Kyo-Il Chung
Kiwook Sohn
Moti Yung (Eds.)

LNCS 5379

Information Security Applications

9th International Workshop, WISA 2008
Jeju Island, Korea, September 2008
Revised Selected Papers

Springer

Kyo-Il Chung Kiwook Sohn Moti Yung (Eds.)

Information Security Applications

9th International Workshop, WISA 2008
Jeju Island, Korea, September 23-25, 2008
Revised Selected Papers

Springer

Volume Editors

Kyo-Il Chung
Electronics and Telecommunications Research Institute (ETRI)
Information Security Research Division
161 Gajeong Dong, YuseongGu
Daejeon, 305-700, Korea
E-mail: kyoil@etri.re.kr

Kiwook Sohn
The Attached Institute of Electronics
and Telecommunications Research Institute (ETRI)
1 Yuseong-Post
Daejeon, 305-702, Korea
E-mail: kiwook@ensec.re.kr

Moti Yung
Google Inc.
Columbia University, Computer Science Department
1214 Amsterdam Avenue
New York, NY 10027, USA
E-mail: moti@cs.columbia.edu

Library of Congress Control Number: 2009920695

CR Subject Classification (1998): E.3, D.4.6, F.2.1, C.2, J.1, C.3, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-642-00305-2 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-00305-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12612604 06/3180 5 4 3 2 1 0

Preface

The 9th International Workshop on Information Security Applications (WISA 2008) was held in Jeju Island, Korea during September 23–25, 2008. The workshop was sponsored by the Korea Institute of Information Security and Cryptology (KIISC), the Electronics and Telecommunications Research Institute (ETRI) and the Ministry of Knowledge Economy (MKE).

WISA aims at providing a forum for professionals from academia and industry to present their work and to exchange ideas. The workshop covers all technical aspects of security applications, including cryptographic and non-cryptographic techniques.

We were very pleased and honored to have served as the Program Committee Co-chairs of WISA 2008. The Program Committee received 161 papers from 18 countries, and accepted 24 papers for the full presentation track. The papers were selected after an extensive and careful refereeing process in which each paper was reviewed by at least three members of the Program Committee.

In addition to the contributed papers, the workshop had three special talks. Anat Zeelim-Hovav gave an invited talk, entitled "Investing in Information Security: A Coin in a Wishing Well?" Tai-Myoung Chung and Dieter Gollmann gave invited talks entitled "Today and Tomorrow of Security and Privacy" and "SOA Security: Service-Oriented Access Control," repectively.

Many people deserve our gratitude for their generous contributions to the success of the workshop. We would like to thank all the people involved in the technical program and in organizing the workshop. We are very grateful to the Program Committee members and the external referees for their time and efforts in reviewing the submissions and selecting the accepted papers. We also express our special thanks to the Organizing Committee members for their hard work in organizing the workshop.

Finally, on behalf of all those involved in organizing the workshop, we would like to thank all the authors who submitted papers and the invited speakers. Without their submissions and support, WISA could not have been a success.

November 2008

Kyo-Il Chung
Kiwook Sohn
Moti Yung

Organization

Advisory Committee

Mun Kee Choi	ETRI, Korea
Hideki Imai	Tokyo University, Japan
Dae-Ho Kim	ETRI, Korea
Sehun Kim	KAIST, Korea
Pil-Joong Lee	POSTECH, Korea
Sang-Jae Moon	Kyungpook National University, Korea
Kil-Hyun Nam	Korea National Defense University, Korea
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Man-Young Rhee	Kyung Hee University, Korea
Min-Sub Rhee	Dankook University, Korea
Joo-Seok Song	Yonsei University, Korea
Dong-Ho Won	Sungkyunkwan University, Korea

General Co-chairs

Chaegyu Kim	ETRI, Korea
Hong-Sub Lee	KIISC, Korea

Steering Committee

Hyun-Sook Cho	ETRI, Korea
Sang-Choon Kim	Kangwon National University, Korea
Hyung-Woo Lee	Hanshin University, Korea
Jae-Kwang Lee	Hannam University, Korea
Dong-Il Seo	ETRI, Korea
OkYeon Yi	Kookmin University, Korea

Organization Committee

Chair	Jae-Cheol Ryou	Chungnam National University, Korea
Finance	Seong-Gon Choi	Chungbuk National University, Korea
	Jintae Oh	ETRI, Korea
Publication	Jaehwoon Lee	Dongguk University, Korea
Publicity	Dong Gue Park	Soonchunhyang University, Korea
	Neungsoo Park	Konkuk University, Korea
Registration	Kilsoo Chun	KISA, Korea
	Dohoon Lee	ETRI, Korea
Local Arrangements	Khi Jung Ahn	Cheju National University, Korea
	Taenam Cho	Woosuk University, Korea

Program Committee

Co-chairs	Kyo-Il Chung	ETRI, Korea
	Kiwook Sohn	ETRI, Korea
	Moti Yung	Columbia University, USA
Members	C. Pandu Rangan	IIT Madras, India
	Hyoung-Kee Choi	Sungkyunkwan University, Korea
	Hyun Cheol Chung	BCNE Global.Co., Ltd., Korea
	Debbie Cook	Columbia University, USA
	Pierre Alain Fouque	ENS, France
	JaeCheol Ha	Hoseo University, Korea
	Hoh Peter In	Korea University, Korea
	Stefan Katzenbeisser	Philips Research, The Netherlands
	Howon Kim	Pusan National University, Korea
	Hyong Shik Kim	Chungnam National University, Korea
	Hyung Jong Kim	Seoul Women University, Korea
	Seok Woo Kim	Hansei University, Korea
	Seung Joo Kim	Sungkyunkwan University, Korea
	Yongdae Kim	University of Minnesota, USA
	Brian King	Indiana University - Purdue University Indianapolis, USA
	Chiu Yuen Koo	Google Inc., USA
	Hong Seung Ko	Kyoto College of Graduate Studies for Informatics, Japan
	Jin Kwak	Soonchunhyang University, Korea
	Deok Gyu Lee	ETRI, Korea
	Dong Hoon Lee	Korea University, Korea
	Pil Joong Lee	POSTECH, Korea
	Chae-Hoon Lim	Sejong University, Korea
	Ji-Young Lim	Korean Bible University, Korea
	Dongdai Lin	SKLIS, Chinese Academy of Sciences, China
	Soohyun Oh	Hoseo University, Korea
	Dan Page	Bristol University, UK
	Susan Pancho-Festin	University of the Philippines, Philippines
	Namje Park	ETRI, Korea
	Vassilis Prevelakis	Drexel University, USA
	Kyung-Hyune Rhee	Pukyong National University, Korea
	Pankaj Rohatgi	IBM Research, USA
	Daehyun Ryu	Hansei University, Korea
	Kouichi Sakurai	Kyushu University, Japan
	Chang-ho Seo	Kongju National University, Korea
	Tsuyoshi Takagi	Future University-Hakodate, Japan
	Jeong Hyun Yi	Soongsil University, Korea
	Heung-Youl Youm	Soonchunhyang University, Korea
	Rui Zhang	AIST, Japan
	Jianying Zhou	Inst. for Infocomm Research, Singapore

Table of Contents

Using Templates to Attack Masked Montgomery Ladder Implementations of Modular Exponentiation

Christoph Herbst and Marcel Medwed

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria
{Christoph.Herbst,Marcel.Medwed}@iaik.tugraz.at

Abstract. Since side-channel attacks turned out to be a major threat against implementations of cryptographic algorithms, many countermeasures have been proposed. Amongst them, multiplicative blinding is believed to provide a reasonable amount of security for public-key algorithms. In this article we show how template attacks can be used to extract sufficient information to recover the mask. Our practical experiments verify that one power trace suffices in order to remove such a blinding factor. In the course of our work we attacked a protected Montgomery Powering Ladder implementation on a widely used microcontroller. As a result we can state that the described attack could be a serious threat for public key algorithms implemented on devices with small word size.

Keywords: RSA, Montgomery Ladder, Base Point Blinding, Side-Channel Attacks, Power Analysis, Template Attacks, Microcontroller, Smart Cards.

1 Introduction

In times where huge amounts of data are processed and distributed in electronic form, security becomes a core subject for all these applications. Cryptographic algorithms are an essential instrument to realize reasonable security measures to achieve privacy, authenticity and integrity for electronic data. The security of most cryptographic algorithms relies on the fact that the used key is only known by entitled parties. Another basic concept of cryptography is also that the result (e.g ciphertext, signature) should contain as little information as possible about the involved key.

In practice, these algorithms have to be implemented on physical devices such as PCs, smart cards or embedded devices. These implementations can have weaknesses which were not considered by the designers of the algorithm. Attacks taking advantage of such vulnerabilities are called implementation attacks. Apart from fault attacks ([1], [2]), a very important type of implementation attacks are side-channel attacks. Side-channel attacks make use of the fact that cryptographic devices convey physical information about the processed secret. In his article [3] Kocher showed that timing information can be used to extract the key of an RSA [4] implementation. Amongst side-channel attacks, power analysis [5] has proven to be very potent. Messerges et al. [6] have been one of the first presenting power analysis attacks on modular exponentiation. Since then, the improvement of the attacks and the development of countermeasures to defeat such attacks has been the topic of many scientific publications. A very powerful variety of such attacks are the so called template attacks, first presented by Chari et al. [7].

In power analysis an adversary has to measure the power consumption of a device while it performs the cryptographic algorithm using the sought secret key. In Simple Power Analysis (SPA) an attacker tries to extract the key from one measured power trace (e.g. by visually inspecting the waveform). However, for Differential Power Analysis (DPA) an attacker uses multiple measured traces to extract the key. DPA attacks predict intermediate values, which occur during the execution of a cryptographic algorithm. With these predicted values and a power model of the device under attack, a hypothetical power consumption is calculated and matched to the measured traces. The best match indicates the best prediction for the intermediate values, and therefore the bits of the key which are included in the prediction.

Template attacks are a special case of SPA attacks. The attacker has to collect only one measurement trace of the device under attack, but he has to have a device which behaves similarly concerning the side-channel leakage. This device is used to build templates for the attacked operations in the implementation. For building the templates, several traces have to be measured for each attacked operation and data value. In the template matching phase each template is matched to the single measured trace. The attacker can derive the secret key from the template which fits best. One of the first results about practical implementations of template attacks have been published in [8]. In [9], a template attack on an ECDSA implementation is given.

Besides new and better attacks, the community also worked on countermeasures against side-channel attacks. In principle, there are two types of countermeasures, namely masking and hiding [10]. Masking tries to break the link between the predicted intermediate values and the values processed by the device. Hiding minimizes the effect of the processed intermediate values on the power consumption. Many different approaches of countermeasures have been proposed for all kinds of algorithms on different levels. Algorithmic countermeasures for RSA have already been mentioned in [3]. Most of them either blind the base and/or the exponent of the modular exponentiation. In [11], countermeasures against DPA and SPA attacks have been proposed. Also Fumaroli et al. have developed a "Blinded Fault Resistant Exponentiation" [12] based on the Montgomery powering ladder.

In this article we will show how templates can be used to attack such blinded implementations on 8-bit and 16-bit microcontrollers. As a reference, we will show how to defeat the base blinding of the proposed algorithm by Fumaroli et al. [12]. Nevertheless, our attack can be applied to most base blinding schemes which are used to secure modular exponentiation against side-channel attacks. Countermeasures based on exponent blinding do not defeat template attacks either. This is because the result of an template attack would be a blinded exponent which is still a valid exponent that could be used for encryption or signing.

The rest of this paper is organized as follows. Section 2 explains the algorithm proposed by Fumaroli et al. [12] which we later on use to demonstrate our attack. In Section 3 we give an introduction to the basics of template attacks. The theory of the filtering step, which is used to extract the desired mask bits, is presented in Section 4. Section 5 describes the practical details of the attack, followed by the results in Section 6. Finally, conclusions are drawn in Section 7.

2 Masked Montgomery Powering Ladder

Initially, the Montgomery powering ladder was developed for fast scalar multiplications on elliptic curves [13]. Joye et al. [14] rediscovered it and extended the scope of the Montgomery ladder to any exponentiation in an Abelian group. In the following, $\mathbb{G}$ denotes a finite multiplicative Abelian group. Contrary to classical binary algorithms used for exponentiation the Montgomery powering ladder behaves very regularly. Thus it provides a certain degree of security against side-channel analysis attacks. In Algorithm 1, the Montgomery ladder proposed by Joye et al. is given. Their version includes already some protection against side-channel analysis. This protection is achieved by avoiding conditional branches and performing the same operations on different registers for each execution. In [14], a brief security analysis against side-channel attacks is given. Joye et al. state that if the writing to registers R0 and R1 can not be distinguished from a single side-channel measurement, the Montgomery ladder can be used to implement an exponentiation secure against simple side-channel analysis attacks. Such an implementation would still be vulnerable to differential side-channel attacks.

