information Theory 46(2), 388–404 (2000)
13. Ganeriwal, S., Ravi, S., Raghunathan, A.: Trusted Platform Based Key Establishment and Management for Sensor Networks (Under Review)
14. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding Routing Information. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 137–150. Springer, Heidelberg (1996)
15. Hu, L., Evans, D.: Secure Aggregation for Wireless Networks. In: Proc. 2003 Symposium on Applications and the Internet Workshops, SAINT 2003, pp. 384–394 (2003)
16. Hu, Y., Johnson, D., Perrig, A.: SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. Ad Hoc Networks Journal 1(1), 175–192 (2003)
17. Hu, Y., Perrig, A., Johnson, D.: Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. Wireless Networks Journal 11(1) (2005)
18. Ibriq, J., Mahgoub, I.: A Hierarchical Key Establishment Scheme for Wireless Sensor Networks. In: Proc. 21st International Conference on Advanced Networking and Application, AINA 2007 (2007)
19. http://www.intel.com/research/exploratory/heterogeneous.htm
20. Jain, N., Agrawal, D.P.: Current Trends in Wireless Sensor Network Design. International Journal of Distributed Sensor Networks 1(1), 101–122 (2005)
21. Krauß, C., Stumpf, F., Eckert, C.: Detecting node compromise in hybrid wireless sensor networks using attestation techniques. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 203–217. Springer, Heidelberg (2007)
22. Karlof, C., Wagner, D.: Secure Routing in Wireless Sensor Networks: Attacks and Countermeasurements. In: Proc. 1st IEEE International Workshop on Sensor Network Protocols and Applications (2003)

23. Kamat, P., Xu, W., Trappe, W., Zhang, Y.: Temporal Privacy in Wireless Sensor Networks. In: Proc. 27th International Conference on Distributed Computing Systems, ICDCS 2007, pp. 23–30 (2007)
24. Kamat, P., Zhang, Y., Trappe, W., Ozturk, C.: Enhancing Source-Location Privacy in Sensor Network Routing. In: Proc. 25th IEEE International Conference on Distributed Computing Systems, ICDCS 2005, pp. 599–608 (2005)
25. Liu, D., Ning, P.: Location-based Pairwise Key Establishement for Relatively Static Sensor Networks. In: Proc. ACM Workshop on Security of Ad hoc and Sensor Networks (2003)
26. Liu, D., Ning, P., Sun, K.: Efficient Self-Healing Group Key Distribution with revocation Capability. In: Proc. ACM Conference on Computer and Communication Security, CCS 2003 (2003)
27. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In: Proc. 5th Annual Symposium on Operating Systems Design and Implementation, OSDI 2002 (2002)
28. Mehta, K., Liu, D., Wright, M.: Location Privacy in Sensor Networks Against a Global Eavesdropper. In: Proc. IEEE International Conference on Network Protocols, ICNP 2007, pp. 314–323 (2007)
29. Ouyang, Y., Le, Z., Chen, G., Ford, J., Makedon, F.: Entrapping Adversaries for Source Protection in Sensor Networks. In: Proc. International Symposium on World of Wireless, Mobile and Multimedia Network, WoWMoM 2006, pp. 23–34 (2006)
30. Oliveira, L.B., Wong, H.C., Loureiro, A.A.: LHA-SP: Secure Protocols for Hierarchical Wireless Sensor Networks. In: Proc. IFIP/IEEE International Symposium on Integrated Network Management, pp. 31–44 (2005)
31. Ozturk, C., Zhang, Y., Trappe, W.: Source-location Privacy in Energy-contained Sensor Network Routing. In: Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN 2004, pp. 88–93 (2004)
32. Patwardhan, A., Parker, J., Joshi, A., Iorga, M., Karygiannis, T.: Secure Routing and Intrusion Detection in Ad Hoc Networks. In: Proc. 3rd International Conference on Pervasive Computing and Communications. IEEE, Los Alamitos (2005)
33. Przydatek, B., Song, D., Perrig, A.: SIA: Secure Information Aggregation in Sensor Networks. In: Proc. ACM SenSys (2003)
34. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J.D.: SPINS: Security Protocols for Sensor Networks. Wireless Networks Journal (WINE) (September 2002)
35. Trusted Computing Group, `www.trustedcomputinggroup.org`
36. Yarvis, M., et al.: Exploiting Heterogeneity in Sensor Networks. In: Proc. IEEE INFOCOM 2005 (2005)
37. Zhu, S., Setia, S., Jajodia, S.: LEAP: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks. In: Proc. ACM Conferenc on Computer and Communication Security, CCS 2003, pp. 62–72 (2003)