Algorithm 1. Montgommery ladder for calculating $x^k \in \mathbb{G}$

```
Require: $x \in \mathbb{G}, k = \sum_{i=0}^{t-1} k_i 2^i \in \mathbb{N}$
Ensure: $x^k \in \mathbb{G}$
1: $R_0 \leftarrow 1$, $R_1 \leftarrow x$
2: for $j = t-1$ down to $0$ do
3:     $R_{\neg k_j} \leftarrow R_0 * R_1$; $R_{k_j} \leftarrow (R_{k_j})^2$
4: end for
5: return $R_0$
```

In [12], Fumaroli and Vigilant proposed a blinded and fault-resistant version of the Montgomery Ladder. They use a base blinding technique to make the algorithm secure against side-channel analysis. Additionally they introduce a checksum to counteract fault attacks on the key bits. In our considerations we do not concern ourselves with the checksum calculation (init(CKS) and update(CKS,k_j)) because it does not influence our attack. Algorithm 2 shows the proposed algorithm by Fumaroli. At the beginning of the calculation the registers R_0 and R_1 are multiplicatively masked by a secret random element $r \in \mathbb{G}$. Register R_2 is initialized with $r^{-1} \in \mathbb{G}$. During the calculation, the registers R_0 and R_1 are then masked with the value $r^{2^{t-j}}$ and R_2 holds the compensation value $r^{-2^{t-j}}$. The multiplication of R_0 by R_2 removes the blinding. The security analysis of this algorithm given in [12] remarks that there exist some elements $r \in \mathbb{G}$ with the property that $r^{2^j} = 1$ for some $j \in \mathbb{N}$. If the registers are blinded with such elements, the registers are permanently unmasked after j iterations. But the authors also state that in the context of RSA the probability of choosing such a weak mask is lower than approximately $\frac{1}{2^{900}}$ and therefore it will be very unlikely that one would choose a weak mask in practice. Due to the fact that in nearly all cases the registers R_0 and R_1 are masked with a random value, no differential side-channel analysis attack can be mounted, because the intermediate values are statistically independent from the input and the output.

Algorithm 2. Side-channel analysis and fault-attack resistant Montgomery ladder by Fumaroli et al. [12]

```
Require: $x \in \mathbb{G}, k = \sum_{i=0}^{t-1} k_i 2^i \in \mathbb{N}$
Ensure: $x^k \in \mathbb{G}$
 1: Pick a random $r \in \mathbb{G}$
 2: $R_0 \leftarrow r$; $R_1 \leftarrow rx$; $R_2 \leftarrow r^{-1}$
 3: init(CKS)
 4: for $j = t-1$ down to $0$ do
 5:     $R_{\neg k_j} \leftarrow R_{\neg k_j} R_{k_j}$
 6:     $R_{k_j} \leftarrow R_{k_j}^2$
 7:     $R_2 \leftarrow R_2^2$
 8:     update(CKS,$k_j$)
 9: end for
10: $R_2 \leftarrow R_2 \oplus CKS \oplus CKS_{ref}$
11: return $R_2 R_0$
```

3 Principle of Template Attacks

Template attacks are an advanced form of SPA techniques. In contrary to DPA, only one or a few traces suffice to mount an SPA attack. The simplest case would be a visual investigation of a square & multiply algorithm where the two basic building blocks, namely the squaring and the multiplication, are easily distinguishable. Such a scenario is described by Messerges et al. in [6]. They show how an RSA private key can be extracted by just looking at the power trace. However, since those vulnerabilities are known, they can be avoided. Hence, plain SPA attacks are not likely to be a threat for modern implementations anymore. Therefore, more advanced attack techniques have been developed. The most powerful SPA attacks are represented by template attacks [7]. This is because they extract most information out of a given power trace by characterizing the noise.

Templates describe the statistical properties of every interesting point within the power trace as well as their dependencies among each other. Given this information, whole algorithms, sequences of instructions, or Hamming weights of operands can be characterized. First, the points containing most of the interesting information, have to be found and extracted. For the attack described in this paper we are interested in the Hamming weight of the words of the mask. Hence, a DPA attack is perfectly suited to learn which points leak the processed Hamming weight best. This is because a DPA can be seen as just another metric for the signal to noise ratio (SNR). Given these points, their statistical properties have to be extracted. Since it is assumed that the single points in a power trace follow a normal distribution and that those points are dependent on each other, a well suited probability model is the multivariate normal distribution (MVN). Such a distribution is defined by a mean vector m and a covariance matrix C. Given a set of N power traces, each containing M points (random variables) for a specific operation, we use the following notation. The vectors t_i denote all M variables of the ith trace and the vectors t'_j denote the N observations of the jth random variable. Finally $\bar{t'}_i$ denotes the expected value of the ith variable. The mean vector and the covariance matrix can now be defined as follows:

$$m = \frac{1}{N}\sum_{i=1}^{K} t_i \tag{1}$$

$$\forall c \in C : c_{i,j} = \frac{1}{N}(t'_i - \bar{t'}_i)^\top \cdot (t'_j - \bar{t'}_j) \tag{2}$$

From equation (2) it can be seen that $c_{i,i}$ contains the variance of the ith variable whereas the other elements ($c_{i,j}$ for $i \neq j$) contain the covariance between the ith and the jth variable. If the variables are almost independent the covariance matrix becomes close to singularity and we run into numerical problems. This fact stresses the need for suitably selected points even more.

For every possible occurring operation a pair (m, C) has to be generated. In our attack scenario we have nine possible Hamming weights and hence build (m_i, C_i) for $i = 0..8$. Given these pairs, the nine operations are fully characterized.

The second part of the attack is called classification phase. At that stage we already have the templates and are given the trace to attack. First, we extract the interesting points which we want to identify with a specific operation. Second, those extracted points t are taken as the input for the MVN probability density functions (PDF) described by the pairs (m_i, C_i). After evaluating the functions

$$p(t; (m_i, C_i)) = \frac{1}{\sqrt{(2\pi)^N |C_i|}} e^{-\frac{1}{2}(t-m_i)^\top C_i^{-1}(t-m_i)} \tag{3}$$

where $|C|$ denotes the determinant of the matrix C, we get nine probabilities. Each of them states how likely it is that the operation characterized by (m_i, C_i)

was executed under the presumption that t was observed. Finally, we opt for the operation which yields the highest probability. This is called the maximum-likelihood (ML) decision rule.

4 Mask Filtering

The previous section described how to extract the Hamming weights of the processed data out of a given trace. This section will show how to recover an operand of a multiplication or a squaring, given the processed Hamming weights, by filtering out impossible operand values. For the blinded Montgomery powering ladder algorithm this ability suffices to unmask it. To mount the attack, we can either focus on the squaring of the mask (line 2) or the masking itself (line 2) in Algorithm 2. It turns out, that if the number of words is sufficiently large, the success probability of the attack becomes almost one. Once the words of the mask are known, the effect of the masking can be eliminated and a template attack on the key, like in [9], can be mounted.

Looking at a multiplication algorithm for multi-precision integers like Algorithm 3, the following observations can be made: During the execution of the algorithm, all Hamming weights of the n input words occur exactly n times. Furthermore there occur n^2 partial products each of which consists of two words. This is crucial as it allows a much more precise differentiation. We will use this information to filter impossible masks. This filtering procedure will first be investigated for squarings and later for multiplications.

Algorithm 3. Multi-precision integer multiplication

```
Require: Multi-precision integers $A, B$ consisting of $l$ words
Ensure: $C = A \cdot B$
 1: $C \leftarrow 0$
 2: for $i$ from 0 to $l-1$ do
 3:     $U \leftarrow 0$
 4:     for $j$ from 0 to $l-1$ do
 5:         $(UV) \leftarrow C[i+j] + A[i] \cdot B[j] + U$
 6:         $C[i+j] \leftarrow V$
 7:     end for
 8:     $C[i+t] \leftarrow U$
 9: end for
10: return $C$
```

For the basic filtering of the information gathered during a squaring operation we will use Algorithm 4, where $hw()$ denotes the Hamming weight function and $hb()$ and $lb()$ return the high and low byte of a 16-bit value. All other filtering approaches (e.g. for a multiplication) which we will use, work very similarly and will not be listed explicitly. In the first part of the algorithm (line 4-4) wrong candidates are filtered out. Since there is only one value involved in every iteration, a wrong outcome indicates a wrong candidate with certainty. We will

Algorithm 4. Mask filtering algorithm

```
Require: Hamming weights of the $l$-word long input value $A$ and the partial products.
    The input Hamming weights are stored in $IPHW_i$ and those of the partial products
    in $PPHW_i$. The Hamming weight of the high byte of $A_i$ times $A_j$ is stored in
    $PPHW_{2(i*l+j)+1}$ and the one of the low byte in $PPHW_{2(i*l+j)}$ respectively. A
    word is assumed to be 8 bit long.
Ensure: C = Value of the mask $A$
 1: $PV_{i,j} \leftarrow 1$ for $i = 1..l$ and $j = 1..256$.
 2: for $i$ from 1 to $l$ do
 3:   for $j$ from 1 to 256 do
 4:     if $hw(j) \neq IPHW_i$ then
 5:       $PV_{i,j} \leftarrow 0$
 6:     end if
 7:     if $hw(hb(j^2)) \neq PPHW_{2(j \cdot l+j)+1}$ OR $hw(lb(j^2)) \neq PPHW_{2(j \cdot l+j)}$ then
 8:       $PV_{i,j} \leftarrow 0$
 9:     end if
10:   end for
11: end for
12: for all $((PV_{i,k} \geq 1) AND (PV_{j,l} \geq 1))$ with $i \neq j$ do
13:   if $hw(hb(i \cdot j)) == PPHW_{2(i*l+j)+1}$ AND $hw(lb(i \cdot j)) == PPHW_{2(i*l+j)}$ then
14:     $PV_{i,k} ++$, $PV_{j,l} ++$
15:   end if
16: end for
17: for $i$ from 1 to $l$ do
18:   $C_i \leftarrow index$ of $max(PV_{i,j})$ with $j = 1..256$
19: end for
20: return $C$
```

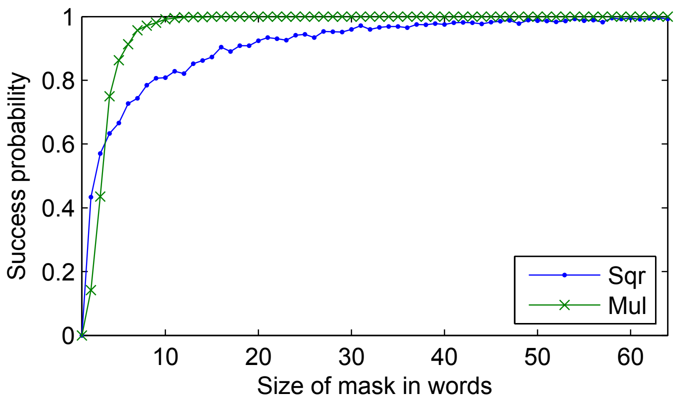


Fig. 1. Success probability of the filtering algorithm depending on the operand size

refer to this as the two sieving steps. In the second part (line 4-4) this cannot be done so easily anymore. This is because a wrong outcome only points out that one of the involved operands is a wrong candidate, but it does not tell which one. Hence, the only possible strategy to make use of this information is to tick both candidates if the outcome has the correct Hamming weight. Hence we will refer to it as the ticking step. Due to this tick approach, it can happen that the returned mask is not the correct one. However, Figure 1 shows that this becomes more and more unlikely with growing operand length.

The difference for a multiplication is that we know one operand completely. Therefore, we just need one sieving step and no ticking step. It can be seen in Figure 1, that the success probability increases much faster for a multiplication. In the next section we will see that an attack on a multiplication also works much better under non-optimal conditions.

4.1 Sieving with Tolerance

It is well accepted in literature that templates can extract the Hamming weight of operands. However, for the above attack we require a 100% accuracy, which is in fact a rather hard requirement. Therefore, we show that we can relax this requirement, by allowing a tolerance. This tolerance is considered during the filtering. It is assumed that the templates reveal the real Hamming weight only with a certain probability. They are allowed to return the real Hamming weight plus/minus a tolerance. First, we take a look at the squaring again. Figure 2 shows the success probabilities for sieving a squaring with a tolerance of one. It becomes clear that without knowing some bits for sure, the attack fails in this scenario. This is because too many wrong positives pass the first two sieving steps and therefore the ticking step fails. However, if some bits are known or even better, if whole words are known, we can decrease the input set PV for the ticking step dramatically in a third sieving step. It turns out that eight to sixteen bits are sufficient to raise the success probability to a reasonable value again. For a tolerance of one and zero known bits, the attack becomes infeasible. For one known word, the success probability falls below one percent for mask sizes greater than ten words. However, for two known words the probability starts to

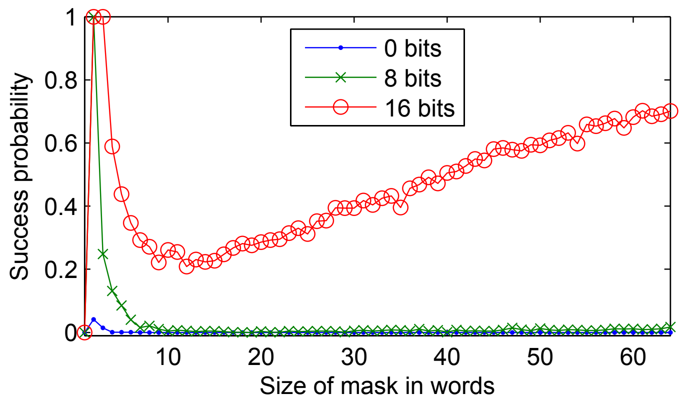


Fig. 2. Success probability of the filtering algorithm with tolerance 1 and different numbers of known bits

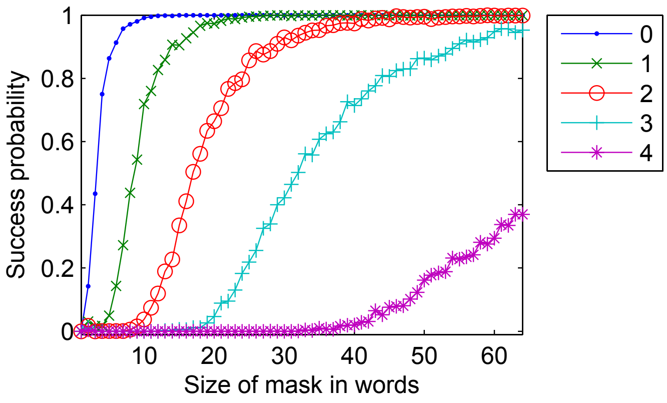


Fig. 3. Success probability filtering an 8-bit multiplication with different tolerances

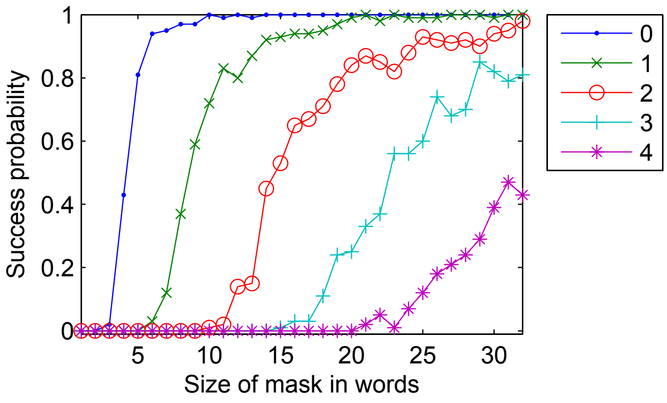


Fig. 4. Success probability filtering a 16-bit multiplication with different tolerances

increase again for masks larger than ten words and already reaches 70 percent for 512-bit operands.

For a multiplication we can relax the requirements even more. Figure 3 shows that the attack is still feasible for a tolerance of four. In other words, for 8-bit operands, one template covers between 50 and 100 percent of the whole spectrum of possible Hamming weights. In Section 6, we show that a hundred percent accuracy of the templates can be achieved on our device for such a tolerance.

Although the sieving step works best with 8-bit, it is still possible to mount the attack on 16-bit architectures, as it can be seen in Figure 4.

5 Practical Attack

The basic idea of the attack is quite simple, if an attacker can extract the random secret value r out of a single measurement he is able to unmask all intermediate values and can apply a conventional DPA or template attack. The core issue is now the extraction of this secret r from one measured trace. Within this section, we explain how to extract the random blinding factor from a single measurement by using a template attack.

A precondition for all kinds of template attacks is that the attacker has a device which behaves identical as the device under attack according to the side-channel leakage. This device is used to build the templates which are later on used to match with one measured side-channel trace from the device under attack. The theory of building templates and matching them is described in Section 3. In practice, an attacker will try not to build a template for the complete instruction sequence which is processing the attacked secret data, but he will try to extract interesting points. These interesting points can be found by performing a DPA attack on known data. When using a correlation attack, the result of the DPA will show the moments in time when the highest correlation between the desired data values and the power consumption of the device will occur. These points in time are chosen to build the templates.

Figure 5 shows the results of a DPA attack on the squaring operation of a 4-word operand. The black trace marks the correlation for one partial product, the gray traces are the correlation for all other partial products. In the black trace, one can identify two groups of correlation peaks with a value of approximately 0.7. These are the moments in time where the products of the words a_0 and a_1 and vice versa are calculated. In Figure 6, the correlation values for an attack on the input values is given. The highlighted trace is the resulting correlation of input word a_0. There are seven groups of peaks with a correlation around 0.6, which identify the moments in time where the input word a_0 is involved in a multiplication.

After identifying all important points in time an attacker can start to build his templates. Our training set consisted of measured traces for 1,000 random input values. For each partial product, templates for all nine possible Hamming weights were built. To build these templates for each partial product the training

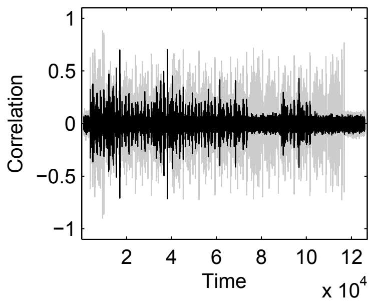


Fig. 5. Result of DPA attack on the intermediate values of a squaring operation for known operands

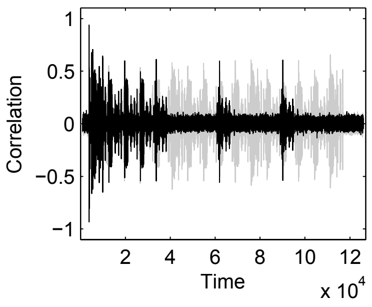


Fig. 6. Result of DPA attack on the input values of a squaring operation

set is partitioned according to the occurring Hamming weights at the position of the partial product. All traces in one subset are then used to build one template.

The templates are then used to identify the Hamming weights of the partial products in one measured trace. An attacker can either use the squaring step of r (Line 7 of Algorithm 2) or the initial multiplication of the input value x with the random secret blinding factor r (Line 2 of Algorithm 2). After identifying the Hamming weights of the partial products, the sieving step explained in Section 4 is applied. If there are enough words available, the result of the sieving step is the random secret value r used in the measured execution of the Montgomery ladder by Fumaroli et al. If the attacker extracts the blinding factor r for each execution, he can perform a classical DPA attack. Using only one measured trace, the attacker can also use another template attack to extract the secret exponent k. To do so, the attacker can use the attack shown in [9] for ECDSA and adapt it to RSA.

6 Results

For our practical experiment we used an assembly implementation of the multiplication used in either the blinding step (Line 2 of Algorithm 2) or the squaring of the blinding factor r (Line 7 of Algorithm 2). The implementation was optimized towards speed and not towards trace quality (e.g., by inserting NOPs, clearing registers). The target device was an 8-bit microcontroller (AT89S8253) from Atmel. The controller is based on the Intel 8052 architecture and features 256 byte internal RAM, 8 registers and an (8x8)-bit hardware multiplier.

We acquired 1,000 measurement traces using random input values and the previously mentioned setup. With these measured traces we have performed a DPA attack to extract the interesting points (See Figures 5 and 6 in Section 5). The prediction accuracy achieved by the templates built using this information can be seen in Table 1. It becomes evident that the accuracy for a tolerance of 0 is rather low. The needed performance of 1.0 is only achieved with a tolerance of 4. However, as shown in Section 4 such a tolerance still results in a reasonable success rate.

Furthermore, we investigated how the word size of the architecture influences the feasibility of such an attack. It turned out that an increasing word size of the architecture did not decrease the success rate. Only the filtering took longer since its complexity depends on the word size. This means that the practical feasibility of the attack is limited by the word size (computational power needed for filtering) and by the operand length in words (success rate of filtering). The largest word size we successfully attacked in our experiments was 16. We did not

Table 1. Success rate of the templates allowing different tolerances

Tolerance	0	1	2	3	4
Performance	0.3593	0.7652	0.9374	0.9874	1.0000

examine larger word sizes. However, most of the microcontroller market share belongs to 8-bit and 16-bit controllers.

7 Conclusions

In this article we presented how templates can be used to attack blinded public-key algorithms. We showed that a multiplicative random mask can be extracted and removed using only a single power trace. Moreover, we could verify our theoretical assumptions with empirical results, produced on a commonly used microcontroller architecture. These practical experiments have shown that the success rate is already 40% for a 512-bit RSA and increases with growing operand length. Therefore the attack is realistic and presents a serious threat to a wide range of RSA, ElGamal, and ECC implementations which rely on base blinding as a side-channel attack countermeasure. To the best of our knowledge this article is the first to deal with side-channel attacks on masked public-key algorithms.

Acknowledgements. The research described in this paper has partly been supported by the Austrian Science Fund (FWF) under grant number P18321-N15 ("Investigation of Side-Channel Attacks") and by Secure Business Austria (SBA).

References

1. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
2. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
3. 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)
4. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
5. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
6. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Power analysis attacks of modular exponentiation in smartcards. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 144–157. Springer, Heidelberg (1999)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
8. Rechberger, C., Oswald, E.: Practical template attacks. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 443–457. Springer, Heidelberg (2005)
9. Medwed, M., Oswald, E.: Template Attacks on ECDSA. Cryptology ePrint Archive, Report 2008/081 (2008), http://eprint.iacr.org/

10. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks- Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
11. Kim, C., Ha, J., Moon, S., Yen, S.-M., Lien, W.-C., Kim, S.-H.: An Improved and Effcient Countermeasure against Power Analysis Attacks. Cryptology ePrint Archive, Report 2005/022 (2005), `http://eprint.iacr.org/`
12. Fumaroli, G., Vigilant, D.: Blinded fault resistant exponentiation. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 62–70. Springer, Heidelberg (2006)
13. Montgomery, P.L.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Mathematics of Computation 48(177), 243–264 (1987)
14. Joye, M., Yen, S.-M.: The montgomery powering ladder. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2003)

Template Attacks on ECDSA*

Marcel Medwed[2,3] and Elisabeth Oswald[1,2]

[1] University of Bristol, Computer Science Department, Merchant Venturers Building, Woodland Road, BS8 1UB, Bristol, UK

[2] Graz University of Technology, Institute for Applied Information Processing and Communications, Inffeldgasse 16a, 8010 Graz, Austria
Elisabeth.Oswald@bristol.ac.uk

[3] Secure Business Austria (SBA), Favoritenstraße 16, 1040 Vienna, Austria
Marcel.Medwed@iaik.tugraz.at

Abstract. Template attacks have been considered exclusively in the context of implementations of symmetric cryptographic algorithms on 8-bit devices. Within these scenarios, they have proven to be the most powerful attacks. In this article we investigate how template attacks can be applied to implementations of an asymmetric cryptographic algorithm on a 32-bit platform. The asymmetric cryptosystem under scrutiny is the elliptic curve digital signature algorithm (ECDSA). ECDSA is particularly suitable for 32-bit platforms. In this article we show that even SPA resistant implementations of ECDSA on a typical 32-bit platform succumb to template-based SPA attacks. The only way to secure such implementations against template-based SPA attacks is to make them resistant against DPA attacks.

1 Introduction

Template attacks bring together statistical modelling and power analysis techniques. They consist of two phases, which can happen sequentially in time (i.e. the first phase is completed before the second phase starts) or interleaved (i.e. there are several instances of first and second phases and they interleave each other). In the first phase, the attacker builds templates, i.e. statistical models for a device executing a certain sequence of instructions using fixed data. In the second phase, the attacker matches the templates with the traces acquired from the device under attack. The attacker chooses the templates for matching based on key hypotheses. The hypothesis that corresponds to the correct key always indicates the correct templates, and hence leads to the best matches. This allows determining the key.

The study of template attacks is also practically relevant because they essentially give a bound on the number of traces that are needed for a power analysis

* The work of the first author has been supported in part by the Austrian Science Fund (FWF) under grant number P18321 and by Secure Business Austria (SBA). The work of the second author has been supported in part by the EPSRC under grant number EP/F039638/1.

attack. This can serve as a measure for the resistance of a device against certain types of power analysis attacks. Further, there is less research available on the application of template attacks, which makes them an interesting research topic.

Research on template attacks has concentrated mainly on their application to implementations of symmetric cryptographic algorithms. The reason for this is that the number of templates that need to be built, and the size of the templates, determines the practical feasibility of such attacks. Simply put, it is not obvious how to apply template attacks, such as they were described for implementations of symmetric cryptographic algorithms on 8-bit platforms, to implementations of asymmetric cryptographic algorithms on more challenging platforms. We are not aware of any article tackling this problem to the best of our knowledge.

This article provides the first results on template attacks on implementations of asymmetric cryptographic algorithms on a 32-bit platform. To be more specific, we describe how a template-based SPA attack can be performed to break implementations of the Elliptic Curve Digital Signature Algorithm (ECDSA). We describe why and how our attacks work, and how ECDSA actually helps to conduct these attacks. In addition to the theoretical description, we also show results of a practical implementation of our attacks on a 32-bit processor.

This article is organized as follows. In Sect. 2 we review previous work on template attacks. In Sect. 3 we discuss ECDSA and power analysis attacks. In Sect. 4 we outline three important observations for the practical application of template-based SPA attacks on ECDSA and we analyse different scenarios for them. In Sect. 5 we describe the template-based SPA attack on our ECDSA implementation in detail. We summarize this contribution in Sect. 6.

2 Template Attacks

Template attacks exploit the fact that the power consumption of a device depends on the data it processes. The power consumption can be characterized by a multivariate normal distribution. In contrast to simple and differential power analysis (SPA and DPA), template attacks consist of two phases. In the first phase, the attacker builds the templates, i.e. characterizes the device. In the second phase, the templates are used for an attack. Template attacks can be used in an SPA and in a DPA scenario, see for instance [1].

In this article, we focus on template attacks in an SPA scenario. Our definition of SPA attacks is that they exploit key-dependent differences within a trace, see [1, p. 102]. We call these attacks template-based SPA attacks and make the following assumptions. During the characterization phase, the attacker has the opportunity to characterize the device under attack. This means, the attacker may invoke the instructions to be characterized with data of her choice. There is no limit on the number of invocations. In other words, the attacker has full control over the device. During the attack phase, we assume that the attacker has only very limited access to the device. Like in a single-shot SPA attack, we assume that the attacker only gets a single trace for an execution of ECDSA on the attacked device.

2.1 Template Building Phase

Characterizing the device (or the power consumption of a device) means determining the probability distribution of the power consumption of certain instructions. It is common to assume that the probability distribution of the power consumption is a multivariate normal distribution. This distribution is defined by a covariance matrix $\mathbf{C}$ and a mean vector $\mathbf{m}$, see (1).

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2 \cdot \pi)^n \cdot det(\mathbf{C})}} \cdot exp\left(-\frac{1}{2} \cdot (\mathbf{x} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{x} - \mathbf{m})\right) \qquad (1)$$

The covariance matrix $\mathbf{C}$ contains the covariances $c_{ij} = Cov(X_i, X_j)$ of the (interesting) points at time indices i and j. The mean vector $\mathbf{m}$ lists the mean values $m_i = E(X_i)$ for all (interesting) points in the trace. We refer to this pair $(\mathbf{m}, \mathbf{C})$ as a *template* from now on.