Trusted Privacy Domains – Challenges for Trusted Computing in Privacy-Protecting Information Sharing

Hans Löhr[1], Ahmad-Reza Sadeghi[1], Claire Vishik[2], and Marcel Winandy[1]

[1] Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
{hans.loehr,ahmad.sadeghi,marcel.winandy}@trust.rub.de
[2] Intel Corporation
claire.vishik@intel.com

Abstract. With the growing use of the Internet, users need to reveal an increasing amount of private information when accessing online services, and, with growing integration, this information is shared among services. Although progress was achieved in acknowledging the need to design privacy-friendly systems and protocols, there are still no satisfactory technical privacy-protecting solutions that reliably enforce user-defined flexible privacy policies. Today, the users can assess and analyze privacy policies of data controllers, but they cannot control access to and usage of their private data beyond their own computing environment.
In this paper, we propose a conceptual framework for user-controlled formal privacy policies and examine elements of its design and implementation. In our vision, a Trusted Personal Information Wallet manages private data according to a user-defined privacy policies. We build on Trusted Virtual Domains (TVDs), leveraging trusted computing and virtualization to construct privacy domains for enforcing the user's policy. We present protocols for establishing these domains, and describe the implementation of the building blocks of our framework. Additionally, a simple privacy policy for trusted privacy domains functioning between different organizations and entities across networks is described as an example. Finally, we identify future research challenges in this area.

1 Introduction

Global connectivity and easy access to distributed applications and digital services over the Internet changed the paradigm of both business and consumer use of information. The Internet offers new opportunities to individuals, e.g., e-commerce and social network services. In addition to personal computers, mobile devices, such as smart phones, allow users to access numerous services through mobile networks from any location.

Together with the new opportunities, new security threats also developed, rapidly growing in number and sophistication. Some security threats, such as identity theft, one of the fastest growing crimes on the Internet, also can cause privacy violations [1,2]. But privacy issues are much broader: individuals frequently generate and reveal a significant amount of personal and sensitive information when they use a service such as online shopping or social networking. Even if a transaction is not personalized, it always leaves a trail that can be aggregated with other information and analyzed, potentially leading to privacy leaks. Also, as devices access networks and services, information about these accesses can be recorded.[1] The users have to trust the application provider to treat their personal data in an appropriate manner, e.g., according to best practices and regulatory requirements reflected in privacy policies. The users can read statements about privacy policies on websites, but the policies do not allow for flexibility in disclosing data necessary to access the service. There are few[2] technical means to support this kind of enforcement. Ideally, the users should be able to grant access to their sensitive information only when the systems are trustworthy and should be allowed to revoke this permission.

Technical measures in the areas of modern IT security and cryptography provide only partial solutions. Because of the inherent vulnerabilities resulting from high complexity of systems, common computing platforms require careful and attentive system administration skills, and complete protections against execution of malicious code and tampering is impossible.

In this paper, we propose a conceptual framework for user-controlled privacy policies and examine first elements of its design and implementation. The goal is to improve the current status of data and privacy protection by supporting legal measures with novel technical solutions based on Trusted Computing (TC) as described below:

– We outline a general approach to creating privacy domains, in which a guardian agent (Trusted Personal Information Wallet) manages private data according to a user-defined privacy policy (Section 2). The agent can migrate to other platforms, but only in approved trusted domains.
– We describe a simple policy that requires trusted privacy domains between different organizations and entities. We build on the idea of Trusted Virtual Domains (TVDs), leveraging trusted computing and virtualization to automatically construct privacy domains for enforcing the user's policy. We describe protocols for establishing these domains and the implementation of the building blocks of the framework (Section 3).
– Finally, we address future research challenges, analyzing currently available policy languages that cannot yet support full solutions for the reliable enforcement of user controlled privacy (Section 4).

[1] Revealing private information is sometimes necessary or unavoidable outside of the Internet (e.g., in supermarkets, due to surveillance, etc.). Although we do not study these methods to gain information about individuals, we note that the revealed information inside and outside the Internet can potentially be linked.

[2] Auditing and certification are examples for at least some technology-related methods, e.g., product evaluation according to Common Criteria or certification according to ISO 27001/27002 for information security management systems in enterprises.