As written before, we assume that we can determine templates for certain sequences of instructions. This means, we execute these sequences of instructions with different data d_i and keys k_j in order to record the resulting power consumption. Then, we group together the traces that correspond to a pair of (d_i, k_j), and estimate the mean vector and the covariance matrix of the multivariate normal distribution. As a result, we obtain a template for every pair of data and key (d_i, k_j): $\hbar_{d_i,k_j} = (\mathbf{m}, \mathbf{C})_{d_i,k_j}$.

In Mangard et al.[1] the notation of reduced templates has been introduced. Reduced templates are templates for which the covariances can be neglected. In addition there are different strategies for template building described in [1]. The different strategies lead to rather different complexities in the template building phase. In particular, building templates for intermediate values having already a power model in mind makes template attacks much more feasible in practice. We will make use of this strategy in this paper.

2.2 Template Matching Phase

During the template matching phase, we use the characterization together with a power trace from the device under attack to determine the key. This means, we evaluate the probability density function of the multivariate normal distribution with $(\mathbf{m}, \mathbf{C})_{d_i,k_j}$ and the power trace of the device under attack, i.e. we compute the probability:

$$p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_i,k_j}) = \frac{exp\left(-\frac{1}{2} \cdot (\mathbf{t} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{t} - \mathbf{m})\right)}{\sqrt{(2 \cdot \pi)^T \cdot det(\mathbf{C})}} \qquad (2)$$

We compute this probability for every template. As a result, we get probabilities for all templates: $p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_1,k_1}), \ldots, p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_D,k_K})$. These probabilities measure how well the templates fit to a given trace. The highest probability indicates the correct template. Because each template is also associated with a key, we can derive the key that is used in the device.

2.3 Previous Work

Template attacks were introduced by Chari et al. [2]. This original work has been investigated further by Rechberger and Oswald [3], who studied practical aspects of template attacks on implementations of RC4 on an 8-bit microcontroller. A result from these first two articles is that the choice and number of the so-called interesting points (i.e. the points in the power traces that are used to build the templates) is crucial for the success of template attacks. It turns out that an approach that works well in practice is to choose points that lead to high correlation values in DPA attacks as interesting points.

In later work, Agrawal et al. [4] studied how to use template attacks to attack masking if the random number generator that produces the masks is biased during characterization. Archambeau et al. [5] investigated how principal subspaces can be used to make template attacks more effective. Oswald and Mangard demonstrated different types of template attacks on masked implementations of AES on an 8-bit smart card. They pointed out that in this particular scenario, template-based DPA attacks on a masked implementation are as effective as the same type of attack on an unmasked implementation. A comprehensive description of template attacks (in SPA and DPA scenarios) is given in Mangard et al. [1].

Previous work has solely focused on the application of template attacks to implementations of symmetric cryptographic algorithms such as AES and RC4. There seem to be two reasons for this.

The first and probably most important reason is that templates were often built for pairs of key and data. When considering asymmetric cryptographic algorithms this is obviously a problem because the numbers that are used in asymmetric cryptography are huge. For instance, in elliptic curve cryptography, field sizes are 2^{160} and larger. It is simply infeasible to build templates for all numbers in a finite field of size 2^{160} that can be used to represent coordinates of elliptic curve points. In addition, it is infeasible to build templates for all points on a given elliptic curve. In contrast, it is feasible to build templates for all 256 values that one byte of the AES state can take (the AES field size is 2^8).

The second reason is that previous work was all experimentally verified on 8-bit microcontrollers. This is mainly because they are easy to get and to use for such experiments. To the best of our knowledge only Gebotys et al. [6] have worked with 32-bit microprocessors. Their previous work shows that working with such platforms is more challenging than 8-bit platforms. In order to study template attacks on asymmetric cryptographic algorithms and verify attacks experimentally, one needs to have a suitable measurement setup.

2.4 Our Contribution

In this article we have found ways to overcome problems encountered in previous work. We are able to build templates for the intermediate points that occur in the ECDSA signature generation. Using these templates, we can break ECDSA implementations in different attack scenarios, which are of general interest for template attacks on public-key cryptosystems. We have verified our attacks experimentally on a 32-bit microprocessor. The microprocessor is based on an

ARM7 architecture which is widely used in handheld computers, etc. Hence, our verification is done on a platform that is relevant in practice. This is the first work that successfully applies template attacks on implementations of ECDSA on a realistic platform.

3 ECDSA and Power Analysis Attacks

In this section we explain why ECDSA is very well suited as target for template-based SPA attacks. First, we briefly describe the ECDSA signature generation. Then we review briefly common strategies to secure implementations of ECDSA. Last, we spell out three observations that can be combined in order to apply template-based SPA attacks to ECDSA implementations.

3.1 ECDSA Signature Generation

ECDSA is the elliptic curve version of the digital signature algorithm (DSA) [7]. This algorithm computes a signature, i.e. a pair of numbers (r, s), for a given message m. It is a probabilistic algorithm because the computation of the value r is based on an ephemeral key k. This ephemeral key changes in every signature operation. Hence, two signature operations on the same message m lead to two different signature values.

Essentially, the difference between ECDSA and DSA is in the calculation of the value r in the signature (r, s). This step involves operations on the specified elliptic curve using the ephemeral key k, see (3). The calculation of the value s makes use of the (static) secret key d, see (4). In (3) and (4), the variable P denotes the base point, and $h(m)$ denotes the hash value of the message m. The scalar multiplication that multiplies the base point P with the ephemeral key k is denoted as $[k]P$.

$$r = x_1 \pmod n, (x_1, y_1) = [k]P \tag{3}$$

$$s = k^{-1}(h(m) + d \times r) \pmod n \tag{4}$$

Breaking a signature scheme typically means either having the ability of producing valid signatures without knowing the secret key d, or, knowing the secret key d. The last meaning of breaking is often referred to as a full break.

For ECDSA, when the parameters are chosen appropriately, for instance by taking one of the recommended curves by NIST [7], theoretical breaks are computationally infeasible. However, if for instance, the ephemeral key is known for a given signature (r, s) then the secret key d can be determined easily from (3) and (4). Even worse, it is sufficient if a small number of bits of the ephemeral key from a couple of signatures are known, lattice attacks can reveal the entire secret key d with low computational effort, see [8] for some theoretical background on lattice attacks and [11] for a practical implementation of the same attacks.

This is where SPA attacks come into play. If an attacker is able to reveal by SPA either the entire ephemeral key, or just a couple of bits of several ephemeral keys, ECDSA implementation can be broken in practice.

3.2 Security of Implementations of ECDSA

Implementations of ECDSA may allow revealing the secret key if they leak information about the ephemeral key or the secret key itself. Hence, the computation of r and s both need to be protected.

The computation of r makes use of the ephemeral key k. This ephemeral key is unknown to the attacker and it changes in each signature operation. Hence, DPA attacks are not possible on this operation, but only SPA attacks.

The computation of s makes use of the secret key d. If the attacker has knowledge about the signature value r, DPA attacks targeting the computation of $d \times r$ are possible.

In this article, we focus on attacks on the computation of the r. It is well known that implementations that show visible differences between EC point doubling and EC point addition operations are highly susceptible to SPA attacks [9]. There are abundant countermeasures against these types of SPA attacks. For instance, unified point operations or Montgomery-type multiplication techniques, see [10] for a comprehensive overview. In addition, scalar blinding has been listed in [9] to prevent SPA attacks. However, all these countermeasures tackle visible differences (i.e. differences that can be exploited using a visual inspection of power traces) only. According to the definition of SPA attacks that we use, any key-dependent leakage within a trace can be exploited. Consequently, countermeasures that only prevent operation-dependent leaks (i.e. so-called SPA leaks) are not sufficient to prevent attacks that use data-dependent leaks (i.e. so-called DPA leaks, see [12] for definitions).

4 Template-Based SPA Attacks on ECDSA

Countermeasures against SPA attacks might not prevent template-based SPA attacks because they exploit not only operation dependent leakages but also data dependent leakages. In this article we show that this is indeed possible for realistic implementations on an 32-bit platform.

4.1 Three Observations for Practical Applications

There are several observations or facts that when combined allow template-based SPA attacks on ECDSA implementations. The first observation, which we mentioned in the previous section, is that knowledge of only a small number of bits of several ephemeral keys of ECDSA signatures is sufficient to determine the secret key via lattice attacks.

The second observation, which is crucial for the practical realization, is that the EC operation in the computation of r in ECDSA uses a fixed base point. This is important because it means that the first couple of bits of k that are processed during the computation of r can only lead to a small set of points (i.e. the multiples of P). This means that we only need to build templates for the points in this (small) set.

The third observation, which is also crucial for the practical implementation of template attacks, is that for software implementations of ECDSA on 32-bit processors, it is sufficient to build templates for intermediate values taking a suitable power model of the device into account. Especially for microprocessors previous work has shown that busses typically leak the Hamming weight (or sometimes the Hamming distance) of the operands. This leads to a dramatic simplification in practice. For instance, instead of building 2^{32} templates for a move instruction we only need to build 33 templates when we know that the device essentially leaks the Hamming weight.

4.2 Different Scenarios for Practical Attacks

These three observations make it clear that template-based SPA attacks are indeed feasible in practice: in order to determine the first f bits of the ephemeral key with a template-based SPA attack, we need to build templates to detect the 2^f multiples of the base point P. It is possible to build such templates because the set of points is rather small and because we can build templates that take the power model of the device into account.

As mentioned in Sect. 1, an attacker might have different ways of working with the characterization and the matching phase (characterization on beforehand or characterization on-the-fly). In combination with the scenario of implementations of public-key algorithms, this leads to a number of different variations of how template-based SPA attacks could be used.

Template Creation on Beforehand

Attacker has full control including secret-key operations. The most traditional assumption for template attacks is that the attacker has full control over the device (or one that is similar to the device under attack) at all times. For instance, an attacker is able to generate secret keys (known to him) and use them to characterize a sequence of instructions (e.g. a point addition) that is part of the $[k]P$ operation. Under this assumption, an attacker could generate templates to detect all 2^f multiples of the base point and use them to extract f bits of k, as described before.

Attacker has limited control excluding secret-key operations. If an attacker cannot invoke secret-key operations with known key material another strategy is required. In the case of public-key cryptography, the attacker can make use of public-key operations, if they use the same implementation as the secret-key operations use. An attacker could be able to generate public keys and use them to characterize a sequence of instructions (e.g. a point addition) that is part of an EC point multiplication.

For instance, in ECDSA the signature verification requires to compute (5)-(9), given a signature (r, s) on a message m and a public key Q.

$$e = HASH(m) \tag{5}$$
$$w = s^{-1} \pmod{n} \tag{6}$$
$$u_1 = e \times w \pmod{n} \tag{7}$$
$$u_2 = r \times w \pmod{n} \tag{8}$$
$$X = (x1, y1) = [u_1]P + [u_2]Q. \tag{9}$$

It is clear that (9) includes an operation (the computation of $[u_1]P$ that is similar to the operation that uses the ephemeral key (i.e. $[k]P$). The attacker only needs to select u_1 appropriately in order to generate templates for the various multiplies of the base point P. In order to generate appropriate values of u_1 the attacker can proceed as follows. The attacker chooses a message m at random and computes e according to (5). Next, the attacker picks the value f of the multiple of the base point, and computes $s = f^{-1} \times e \pmod{n}$. The signature verification[1] applied to a signature (r, s) (r chosen at random, s chosen as described before), on a message m, using a public key Q will then require the computation of $[u_1]P$ with $(u_1 = e \times w = f)$ in (9). This allows the attacker to build templates for multiples f of the base point P.

Template Creation On-the-Fly

Attacker has full control including secret-key operations. The scenario described before can easily be adapted to a case where the attacker can interleave characterization and matching and hence build templates on-the-fly. In this latter case, the attacker simply invokes the secret-key operations that are being characterized with known and unknown secret-key material. For instance, suppose that the attacker has already determined several bits of the ephemeral key k and attempts to recover the next bit. In this case, only two EC points can processed in the subsequent EC scalar multiplication algorithm. Consequently, the attacker builds templates for the two points in question on-the-fly and checks out via template matching which is the correct point. This gives the next bit of k.

Attacker has limited control excluding secret-key operations. Also in this case the scenario as described in the previous section can be easily adapted to on-the-fly characterization. As described in the previous paragraph, if the scenario is to recover the next bit of k, the attacker only needs to build templates to distinguish two EC points. In case the attacker has to rely on the public-key operations, the choices of u_1 can be made such that these two points can be characterized.

5 Practical Template-Based SPA Attacks on EC Point Multiplication

In order to demonstrate the feasibility of the attack strategies that we described in the previous section, we have implemented ECDSA on a suitable platform.

[1] The signature verification will return that the signature is invalid.

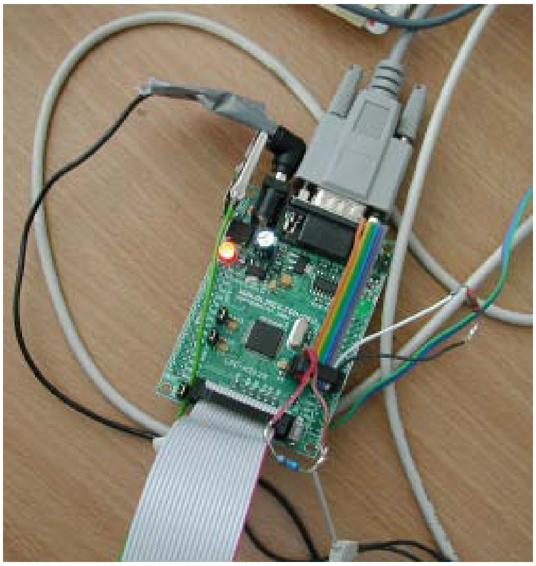


Fig. 1. Customized board with a 32-bit processor based on the ARM7 architecture

One of the most widely used 32-bit processor architectures today is the ARM7. Hence, we have decided to use an ARM7-based platform for our practical experiments. Figure 1 shows our ARM7-based platform that we use for power analysis attacks.

Our ECDSA implementation is optimized for the P192 NIST curve. We have taken care that the optimizations do not allow conducting SPA attacks using visual inspection: all finite field operations have data independent running time. We also make sure that variances that are due to microarchitectural features of the microprocessor, such as the pipeline or the early termination in the multiplier, do not lead to SPA vulnerabilities. To make sure that there are no SPA problems in the point multiplication algorithm we have implemented a typical double-and-always-add algorithm. We have also implemented window-versions of this algorithm. Summarizing, we have made sure that our implementation does not allow reading off the ephemeral key by a visual inspection. Hence there are no SPA leaks, but only DPA leaks in our implementation.

We can exploit these DPA leaks using template-based SPA attacks. In the subsequent sections, we first explain how we built the templates, and then we describe two scenarios to extract the ECDSA secret key by template-based SPA attacks on the ECDSA ephemeral key. Thereafter, we discuss the effectiveness of common countermeasures against our attacks.

5.1 Template Building Phase

In order to find the interesting points for building templates, we performed DPA attacks on the EC point multiplication operation. This means, we executed the

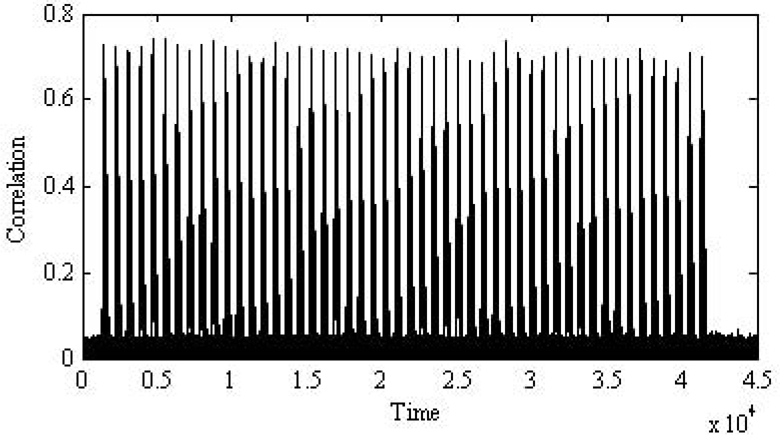


Fig. 2. Correlation coefficients of intermediate values of an EC operation

EC point multiplication algorithm several times with different input data and correlated the Hamming weight of a number of intermediate values to the power traces.

Figure 2 shows the results of several such DPA attacks. Each peak in Fig. 2 corresponds to an intermediate value that leads to a high correlation and is therefore a candidate for selection as point of interest. In one step of this double-and-always-add algorithm, there are about 200 intermediate values that lead to high correlation coefficients. We have decided to select the points of interest by following the rule-of-thumb that Rechberger and Oswald gave in [3]: for each intermediate value that leads to a high correlation coefficient, we take the point in the power trace that leads to the highest absolute value in the correlation trace. This strategy helps to reduce the size of the templates because it restricts the number of points per intermediate value.

Nevertheless, having only this restriction still would lead to rather large templates. This means that templates would consist of a large number of interesting points which often leads to numerical problems, see [1, p. 108]. Hence we have decided to impose one further restriction on the selection of the points of interest. We have decided to select intermediate values that are “far away” (in terms of their Hamming distance) from their corresponding points in another template. Remember that we only have to build a small number of templates: if we build all templates before we start the actual attack (such as assumed in a classical template attack setting), we only need to build 2^f templates (one for each elliptic curve point that can occur at the beginning of the point multiplication algorithm). For each of these 2^f templates we have about 200 intermediate values that we can include. Some of these intermediate values will be very close in terms of their Hamming distance, while others will be farther away. The intermediate values that are farther away lead to templates that can be distinguished easier, because the mean vectors of their probability distributions are farther away.

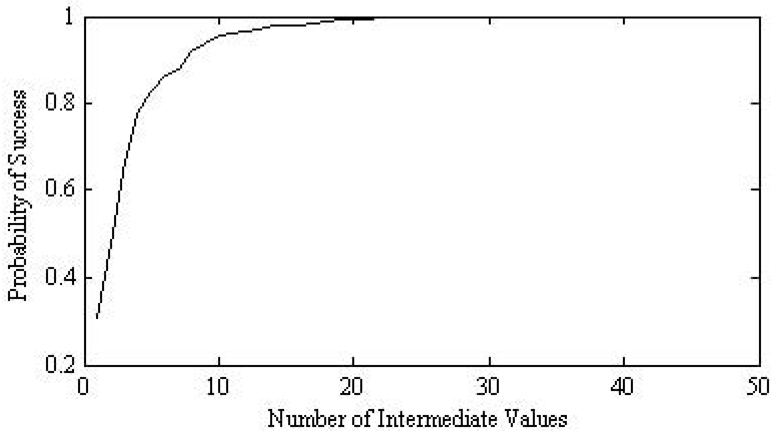


Fig. 3. Probability of success for an increasing number of intermediate values

With these considerations in mind, we have investigated the probability of success for the template matching for an increasing number of intermediate values. It turns out that for about 50 intermediate values (satisfying the property described before), the template matching succeeds with almost certainty, see Fig. 3.

5.2 Template Matching with Pre-computed Templates

In this section we describe the template matching step of a template-based SPA attack with pre-computed templates. In the scenario of attacking ECDSA implementations, the template matching phase includes acquiring one power trace for one ECDSA signature generation operation. The ECDSA operation is computationally expensive and hence acquiring a power trace (with reasonable quality) for an entire EC point multiplication might be not possible in practice. For instance, the memory of the data acquisition instrument might be too limited to store the entire trace. Hence, in practice we might only get a part of the power trace as input for a power analysis attack.

For our template-based SPA this limitation does not matter. We have pointed out already (for instance in Sect. 2.4), that lattice attacks require only a small number of bits of several ECDSA operations. Using template-based SPA attacks we can extract these bits step by step. In order to extract the first bit, we match the first two multiples of the base point with the corresponding part of the newly acquired power trace (if we attack a window method, we match the first 2^w points, where w corresponds to the window size). The template matching results in matching probabilities, see (2). Hence the template that leads to the highest probability indicates the correct bit of the ephemeral key. By applying this method iteratively, we can extract f bits of the ephemeral key. From then on we proceed with a lattice attack in order to derive the ECDSA secret key.

Our practical attack using this strategy succeeds with probability close to one because our template matching succeeds with probability close to one.

5.3 Template Matching with On-the-Fly Created Templates

In all previous work, template attacks have been considered in the symmetric setting. As a consequence, the assumption has developed that template building always occurs before template matching. It appears that this is due to the fact that in the symmetric setting, it seems unlikely that the device under attack allows access to a known (and changeable key) and an unknown key at the same time. The asymmetric setting however is rather different. When we attack for instance an ECDSA signature generation operation with an unknown secret key, it is plausible to assume that we may invoke the ECDSA signature verification operation with (different) known public key(s). We have described several different scenarios for practical template attacks in Sect. 4.2.

This changes the situation for template building and matching. Now we can assume that we can interleave the template building with the matching steps. For instance, we can first use the ECDSA signature verification operation to build a template for a pair of EC points (again assuming a double-and-always-add algorithm, and producing a suitable value for u_1 such as described in Sect. 4.2). Then, we match the two points in order to extract one bit of the ephemeral key. Repeating this procedure for the remaining bits allows deriving the secret key using lattice attacks, such as in the attack that we described in the previous section.

An advantage here is that we can use different sets of interesting points for different pairs (assuming a double-and-always-add algorithm) of templates. Remember that we pointed out that we had to impose an important restriction for the selection of interesting points: we selected those points only that were far away (from the corresponding point in the other template) in terms of their Hamming weights. In this scenario, we can adaptively chose interesting points for each pair of on-the-fly generated templates.

5.4 Countermeasures

We have conducted our experiments on implementations of ECDSA that use typical SPA-resistant point multiplication algorithms. Such algorithms do not prevent template-based SPA attacks because they only take care of SPA leaks (operation dependent leaks that are easy to identify using visual inspection).

However, also certain blinding techniques do not prevent our attacks. For instance, scalar blinding (i.e. using $k' = k + ran * ord(P)$, where ran is a random number) does not prevent our attacks. This is because k' and k are still congruent modulo the order of the base point and hence fulfill the ECDSA signing equation. This means, in a lattice attack only the size of the lattice grows, but the attack can still succeed if more bits of k' can be extracted via power analysis.

Point blinding (i.e. using $P' = ran * P$) does not necessarily protect against our attacks either. This is because the point blinding operation can be attacked using our technique in order to determine P'. With P' our original attack can be mounted in order to extract some bits of k. Hence, the attack is still feasible, although it requires a more effort in template building. This means, we either build a larger set of templates before the attack, or we switch to our template creation on-the-fly technique.

As a matter of fact, the only types of countermeasures that prevent our template-based SPA attacks are countermeasures that provide DPA resistance by randomizing the coordinates of the base point P.

6 Conclusion

We have described template-based SPA attacks for implementations of ECDSA. It has turned out that such attacks can break implementations that are secure against other types of SPA attacks. We have combined three observations in our attacks. First, in order to determine an ECDSA secret key, only few bits of several ECDSA ephemeral keys need to be known. Second, ECDSA uses a fixed base point. Hence, we need to build templates only for a number of multiples of this base point. Third, for typical software implementations of ECDSA on microprocessors, it is sufficient to build templates for intermediate values taking the power model of the microprocessor into account. We have also discussed four different scenarios for building templates. These scenarios depends upon the amount of control the attacker has over the device and whether templates are built on beforehand or on-the-fly. It has turned out that because we are working with public-key cryptography, the attacker can even make use of the signature verification operation for template building. In our practical implementation of template-based SPA attacks, we have chosen the interesting points for templates according to criteria that increase the success probability in the matching phase. We have aimed at reaching a success probability that is close to one. Hence, our attacks succeed with almost certainty on our practical implementation. It turns out that our attacks (in the presented scenario) can only be prevented by DPA countermeasures that randomize the coordinates of the base point. This article presents the first work on template attacks in the context of implementations of asymmetric cryptographic algorithms.

References

1. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks – Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
2. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
3. Rechberger, C., Oswald, E.: Practical template attacks. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 443–457. Springer, Heidelberg (2005)
4. Agrawal, D., Rao, J.R., Rohatgi, P., Schramm, K.: Templates as master keys. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 15–29. Springer, Heidelberg (2005)

5. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
6. Gebotys, C.H., Ho, S., Tiu, C.C.: EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 250–264. Springer, Heidelberg (2005)
7. National Institute of Standards and Technology (NIST): FIPS-186-2: Digital Signature Standard (DSS) (2000), `http://www.itl.nist.gov/fipspubs/`
8. Nguyen, P.Q., Shparlinski, I.E.: The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. Design, Codes and Cryptography 30, 201–217 (2003)
9. Coron, J.S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
10. Joye, M.: V, Defences Against Side-Channel Analysis. In: Advances In Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, vol. 317, pp. 87–100. Cambridge University Press, Cambridge (2005)
11. Demuth, M.: Lattice attacks on ECDSA. Master's thesis, Graz University of Technology (2006)
12. Jaffe, J.: Introduction to Differential Power Analysis. In: ECRYPT Summerschool on Cryptographic Hardware, Side Channel and Fault Analysis (2006)

Compact ASIC Architectures for the 512-Bit Hash Function Whirlpool

Takeshi Sugawara[1], Naofumi Homma[1], Takafumi Aoki[1], and Akashi Satoh[2]

[1] Graduate School of Information Sciences, Tohoku University
{sugawara,homma}@aoki.ecei.tohoku.ac.jp, aoki@ecei.tohoku.ac.jp
[2] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology
akashi.satoh@aist.go.jp

Abstract. Compact hardware architectures are proposed for the ISO/IEC 10118-3 standard hash function Whirlpool. In order to reduce the circuit area, the 512-bit function block $\rho[k]$ for the main datapath is divided into smaller sub-blocks with 256-, 128-, or 64-bit buses, and the sub-blocks are used iteratively. Six architectures are designed by combining the three different datapath widths and two data scheduling techniques: interleave and pipeline. The six architectures in conjunction with three different types of S-box were synthesized using a 90-nm CMOS standard cell library, with two optimization options: size and speed. A total of 18 implementations were obtained, and their performances were compared with conventional designs using the same standard cell library. The highest hardware efficiency (defined by throughput per gate) of 372.3 Kbps/gate was achieved by the proposed pipeline architecture with the 256-bit datapath optimized for speed. The interleaved architecture with the 64-bit datapath optimized for size showed the smallest size of 13.6 Kgates, which requires only 46% of the resources of the conventional compact architecture.

Keywords: Hash function, Whirlpool, Hardware architecture, Cryptographic hardware.

1 Introduction

Whirlpool [1, 2] is an ISO/IEC 10118-3 [3] standard 512-bit hash function based on the Miyaguchi-Preneel scheme using the SPN-type compression function W with the 512-bit round function $\rho[k]$ that is similar to the block cipher AES [4]. High-performance hardware architectures for Whirlpool were proposed and their performances were evaluated using the ASIC library described in [5]. However, the 512-bit function $\rho[k]$ requires a large amount of hardware resources, resulting in a much larger circuit area compared to other hash functions, such as SHA-256/-512 [3, 6]. Several compact architectures were also examined in [7, 8, 9], but these architectures are limited to the 64- or 8-bit datapath-width on an FPGA supporting a large built-in memory (i.e., Block RAM).