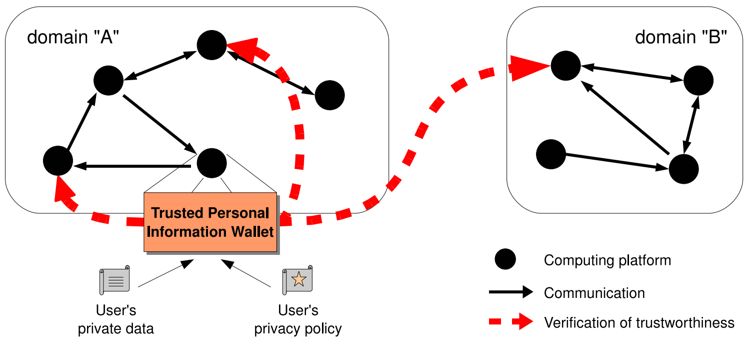


Fig. 1. Basic idea of the overall architecture

2 Framework for Privacy Domains

We propose to support the enforcement of privacy policies by establishing trusted domains. These policies enable the user (individual or organization) to specify fine-grained instructions for the use of private information. As the level of online activities increases and entities or organizations with complex rules interoperate, the policies may become very complex and benefit from automatic enforcement.

The proposed architecture provides mechanisms to protect sensitive and private information across IT domains and systems. The deployment of Trusted Computing technologies for privacy protection can help achieve this goal. To ensure that private information is not re-distributed to unauthorized parties, it needs to be technically bound to only those receivers that are known to comply with the policies. Communication endpoints need to attest reliably to their compliance to specified policies.

To enforce policies, we propose a "guardian agent" for the user: a *Trusted Personal Information Wallet* that is transferable between platforms and performs "verification" of the trustworthiness of a remote IT system, i.e., compliance to a specified policy. The verification helps guarantee the enforcement of the user's privacy policy when sensitive information is transmitted. Figure 1 shows an abstract illustration of the proposed concept.

In order to achieve technical enforcement of the security and privacy policies, we develop a security architecture that allows the user to share sensitive information between computing platforms while ensuring the participating platforms have technical means to comply with the policies.

Figure 2 shows a high-level view of the process of policy enforcement. A privacy policy in a machine-readable format is incorporated into the wallet. (step 1). The wallet interprets the policy and configures security and privacy services of the underlying computing platform (step 2). The security services enforce the policy by controlling communication between applications in different domains (step 3). To reliably enforce the policy, trusted security & privacy services have to run on all participating platforms, e.g., based on a security-enhanced hypervisor [3], which allows the system owners to use legacy applications and operating

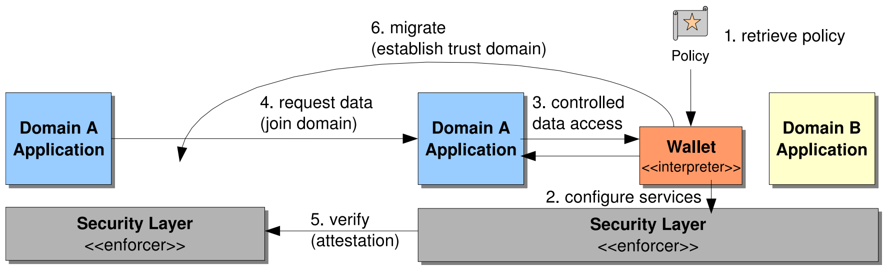


Fig. 2. Envisioned architecture for policy enforcement

systems in virtual machines, eliminating the need for new client and server side applications.

For data transmission, we propose new protocols based on existing attestation schemes of TC technology. When a user or application agent of another platform requests to access sensitive information (step 4), the security services of the source platform first verify the trustworthiness of the target platform using attestation mechanisms (step 5) to ensure the destination provides the required security mechanisms to enforce the policy. After successful verification, the wallet migrates to the destination platform (step 6) in order to act as policy decision module and to configure the security services of the target to enforce the defined policy. Service providers do not need to implement additional functionality on their server side (except for the underlying security layer) to interpret the policy or a clearinghouse for the policy interpretation. The wallet will interpret the policy and use the underlying security services of each platform to enforce it.

3 Experience with Trusted Virtual Domains

As a first step towards realizing privacy domains and policy enforcement as described before, we employ the concept of Trusted Virtual Domains (TVDs) [4,5]. In this section, we briefly review this concept and describe its novel application as privacy policy enforcement as well as our implementation of TVDs.

3.1 Concept of TVDs

A Trusted Virtual Domain (TVD) is a coalition of virtual and/or physical machines that can trust each other based on a security policy that is uniformly enforced independently of the boundaries of physical computing resources. It leverages the combination of TC and virtualization techniques in order to provide confinement boundaries for an isolated execution environment — a domain — hosted by several physical platforms.