In this paper, we propose compact hardware architectures for ASIC implementations, which partition the function block $\rho[k]$ into sub-blocks with 256-, 128-, or 64-bit datapath-widths. The proposed architectures generate round keys on the fly to eliminate the requirement for memory resources to hold pre-calculated round keys. This feature is especially effective in ASIC implementations in which memory is expensive. The pipelining technique for the compact architectures is also investigated to achieve higher throughput with a smaller circuit area. In total, six hardware architectures for Whirlpool that combine two schemes (interleave and pipeline) with the three datapath-widths are designed. The corresponding throughput and gate count are then evaluated using a 90-nm CMOS standard cell library. Performance comparisons with the Whirlpool hardware using a 512-bit datapath-width [5] and the SHA-256/-512 hardware [10] synthesized using the same library are also presented.

2 The 512-Bit Hash Function Whirlpool

The Whirlpool compression function W has two datapaths, and in each path, the 512-bit function ρ[k] is used 10 times, as shown in Fig. 1. One of the paths receives the 512-bit hash value H_{i-1} that was generated from 512-bit message blocks m_1~m_{i-1} in the previous cycles and then outputs ten 512-bit keys K^1~K^{10} by using ten 512-bit constants c^1~c^{10}. Another path processes the current message block m_i using the keys K^1~K^{10}. No hash value is calculated before receiving the first message block m_1, and thus 0 is assigned to H_0.

The function $\rho[k]$ consists of four 512-bit sub-functions γ, π, θ, and $\sigma[k]$, which are similar to *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*, respectively, of AES. Each sub-function treats a 512-bit block as an 8×8-byte matrix. The first function γ is a nonlinear substitution function consisting of sixty-four 8-bit S-boxes, and the S-box is defined as a combination of three types of 4-bit mini-boxes, namely, E, E^{-1}, and R. The following function π rotates each row of the matrix by 0 ~ 7 bytes. The function θ then operates matrix multiplication using the parameters shown in Fig. 1. When the individual input and output bytes of the multiplication are defined as a_{ij} and b_{ij} ($0 \le i, j \le 7$), respectively, the eight bytes of the column j = 0 are calculated as

$$b_{i0} = a_{i0} \oplus (9 \otimes a_{i1}) \oplus (2 \otimes a_{i2}) \oplus (5 \otimes a_{i3}) \oplus (8 \otimes a_{i4}) \oplus a_{i5} \oplus (4 \otimes a_{i6}) \oplus a_{i7},$$

where the byte multiplication is performed over a Galois field $GF(2^8)$ defined by the following primitive polynomial:

$$p_8(x) = x^8 + x^4 + x^3 + x^2 + 1.$$

The last function $\sigma[k]$ is a simple 512-bit XOR operation with the 512-bit value k. In the function $W[H_{i-1}]$ of Fig. 1, k is given by constants c^1 ~ c^{10} for the right-hand 10-round path and is given by keys K^1 ~ K^{10} for the left-hand 10 rounds.

Two different implementations are available for the 8-bit input/output S-box of the function γ: (I) a simple 8-bit input/output lookup table version, and (II) the mini-box structure shown in the leftmost part of Fig. 1. The mini-boxes E, E^{-1}, and R in (II) are implemented using 4-bit input/output lookup tables. For the pipeline architecture described later, the pipeline register is placed between the mini-boxes E, E^{-1}, and the XOR gates in (II) in order to partition the critical path of $\rho[k]$ within the S-box.

3 Compact Hardware Architectures

3.1 Data Manager

The data manager is a circuit component for the function π, which performs byte-wise permutation using a series of shift registers and selectors. Reference [8] used a simple data manager that processes one 8-byte row of the 8×8-byte matrix every cycle. In order to increase throughput using a compact circuit, we extend the data manager to 128-bit and 256-bit datapath widths, which have four and two pipeline stages, respectively.

Let us denote the byte permutation of the function π as shown in Fig. 2, and its straightforward implementations are shown in Fig. 3. The function block π is implemented by metal wire interconnection and no transistor device is used, although a large number of selectors and long data buses in the feedback path have an adverse impact on hardware performance with respect to size and speed. In order to settle this problem, the four-stage (128-bit datapath) and two-stage (256-bit datapath) data managers shown in Figs. 4 and 5, respectively, are introduced herein. In these figures, the square boxes denote 8-bit registers. The four-stage data manager receives 16 bytes

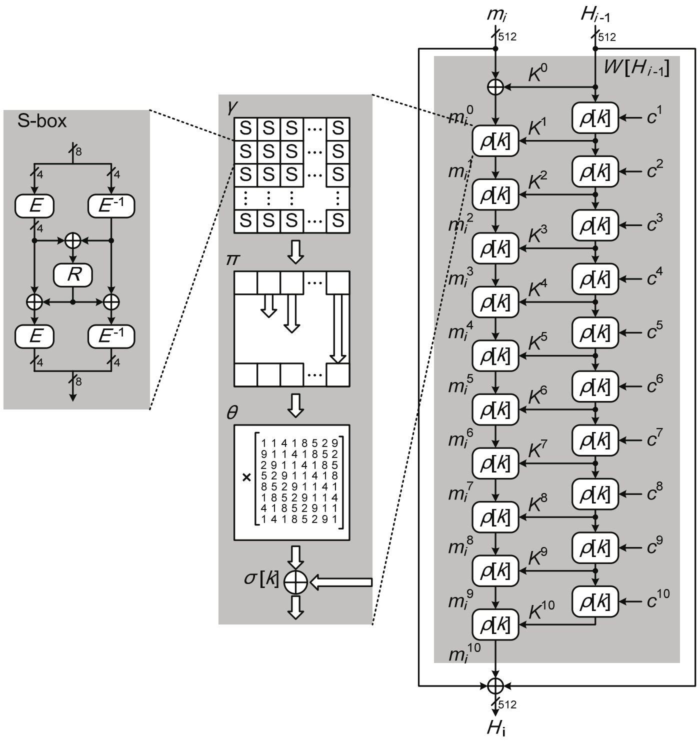


Fig. 1. Whirlpool algorithm

$$\begin{bmatrix} 0 & 8 & 16 & 24 & 32 & 40 & 48 & 56 \\ 1 & 9 & 17 & 25 & 33 & 41 & 49 & 57 \\ 2 & 10 & 18 & 26 & 34 & 42 & 50 & 58 \\ 3 & 11 & 19 & 27 & 35 & 43 & 51 & 59 \\ 4 & 12 & 20 & 28 & 36 & 44 & 52 & 60 \\ 5 & 13 & 21 & 29 & 37 & 45 & 53 & 61 \\ 6 & 14 & 22 & 30 & 38 & 46 & 54 & 62 \\ 7 & 15 & 23 & 31 & 39 & 47 & 55 & 63 \end{bmatrix} \rightarrow \boxed{\pi} \rightarrow \begin{bmatrix} 0 & 15 & 22 & 29 & 36 & 43 & 50 & 57 \\ 1 & 8 & 23 & 30 & 37 & 44 & 51 & 58 \\ 2 & 9 & 16 & 31 & 38 & 45 & 52 & 59 \\ 3 & 10 & 17 & 24 & 39 & 46 & 53 & 60 \\ 4 & 11 & 18 & 25 & 32 & 47 & 54 & 61 \\ 5 & 12 & 19 & 26 & 33 & 40 & 55 & 62 \\ 6 & 13 & 20 & 27 & 34 & 41 & 48 & 63 \\ 7 & 14 & 21 & 28 & 35 & 42 & 49 & 56 \end{bmatrix}$$

Fig. 2. Input/output relationship of the function π

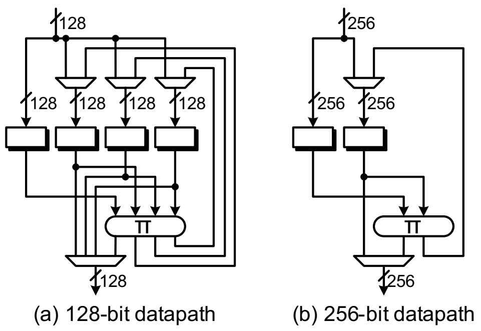


Fig. 3. Straightforward implementations for data managers using 128- and 256-bit datapaths

(128 bits) corresponding to the two rows of the left-hand matrix in Fig. 2 and outputs 16 bytes for the two rows of the right-hand matrix during every cycle. For example, the first 16 input bytes for the data manager are {0, 1, 8, 9, 16, 17, 24, 25, 32, 33, 40, 41, 48, 49, 56, 57}, and the 16 bytes {0, 1, 15, 8, 22, 23, 29, 30, 36, 37, 43, 44, 50, 51, 57, 58} are output after four clock cycles by controlling registers and selectors. The proposed data manager uses only 12 bytes (96 bits) of two-way selectors, whereas the straightforward implementation in Fig. 3(a) requires 96 bytes (768 bits) of equivalent two-way selectors. By reducing the number of registers, the critical path is shortened, and thus, the new data manager provides a smaller circuit area with a higher operating frequency in comparison to the conventional schemes. The two-stage data manager processes four rows of Fig. 2 simultaneously using 16-byte (128-bit) selectors, while the straightforward implementation in Fig. 3(b) requires four times the number of selectors.

3.2 Datapath Architectures

The compact Whirlpool hardware architectures using the new data managers are described in this Section. The 128-bit interleave and pipeline architectures based on the four-stage data manager are described in the following subsections. Other architectures for the two- and eight-stage data managers with a 256- and 64-bit datapath-width, respectively, are depicted in Appendix (Figs. 11~13). The structure of the interleave architectures for the three data managers are all the same except for the size

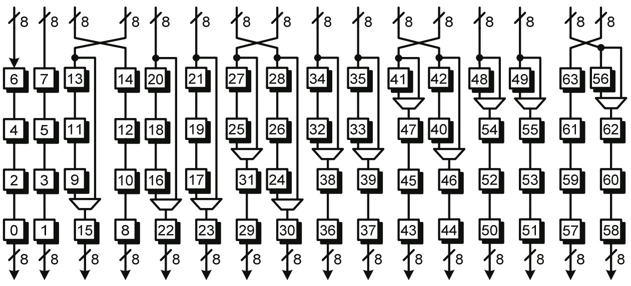


Fig. 4. Four-stage data manager

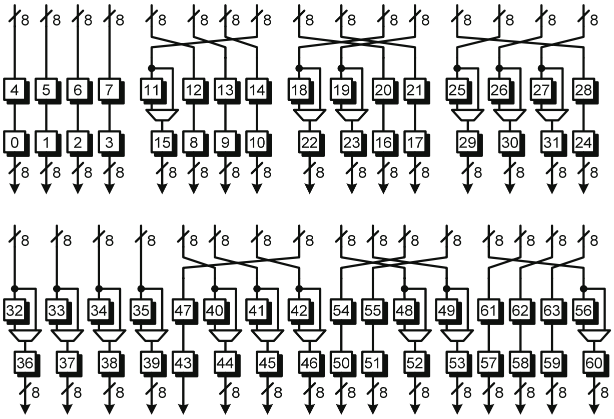


Fig. 5. Two-stage data manager

of operating units. In contrast, the pipeline architectures of Figs. 8, 12, and 13 have different numbers of pipeline stages, because the number of operating cycles varies with the datapath width.

3.2.1 Interleave Architecture

The datapath of the 128-bit interleave architecture are shown in Fig. 6. Two four-stage data managers are used for the architecture. A 512-bit message block or a 512-bit key are stored to a 512-bit shift register in each data manager using four clock cycles. There is only one $\gamma\theta$-function block, and it is used alternatively for data randomization and key scheduling every four clock cycles. In the architecture, the order of the substitution function γ and the permutation function π of the data manager are reversed from the original order in Fig. 1 so that the datapath and sequencer logic are simplified. This has no effect on the operations in data randomization, but the data manager for the key scheduling performs the function π before the key is used in $\sigma[k]$, and thus the inverse function π^{-1} should be applied to the key for adjustment. Fortunately, the function π^{-1} is implemented as rewiring in an ASIC chip in the same

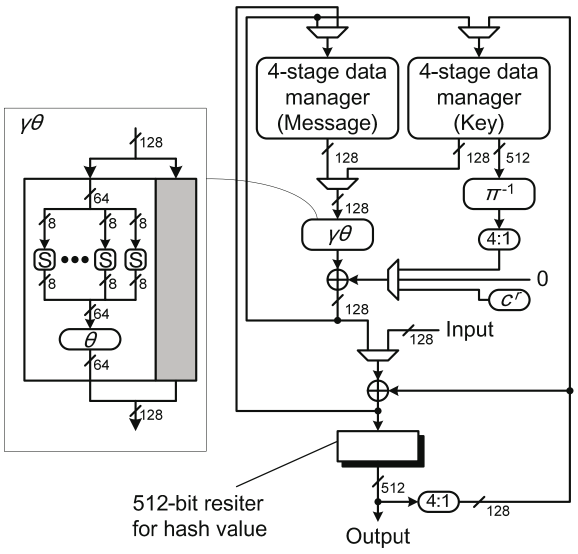

γθ
128
64
8
S
θ
4-stage data manager (Message)
4-stage data manager (Key)
512
π-1
4:1
0
c^r
Input
512-bit resiter for hash value
Output

Fig. 6. 128-bit interleave architecture

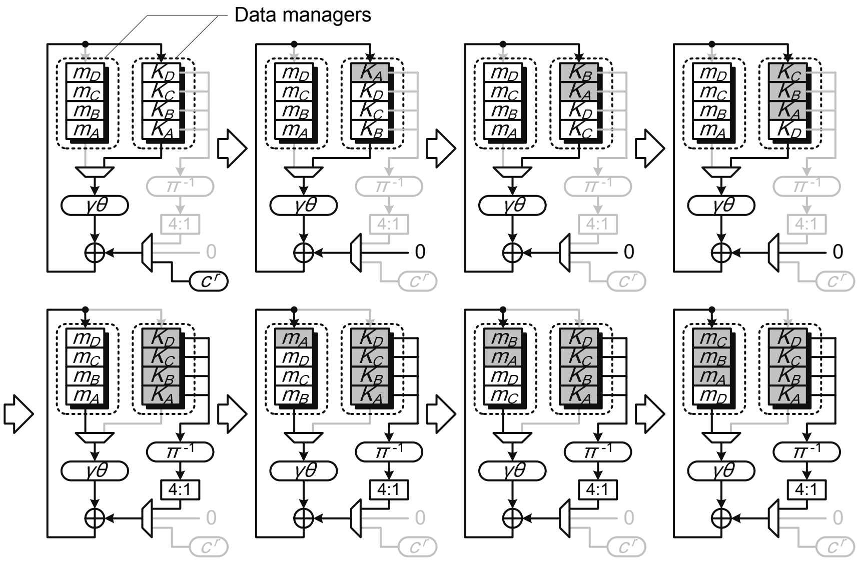


Fig. 7. Example operation of the 128-bit interleave architecture

manner as the function π and no additional selectors are required in this case. Therefore, the function π^{-1} has no impact on hardware resources. Fig. 7 shows an example operation of the architecture. The figure depicts the architecture processing two $\rho[k]$

functions taking eight cycles. Therefore, four cycles are required to perform the function $\rho[k]$, and thus eight cycles are required for each round of the compression function W. The function W uses 10 rounds to process one 512-bit message block, and an additional four clock cycles is needed for data I/O. As a result, the interleave architecture with the 128-bit datapath and the four-stage data manager requires 84 (= 4 × 2 × 10 + 4) cycles for each message block. In a similar manner, the interleave architecture with 256- and 64-bit datapaths (two- and eight-stage data managers) in Figs. 10 require 42 (= 2 × 2 × 10 + 2) and 168 (= 8 × 2 × 10 + 8) clock cycles, respectively.