A TVD-enforcing system supports the creation of virtual networks on physical or virtual systems. Members of a TVD can “see” and access other TVD members, but it is closed to non-members. Different instances of several TVDs can

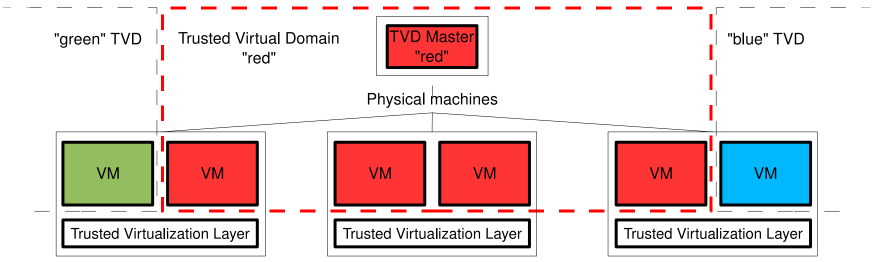

"green" TVD
Trusted Virtual Domain "red"
TVD Master "red"
"blue" TVD
Physical machines
VM
VM
VM
VM
VM
VM
Trusted Virtualization Layer
Trusted Virtualization Layer
Trusted Virtualization Layer

Fig. 3. Conceptual view of trusted virtual domains (TVDs)

execute on the same physical platform because the underlying virtual machine monitor isolates virtual machines of different TVDs in separate compartments and isolated virtual networks.

Figure 3 shows an example of three TVDs (identified by colors) distributed over different physical machines. The decision whether a virtual or real machine is allowed to join the TVD is enforced based on a *TVD policy*. A special node in the TVD (*TVD Master*), e.g., implemented as a central server, controls the access to the TVD by following the admission control rules specified in the TVD policy. These rules include integrity measurements of the platforms and virtual machines that are allowed to join the domain. TC technology is used to establish trust in the reported measurements, e.g., following the Trusted Computing Group (TCG) approach, hash values of the software boot stack (BIOS, bootloader, virtualization layer as well as loaded virtual machines) are stored in and signed by a Trusted Platform Module (TPM) [6] and reported to the TVD Master during attestation. The TVD Master can reliably verify whether the reported values comply with the TVD policy and whether it can rely on the enforcement mechanisms of the local platforms.[3]

TVDs were first proposed by Griffin et al. [4] and Bussani et al. [5]. Recent research describes secure network virtualization [7], and discusses the management of TVDs in data centers [8]. The OpenTC project[4] has addressed some areas of implementing TVDs in the context of enterprise rights management and managing virtual data centers. A major issue is how the domain can be managed securely: individual machines must be able to join a domain only if they fulfill the requirements for joining, and the procedures for a platform to leave a domain must be securely constructed. These aspects of TVDs have not been studied in details yet. We describe the TVD establishment and join protocols and how TC functionality is used (see Section 3.3). The idea of applying the TVD concept to secure information sharing has been addressed by Katsuno et al. [9]. We extend this idea to privacy policy enforcement.

[3] The definition of the required integrity measurement values in the TVD policy presupposes the knowledge about the security properties of the corresponding software. In practice, trust can be achieved via independent trusted third parties that evaluate and certify IT products according to standards like Common Criteria.

[4] See `http://www.opentc.net`

3.2 Realizing a Simple Privacy Policy with TVD

Let us consider a very simple privacy policy: only members of a particular TVD have access to the private information. The TVD policy expresses the requirements for virtual machines to join the TVD and to access this information. The TVD policy is used to implement the privacy policy, and the TVD infrastructure provides the policy enforcement for the wallet.

The wallet can act as TVD Master. In this case, it is directly responsible for policy enforcement. All parties that want to access the information have to join the TVD first. As they request to join, the wallet verifies the security properties of the joining parties using attestation. If the verification succeeds, the joining party becomes a member of the TVD and can then access sensitive information. The wallet can specify a set of "good" values for the platform configuration that are necessary to access the data.

Application scenarios for the case where the wallet is the TVD Master include those where the private information of one user is distributed to "homogeneous" data consumers, e.g., in an e-health scenario, the medical data and health records of patients are only accessible to computing platforms of medical personnel, but not to systems used by other departments.

In other classes of scenarios, where users belonging to a group want to exchange private data, it is unrealistic to have a virtual domain managed by a user's wallet. In these cases, a trusted party could provide a TVD Master responsible for policy enforcement for the group. The wallet of a user who wishes to exchange information within a group could attest the responsible TVD Master (e.g., using TCG attestation) before joining. If this attestation includes both the platform configuration of the TVD Master and the TVD policy, the wallet can ensure that information is only distributed within a TVD, where the master enforces a TVD policy that complies to the user's own privacy policy. The wallet can migrate to any node in the TVD (using conventional VM migration), and the required verification of the security properties of the destination is handled by the TVD establishment.

3.3 Implementation

Our prototype is based on the idea that a local proxy of the corresponding TVD Master, the *TVD Proxy*, is running on each physical platform that is supposed to execute virtual machines as part of a TVD. The TVD Proxy is responsible for the local enforcement of the TVD policy and performs the admission control for joining virtual machines. Since instances of multiple TVDs should be able to run isolated on one computing platform, there can be several TVD Proxies (one for each corresponding TVD) on one platform.

The main components of the trusted virtualization layer are as follows (see also Figure 4):

– *TVD-Proxy-Factory*: service that creates and manages TVD Proxies. During the establishment of the TVD, the TVD Master deploys the policy P and

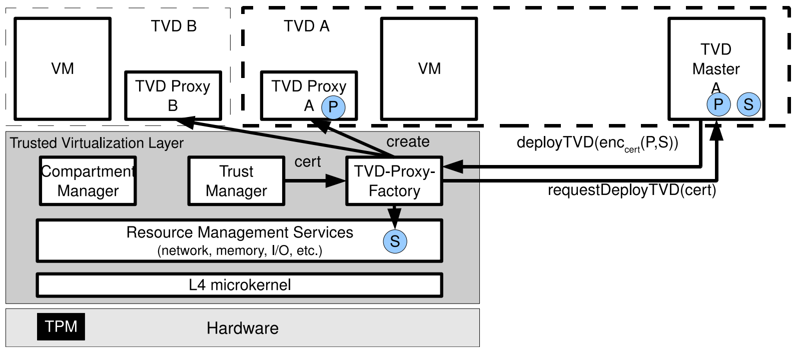


Fig. 4. TVD implementation architecture

corresponding credentials S (cryptographic keys and certificates for, e.g., network encryption) to the TVD-Proxy-Factory. To "verify" the trustworthiness of the platform and its virtualization layer, the TVD Master requests a remote attestation of the integrity measurements, using trusted computing functionality of a TPM [6].

– *CompartmentManager*: service responsible for starting and terminating virtual machines (compartments) and taking integrity measurements of the virtual machines on start-up. This service also defines access rights for communication between active compartments.
– *TrustManager*: service providing an interface to the underlying TPM and used to create new binding keys, generate certificates for these keys, and unbind data encrypted with a binding key. The binding key is protected by the TPM and bound to the integrity measurements of the underlying platform and its trusted virtualization layer. The certificate includes these integrity measurements and permits a remote party to establish a *trusted channel* to the platform, i.e., a secure channel (providing confidentiality and integrity) bound to the integrity of the endpoint(s).

We have implemented this design based on an existing security kernel, Turaya[5], which comprises two layers: a *hypervisor layer* based on an L4 microkernel and resource management services (memory management, I/O drivers), and a *trusted software layer* providing security services, e.g., secure storage, virtualized network, compartment management, and trusted channel establishment.

The L4 microkernel ensures isolation of processes and controls inter-process communication (IPC). Compartments can be native L4 tasks or para-virtualized Linux instances (L4Linux). Communication between compartments can be allowed or denied by applying access rights to their IPC interfaces. The microkernel enforces the IPC access control.

To support wallet functionality, it is necessary to establish a TVD and attach a virtual machine to the TVD. A TVD is established in two phases:

[5] `http://www.emscb.com/content/pages/turaya.htm`

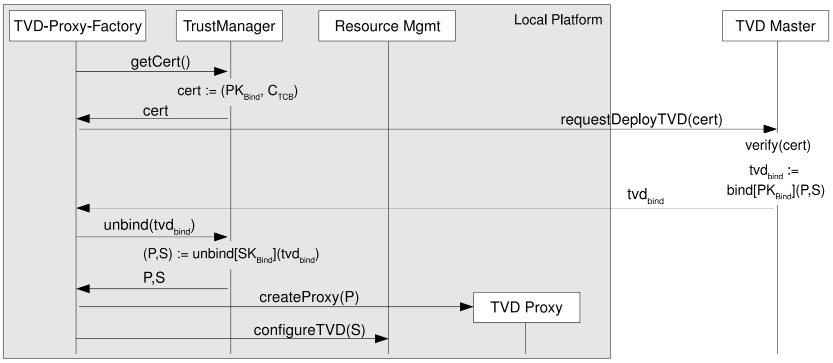


Fig. 5. TVD deployment protocol

1. *Deploy TVD*: First, the local TVD infrastructure must be set up, including the deployment of the TVD policy and TVD credentials from the TVD Master to the trusted virtualization layer of the local platform.
2. *Join TVD*: When policy and credentials are deployed, the local TVD Proxy enforces the policy and determines if local VMs are allowed to join the TVD.