3.2.2 Pipeline Architecture

Pipeline architecture divides the functions γ and θ by inserting pipeline registers to shorten the critical paths and improve the operation frequency. In the 128-bit datapath architecture, the number of pipeline stages can be increased up to five without causing pipeline stall. The upper limits of the number of pipeline stages are three and nine for the 256-bit and 64-bit versions, respectively. The maximum numbers of stages are used for the performance comparison in the next section.

The datapath and operation of the 128-bit pipeline architecture are shown in Figs. 8 and 9, respectively. The functions γ and θ are partitioned into four sub-blocks as stages 0 ~ 3, and the XOR gates for key addition $\sigma[k]$ followed by selectors

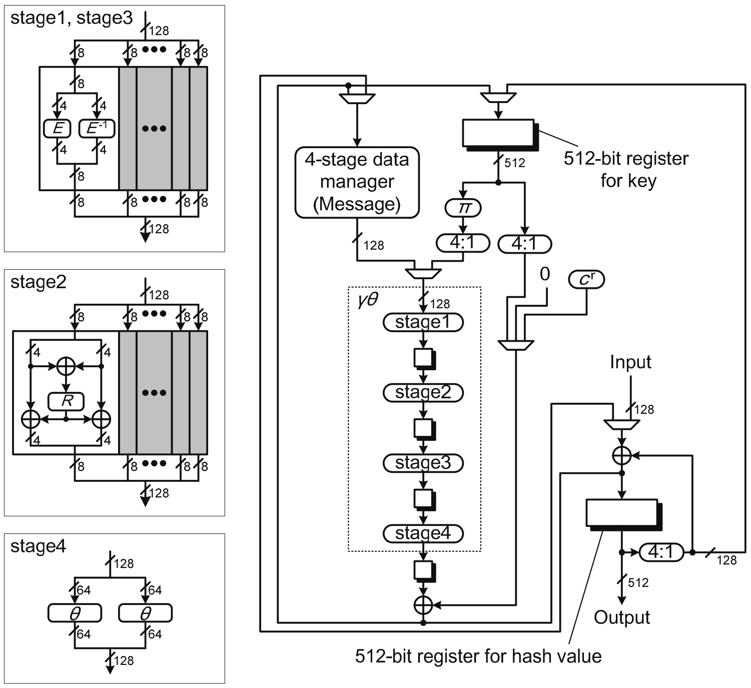


Fig. 8. 128-bit pipeline architecture

correspond to the final stage (stage 5). The partitioned sub-blocks perform message randomization and key scheduling simultaneously, as shown in Fig. 9. The data manager is only used for the message randomization, and the key scheduler uses the 512-bit register with a 4:1 selector. This is because the dependency between key bytes cannot be satisfied, even when using the function π^{-1} as in the interleave architecture. This 128-bit architecture performs two $\rho[k]$ operations in eight cycles and thus requires 80 cycles for the function W, which is the same as the 128-bit interleave architecture. In addition to the 80 cycles, four cycles for data I/O and other four cycles to empty the series of pipeline registers are required. Therefore, one 512-bit message block is compressed in 88 (= 80 + 4 + 4) cycles. Similarly, the 256- and 64-bit versions require 44 (= 40 + 2 + 2) and 176 (= 160 + 8 + 8) cycles, respectively.

4 Performance Evaluation

The proposed Whirlpool architectures were designed in Verilog-HDL and were synthesized by Synopsys Design Compiler (version Y-2006.06-SP2) with the STMicroelectronics 90-nm CMOS standard cell library (1.2-volt version) [11], where two optimization options, size and speed, were specified. Hardware sizes were estimated based on a two-way NAND equivalent gate, and the speeds were evaluated under worst-case conditions. The efficiency is defined as the throughput per gate, and thus higher efficiency indicates better implementation. For performance comparison, the Whirlpool circuits with 512-bit datapath architecture from [5] and the SHA-256/-512 circuits proposed in [10] were also designed and evaluated using the same library.

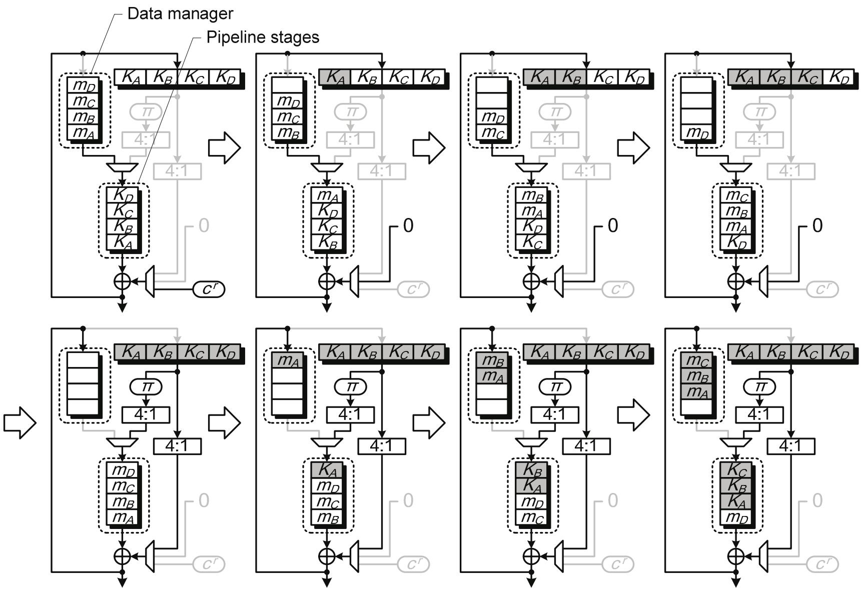


Fig. 9. Example operation of 128-bit pipeline architecture

The synthesis results are shown in Table 1, where the two types of S-box described in Section 2 are referred to as the *GF*(2^8) and *GF*(2^4) tables. Only the *GF*(2^4) table is used for the pipeline architectures so that the pipeline register can be placed in the middle of the S-box. The results are also displayed in Fig. 10 where the horizontal and vertical axes are gate count and throughput. Note that the largest implementation with 179.0 Kgates is outside of this graph. In the figure, the circuit is smaller when the corresponding dot is located in the left region, and is faster when the dot is in the upper region. As a result, implementations plotted at the upper left region of the graph have better efficiency.

The interleave architecture with a wider datapath including the conventional datapath always obtained higher efficiency. This is because throughput is halved if the datapath width is halved, whereas the hardware size cannot be halved due to the constant size of the data registers. In contrast, the 256-bit datapath achieved higher efficiency than 512-bit version for the pipeline architecture. In this case, the proposed

Table 1. Performance comparison in the 90-nm CMOS standard cell library

Design	Algo -rithm	Message Block (bits)	Cycle	S-box	Datapath architecture	Optimize	Area (gates)	Operating Frequency (MHz)	Through-put (Mbps)	Efficiency (Kbps/ gate)
This work	Whirl -pool	512	42	*GF* (2^8) Table	256-bit Interleave	Area	36,201	266.67	3,251	89.80
						Speed	64,796	568.18	6,926	106.90
				GF (2^4) Table		Area	21,495	266.67	3,251	151.23
						Speed	42,972	529.10	6,450	150.10
			84	*GF* (2^8) Table	128-bit Interleave	Area	22,527	269.54	1,643	72.93
						Speed	37,865	571.43	3,483	91.98
				GF (2^4) Table		Area	15,891	268.10	1,634	102.83
						Speed	28,337	537.63	3,277	115.64
			168	*GF* (2^8) Table	64-bit Interleave	Area	16,675	268.10	817	49.00
						Speed	24,029	568.18	1,732	72.06
				GF (2^4) Table		Area	13,614	268.10	817	60.02
						Speed	20,554	546.45	1,665	81.02
			44	*GF* (2^4) Table	256-bit Pipeline	Area	21,395	558.66	6,501	303.84
						Speed	35,520	1,136.36	13,223	372.27
			88		128-bit Pipeline	Area	16,677	574.71	3,344	200.50
						Speed	23,230	1,111.11	6,465	278.29
			176		64-bit Pipeline	Area	14,762	564.97	1,644	111.34
						Speed	19,500	1,041.67	3,030	155.40
[5]	Whirl -pool	512	10	*GF* (2^8) Table	512-bit Parallel	Area	103,633	211.42	10,825	104.45
						Speed	179,035	546.45	27,978	156.27
				GF (2^4) Table		Area	43,726	210.97	10,802	247.03
						Speed	103,408	523.56	26,806	259.23
			21	*GF* (2^8) Table	512-bit Interleave	Area	58,792	210.08	5,122	87.12
						Speed	97,541	518.13	12,633	129.51
				GF (2^4) Table		Area	29,577	210.08	5,122	173.18
						Speed	64,549	500.00	12,190	188.86
				GF (2^8) Table	512-bit Pipeline (A)	Area	60,066	364.96	8,898	148.14
						Speed	77,312	671.14	16,363	211.65
				GF (2^4) Table		Area	31,938	363.64	8,866	277.59
						Speed	40,476	564.97	13,775	340.31
				GF (2^4) Table	512-bit Pipeline (B)	Area	30,105	363.64	8,866	294.50
						Speed	40,330	574.71	14,012	347.43
[10]	SHA -256	512	72			Area	9,764	362.32	2,576	263.88
						Speed	13,624	490.20	3,486	255.86
	SHA -512	1024	88			Area	17,104	209.64	2,439	142.63
						Speed	27,239	404.86	4,711	172.95

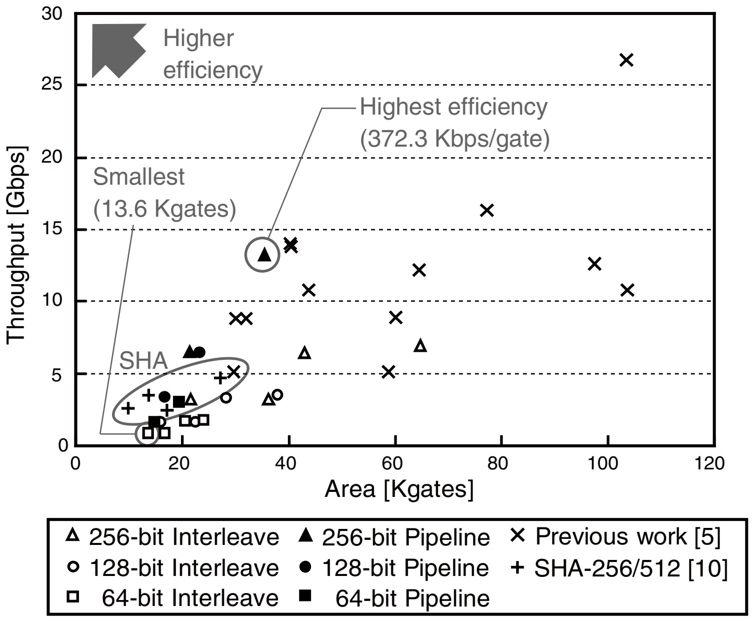


Fig. 10. Throughput versus area for each implementation

efficiency than 512-bit version for the pipeline architecture. In this case, the proposed deep pipeline scheme allows a much higher operating frequency, and consequently the high throughput with the small circuit resulted in a higher efficiency. However, the operating frequencies of the 128-bit (five-stage) and 64-bit (nine-stage) pipeline architectures were not improved compared with that of the 256-bit architecture, even though the number of pipeline stages was increased. The major reason for this is the increasing additional selectors in the critical path from the key register to the data manager through key addition. Therefore, we must consider the hardware resources and the signal delay time caused by additional selectors for optimizing datapath in deep pipeline operation.