Staged establishment of the TVD was selected to avoid a central admission control that would result in considerable performance trade-offs. In this approach, the TVD policy enforcement is partially delegated to the local platforms, but the TVD Master must verify the trustworthiness (integrity state) of the platforms to establish if they can be trusted. This is done during the deployment phase.

Deploy TVD. When TVD-Proxy-Factory receives a request to deploy a TVD, TrustManager generates a binding certificate $cert := (PK_{Bind}, C_{TCB})$. The TrustManager uses the TPM to generate a new binding key pair (SK_{Bind}, PK_{Bind}), where the secret key part is protected by the TPM and bound to the integrity measurement of the trusted virtualization layer (C_{TCB}). The TVD-Proxy-Factory requests deployment from the TVD Master of the desired TVD and sends the binding certificate, including the binding key PK_{Bind}.

The TVD Master checks whether the integrity measurement of the platform matches the TVD policy. If it does, the TVD Master encrypts the TVD policy P and the corresponding TVD credentials S with the binding key PK_{Bind}, and sends the encrypted data to the local TVD-Proxy-Factory. See Figure 5.

The TVD-Proxy-Factory requests the TrustManager to unbind the data and retrieves the TVD policy and credentials (P, S). It creates a new TVD Proxy, passes the TVD policy P to it and configures the underlying resource management services (e.g., virtual network switch) with the credentials S. Now the TVD infrastructure is set up locally and ready to join virtual machines.

Join TVD. The user creates the VM using the CompartmentManager. The CompartmentManager measures the integrity of the VM image (i.e., hashing the image file), stores the measurement for future requests (during runtime),

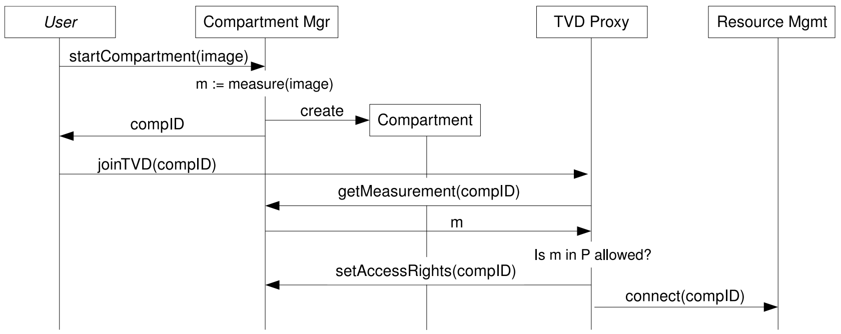


Fig. 6. TVD join protocol

starts the VM in a compartment, and returns a compartment identifier (unique during runtime of the platform). The user can request to join the compartment to the TVD by passing the compartment ID to the TVD Proxy.

The TVD Proxy obtains the integrity measurement m of the given compartment ID from the CompartmentManager. If the value m is listed in the TVD policy P as allowed to join, the TVD Proxy configures the underlying resource management to connect the compartment to the virtual resources of the TVD, e.g., “plugging” a virtual network connector to the VM.[6]

4 Remaining Challenges and Related Work

Privacy policy languages are designed to translate the privacy policies for users and organizations into statements that can be interpreted by IT systems. In [10] the authors give an overview of common policy languages. W3C’s Platform for Privacy Preferences (P3P) was designed to express website privacy policies in machine-readable format [11], and P3P *Preference Exchange Language* (APPEL) is used to express privacy preferences of an individual and to query the P3P data[12,13]. *CPExchange* was developed to facilitate business-to-business communication about privacy policies [14]. For internal privacy policies of organizations, IBM proposed *Enterprise Privacy Authorization Language* (EPAL) [15]. Another language for describing both privacy and security policies in a machine readable format is the *eXtensible Access Control Markup Language* (XACML) [16]. Other initiatives, such as DPAL [17], and XPref [18], addressed various aspects of expressing privacy requirements and related concepts. Due to the growth of services that require the transfer of context sensitive information (e.g., time and location), the Internet Engineering Task Force (IETF) initiative started work on *Geopriv*, a language that can express policies for granting access on the basis of presence and location information [17].

[6] The details of the resource isolation and realization of TVDs on this level are out of scope for this paper. Cabuk et al. [7] show how to realize network isolation based on VLAN tagging.

In addition to the earlier work on access control policies and (privacy) languages, recent research has analyzed and developed methodologies for evaluating actual policies to compare them with the policies the users desired to use, e.g., Bauer et al. [19] conducted user study of access control policies. Cornwell et al. [20] have analyzed policy management in different applications in mobile computing and developed applications where users can define policies to control the usage of private information, e.g., location-based or contextual information. Sadeh et al. [21] analyzed user interfaces for policy definition and mechanisms for auditing the disclosure of private information.

We conclude that, while the need to ensure user control and enforcement of privacy policies was recognized, most research so far focuses on formal languages defining privacy and related policies in various contexts, user requirements for such policies, and approaches for applications to incorporate user controlled flexible policies. However, little attention was given to the mechanisms to support automatic enforcement and interpretation of these policies. In this paper, we propose an approach to policy enforcement that takes into consideration the results of earlier research, including user requirements and design of formal policy languages. The new framework offers a realistic approach to the control and enforcement of privacy policies in a variety of contexts. We think that TVDs can help construct the *privacy domains* to support privacy protection of sensitive data that need to be shared. The process to build domains where the protection of sensitive data is governed by privacy policies determined by users still needs to be defined. Policy management for privacy domains remains a major challenge as complex privacy policies need to be enforced within a domain, when a machine joins or leaves the domain, and for inter-domain communication.

The idea of the Trusted Personal Information Wallet is derived from previous work [22], which uses a password wallet as authentication agent to access web sites. It protects private data (credentials) of a user during the authentication to a remote server. This approach uses Trusted Computing technology to ensure that the wallet is executed in a trusted environment. In addition to protecting the credentials, *SpyBlock* [23] protects against the unintentional disclosure of sensitive information (like credit card numbers, name, address, etc.) as a result of malicious transactions [24].

Since the Trusted Personal Information Wallet acts as an agent for the user's private data and it can migrate to other platforms, it is comparable to mobile agents. Wilhelm et al. [25] propose to use a tamper-resistant hardware to provide a secure execution environment for mobile agent code. Balfe and Gallery [26] outline how attestation can be used to ensure that an agent only visits host platforms behaving in an expected manner and that access to the private agent data complies to the desired security policies. In [27], the main approach is the protection of an agent's private cryptographic key by binding the key to a TPM. In contrast, the wallet (agent) in the framework proposed here does not directly use the TPM, but relies on the TVD infrastructure to (automatically) deploy a trusted execution environment and enforce privacy policies.

5 Conclusion

In this paper, we proposed a conceptual framework for privacy policy management and enforcement to ensure security and trust for sharing of private or sensitive information. We believe that Trusted Computing technology, in particular the concept of trusted virtual domains (TVDs), can efficiently support privacy policy enforcement. We think that future research will lead to the development of trusted privacy-enhancing architectures that will be applicable to several use cases, e.g., e-commerce, enterprise rights management, e-health, and other areas. Here we outline only the first steps towards the definition of such architectures. In addition, the definition and enforcement of more complex privacy policies will be a subject of future work.

Acknowledgment. This work has been partially funded by the European Commission as part of the OpenTC project (http://www.opentc.net).

References

1. Anti Phishing Working Group: Phishing Activity Trends Report(s) (2005-2007), http://www.antiphishing.org
2. Evers, J.: Phishers get personal (May 2005), http://news.com.com/Phishers+get+personal/2100-7349_3-5720672.html
3. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: Secure hypervisor approach to trusted virtualized systems. Technical Report RC23511, IBM Research Division (February 2005)
4. Griffin, J.L., Jaeger, T., Perez, R., Sailer, R., van Doorn, L., Cáceres, R.: Trusted Virtual Domains: Toward secure distributed services. In: Proceedings of the 1st IEEE Workshop on Hot Topics in System Dependability (HotDep 2005) (June 2005)
5. Bussani, A., Griffin, J.L., Jansen, B., Julisch, K., Karjoth, G., Maruyama, H., Nakamura, M., Perez, R., Schunter, M., Tanner, A., Doorn, L.V., Herreweghen, E.A.V., Waidner, M., Yoshihama, S.: Trusted Virtual Domains: Secure foundations for business and IT services. Technical Report RC23792, IBM Research (2005)
6. Trusted Computing Group: TPM main specification, version 1.2 rev. 103 (July 2007), https://www.trustedcomputinggroup.org
7. Cabuk, S., Dalton, C.I., Ramasamy, H., Schunter, M.: Towards automated provisioning of secure virtualized networks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007), pp. 235–245. ACM Press, New York (2007)
8. Berger, S., Cáceres, R., Pendarakis, D., Sailer, R., Valdez, E., Perez, R., Schildhauer, W., Srinivasan, D.: TVDc: Managing security in the trusted virtual datacenter. SIGOPS Oper. Syst. Rev. 42(1), 40–47 (2008)
9. Katsuno, Y., Kudo, M., Perez, P., Sailer, R.: Towards Multi-Layer Trusted Virtual Domains. In: The 2nd Workshop on Advances in Trusted Computing (WATC 2006 Fall), Tokyo, Japan, Japanese Ministry of Economy, Trade and Industry (METI) (November 2006)
10. Kumaraguru, P., Cranor, L., Lobo, J., Calo, S.: A survey of privacy policy languages. In: Workshop on Usable IT Security Management (USM 2007): Proceedings of the 3rd Symposium on Usable Privacy and Security. ACM, New York (2007)

11. Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., Reagle, J.: The Platform for Privacy Preferences 1.0 (P3P 1.0) specification. Technical report (April 2002)
12. Cranor, L.: Web Privacy with P3P. O'Reilly & Associates, Sebastopol (2002)
13. Cranor, L., Langheinrich, M., Marchiori, M.: A P3P Preference Exchange Language 1.0 (APPEL 1.0). Technical report, WWW Consortium (June 2005)
14. Bohrer, K., Holland, B.: Customer Profile Exchange (CPExchange) Specification, Version 1.0. Technical report (October 2000)
15. Schunter, M., Ashley, P., Hada, S., Karjoth, G., Powers, C.: Enterprise Privacy Authorization Language (EPAL 1.1). Technical report, IBM (2003)
16. Moses, T.: eXtensible Access Control Markup Language (XACML) version 2.0. Technical report, Oasis (2005)
17. Schulzrinne, H., Tschofenig, H., Morris, J., Cuellar, J., Polk, J., Rosenberg, J.: A document format for expressing privacy preferences (August 2006), http://tools.ietf.org/html/draft-ietf-geopriv-common-policy-11
18. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: An XPath-based preference language for P3P. In: WWW 2003: The 12th International Conference on World Wide Web, pp. 629–639 (2003)
19. Bauer, L., Cranor, L.F., Reeder, R.W., Reiter, M.K., Vaniea, K.: A user study of policy creation in a flexible access-control system. In: SIGCHI Conference on Human Factors in Computing Systems (CHI 2008). ACM, New York (2008)
20. Cornwell, J., Fette, I., Hsieh, G., Prabaker, M., Rao, J., Tang, K., Vaniea, K., Bauer, L., Cranor, L., Hong, J., McLaren, B., Reiter, M., Sadeh, N.: User-controllable security and privacy for pervasive computing. In: 8th IEEE Workshop on Mobile Computing Systems and Applications (HotMobile 2007). IEEE, Los Alamitos (2007)
21. Sadeh, N., Hong, J., Cranor, L., Fette, I., Kelley, P., Prabaker, M., Rao, J.: Understanding and capturing people's privacy policies in a mobile social networking application. Journal of Personal and Ubiquitous Computing (2008)
22. Gajek, S., Sadeghi, A.R., Stüble, C., Winandy, M.: Compartmented security for browsers – or how to thwart a phisher with trusted computing. In: 2nd Intl. Conference on Availability, Reliability and Security (ARES 2007), pp. 120–127 (2007)
23. Jackson, C., Boneh, D., Mitchell, J.: Spyware resistant web authentication using virtual machines (2006), http://crypto.stanford.edu/spyblock/
24. Jackson, C., Boneh, D., Mitchell, J.: Transaction generators: Root kits for web. In: 2nd USENIX Workshop on Hot Topics in Security (HotSec 2007) (2007)
25. Wilhelm, U.G., Staamann, S.M., Buttyan, L.: A pessimistic approach to trust in mobile agent platforms. IEEE Internet Computing 4(05), 40–48 (2000)
26. Balfe, S., Gallery, E.: Mobile Agents and the Deus Ex Machina: Protecting Agents using Trusted Computing. In: Proceedings of the 2007 IEEE International Symposium on Ubisafe Computing (UbiSafe 2007). IEEE Computer Society Press, Los Alamitos (2007)
27. Xian, H., Feng, D.: Protecting mobile agents' data using trusted computing technology. Journal of Communication and Computer 4(3), 44–51 (2007)

Author Index