The 64-bit interleave architecture in conjunction with the $GF(2^4)$ S-box and the area optimization option achieved the smallest size of 13.6 Kgates with a throughput of 817 Mbps. The circuits using the $GF(2^4)$ S-box are smaller and have higher efficiency than those using the $GF(2^8)$ S-box. The $GF(2^8)$ S-box leads to a large circuit but is still suitable for high-speed implementation. Generally, the interleave architecture is smaller than the pipeline architecture which requires a number of pipeline registers. The smallest circuit based on the conventional scheme is 29.6 Kgates for the 512-bit pipeline architecture with the $GF(2^4)$ S-box. Therefore, the gate count of 13.6 Kgates obtained by the proposed interleave architecture is 54% smaller than that of the conventional scheme. The 256-bit pipeline version optimized for speed achieved the highest efficiency of 372.3 Kbps/gate (= 13.2 Gbps/35.5 Kgates) among the Whirlpool implementations. This means that the pipeline architecture provides the optimal balance between speed and size.

In comparison with the SHA-256 and -512 circuits, the smallest Whirlpool circuit is larger than the area-optimized SHA-256 circuit with 9.8 Kgates but is smaller than the SHA-512 circuit with 17.1 Kgates. The highest throughput of the proposed architectures (13.2 Gbps) is 2.8 times higher than that (4.7 Gbps) of the speed-optimized SHA-512, and the highest efficiency of 372.3 Kbps/gate is 1.4 times higher than 263.8 Kbps/gate of the area-optimized SHA-256. The proposed Whirlpool architectures also achieved a wide variety of size and speed performances, while the SHA-256 and -512 circuits have only four implementations. Consequently, the proposed Whirlpool hardware has great advantages in both performance and flexibility.

5 Conclusion

In the present paper, compact hardware architectures with interleave and pipeline schemes were proposed for the 512-bit hash function Whirlpool, and their performances were evaluated using a 90-nm CMOS standard cell library. The fastest throughput of 13.2 Gbps @ 35.5 Kgates, the smallest circuit area of 13.6 Kgates @ 817 Mbps, and the highest efficiency of 372.3 Kbps/gate were then obtained using the proposed architectures. These results indicate that the proposed architectures can provide higher performance with respect to both size and efficiency, as compared to the conventional 512-bit architectures. In addition to the peak performance in size and speed, the flexibility of the proposed architectures enables various design options to meet a variety of application requirements.

Further research to reduce the overhead of additional selectors in the pipeline architectures is currently being conducted. The method will further improve both size and speed.

References

1. The Whirlpool Hash Function, `http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html`
2. Barreto, P., Rijmen, V.: The Whirlpool Hash Function, `http://planeta.terra.om.br/informatica/paulobarreto/whirlpool.zip`
3. ISO/IEC 10118-3:2004, "Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions, `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39876`
4. NIST, Advanced Encryption Standard (AES) FIPS Publication 197, (November 2001), `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`
5. Satoh, A.: ASIC Hardware Implementations for 512-Bit Hash Function Whirlpool. In: Proceedings ISCAS 2008, pp. 2917–2920 (May 2008)
6. NIST, Secure Hash Standard (SHS), FIPS PUB 180-2 (August 2002), `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`
7. Pramstaller, N., Rechberger, C., Rijmen, V.: A Compact FPGA Implementation of the Hash Function Whirlpool. In: Proceedings of the 2006 ACM/SIGDA, pp. 159–166 (2006)
8. McLoone, M., McIvor, C.: High-speed & Low Area Hardware Architectures of the Whirlpool Hash Function. J. VLSI Signal Processing 47(1), 47–57 (2007)

9. Alho, T., Hämäläinen, P., Hännikäinen, M., Hämäläinen, T.: Compact Hardware Design of Whirlpool Hashing Core. In: Proceedings of the DATE 2007, pp. 1247–1252 (April 2007)
10. Satoh, A., Inoue, T.: ASIC-hardware-focused comparison for hash functions MD5, RIPEMD-160, and SHS. Integration, the VLSI Journal 40(1), 3–10 (2007)
11. Circuits Multi-Projets (CMP), CMOS 90 nm (CMOS090) from STMicroelectronics, `http://cmp.imag.fr/products/ic/?p=STCMOS090`

Appendix

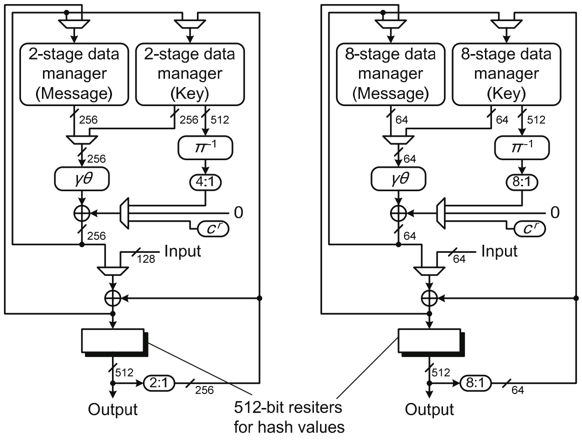

2-stage data manager (Message)
2-stage data manager (Key)
8-stage data manager (Message)
8-stage data manager (Key)
256
512
64
π^{-1}
$\gamma\theta$
4:1
8:1
0
c^r
128
Input
2:1
Output
512-bit resiters for hash values

Fig. 11. 256-bit (left) and 64-bit (right) interleave architectures

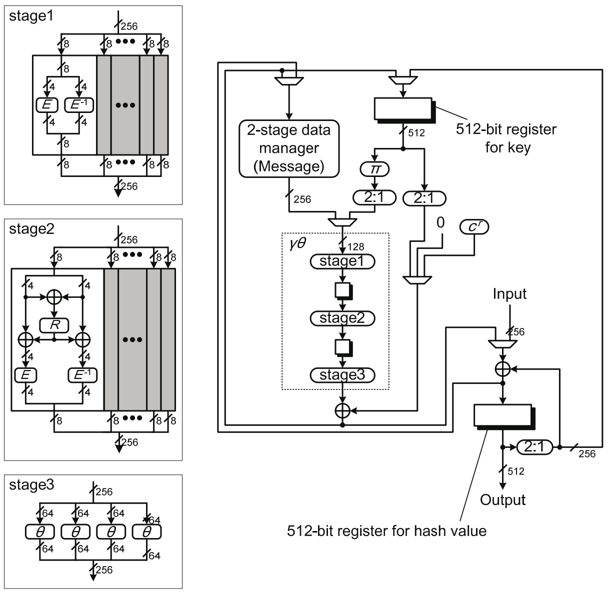


Fig. 12. 256-bit pipeline architecture

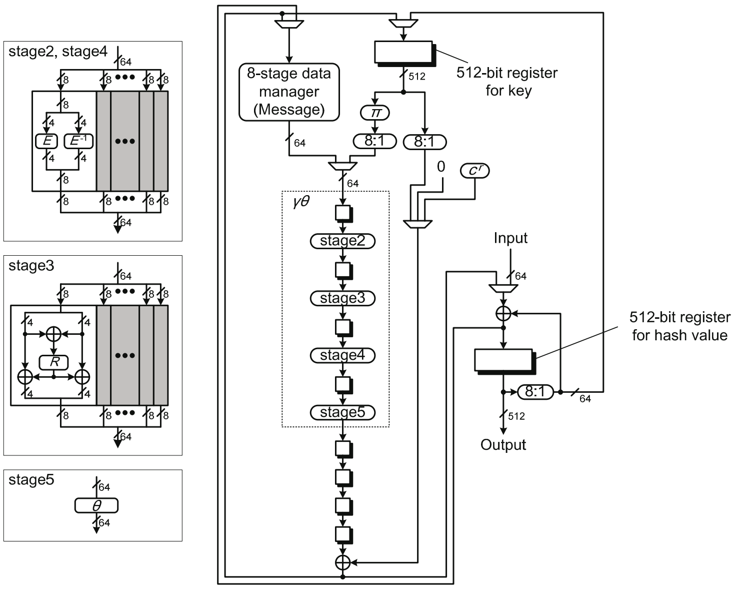


Fig. 13. 64-bit pipeline architecture

Improved Constant Storage Self-healing Key Distribution with Revocation in Wireless Sensor Network

Qingyu Xu and Mingxing He

School of Mathematics and Computer Engineering, Xihua University, Chengdu, Sichuan, 610039, China
xqu1002@163.com, he_mingxing64@yahoo.com.cn

Abstract. Recently, Dutta et al. [1] proposes a constant storage self-healing key distribution with revocation in wireless sensor network. The advantage of [1] is that it requires constant storage of personal keys for each user. In this paper, we show that Dutta's scheme is insecure against the proposed attack, whereby users can access any other's secret keys and session keys which they should not know according to Dutta's scheme. Moreover, we proposed two kinds of improved schemes which are resistant to this kind of attack. The first scheme is unconditional secure and we analyze it in the security framework of [1]. The second scheme is in the model of computational secure. In the end, we analyze the second scheme and show that it is a self-healing key distribution scheme with revocation and achieves both forward and backward secrecy.

Keywords: key distribution self-healing, key revocation, storage complexity, unconditional secure, computational secure, wireless sensor network.

1 Introduction

Recently, a kind of so-called "self-healing" key distribution schemes with revocation capability have been proposed by Staddon et al. [2]. The schemes enable a dynamic group of users to establish a group key over an unreliable network, and have the ability to revoke users from and add users to the group while being resistant to collusion attack. In these schemes users who have missed up to a certain number of previous rekeying operations can recover the missing group keys without requesting additional transmission from the group manager. The only requirement is that the user must be a group member both before and after the session.

Wireless sensor network consists of a large number of small, low cost sensor nodes which have limited computing and energy resources. Wireless sensor network have wide applications in military operations and scientific explorations, where there is not network infrastructure to support and the adversary may intercept, modify, or partially interrupt the communication. In such applications, security becomes a critical concern. Self-healing key distribution is a potential candidate to establish session keys for secure communication to large groups in wireless sensor network, where frequent membership changes may be necessary and the ability to revoke users is desirable. Moreover, in such situations the session keys need to be used for a short time-period or need to be updated frequently. Therefore, self-healing is a good property for key distribution in wireless sensor network.

Self-healing key distribution with revocation was first introduced by Staddon et al. in [2]. They provided formal definitions and security notions that were later generalized by Liu et al. [3] and Blundo et al. [4]. The constructions given in [2] suffered from high storage and communication overhead. Liu et al. [3] introduced a novel personal key distribution scheme by combining the key distribution scheme with the self-healing technique in [2]. They proposed a new construction that improved the storage and communication overhead greatly. [9] introduced the concept of sliding window. A window determines the range (the sessions) that the self-healing can apply to. Only the previous and subsequent sessions which are in the window can be used to recover the lost session keys. Blundo et al. [4] proposed a new self-healing technique different from that given in [1] under a slightly modified setting. More recently, Hong et al. [5] proposed self-healing key distribution constructions with less storage and communication complexity. B. T and M. He [6] designed a constant storage scheme which enabled users selecting their personal keys by themselves instead of being distributed by group manager, and the users could reuse their personal keys and conceal the requirement of a secure channel in setup step. [11] have given some lower bounds on the user memory storage and communication complexity required for implementing such schemes, and shown that they are tight by describing simple constructions. They have also shown that the lower bounds cannot be attained at the same time. [12] consider the application environment that a coalition of users sponsor a user outside the group for one session. Dutta et al. [13] apply one-way key chain to their constructions. The schemes greatly reduce communication complexity. The schemes are scalable to very large groups in highly mobile, volatile and hostile network. Dutta et al. [1] [7] designed the schemes which were more efficient in the storage and the communication overhead, since the communication complexity of Dutta's scheme was independent of the size of the group, instead that they depended on the number of compromised group members that may collude together. Dutta's scheme was not restricted to m sessions in setup phase. However, we found out some weaknesses in the two schemes of [1] and [7].

Our Contribution: First, we point out the weakness of Dutta's scheme in [1]. Second, we slightly modify the Dutta's scheme so that the improved scheme is resistant to the proposed attack. The proposed scheme is also unconditional secure. Third, we design another scheme which is more efficient in communication complexity. In the application environment requiring strong resistance to revoker's collusion attack or revoked users being more than the active users, the advantage of second scheme is obvious. The second scheme is in the model of computational secure.

2 Overview of Dutta's Scheme

The following notations are used throughout our paper.
$\mathcal{U}$: set of all users in the networks;
$\mathcal{U}_{act_j}$: set of active users in *j-th* session;
U_i: *i-th* user in $\mathcal{U}$;
U_{act_i}: *i-th* user in $\mathcal{U}_{act_j}$;
$\mathcal{R}_{act_j}$, the set of all active users' secret values in *j-th* session;
r_{act_i}: *i-th* user's secret value in $\mathcal{R}_{act_j}$;
GM: group manager;
n: the number of users in $\mathcal{U}$;
t: the maximum number of revoked user;
a_j: the number of users in $\mathcal{U}_{act_j}$;
m: total number of sessions;
F_q: a field of order q;
S_i: set of personal secrets of user U_i;
$s_{i,j}$: personal secret of user U_i in *j-th* session;
K_j: session key generated by the GM in *j-th* session;
B_j: broadcast message by the GM in *j-th* session;
H: entropy function of information theory;
$Z_{i,j}$: the information learned by U_i through B_i and S_i;
L_j: set of all revoked users before and in *j-th* session;
L: any set of some users;
J: set of users joining in;
$A_j(x)$: access polynomial in *j-th* session;
$\Lambda_j(x)$: revocation polynomial in *j-th* session;
$\phi_i(x)$: broadcast polynomial in *j-th* session;
$\psi(x)$:personal secret polynomial;
f: a random one way permutation;
h_{td}: secure one way trapdoor function;
f_1: secure one way function.

2.1 Security Model of Dutta's Scheme

Definition 1. (Session Key Distribution with b-bit privacy) Let $t, i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.
1.1) D is session key distribution scheme with b-bit privacy if

(1.1.a) For any user U_i, the session key K_j is determined by $Z_{i,j}$, which in turn is determined by B_j and S_i. i.e.

$$H(K_j|Z_{i,j}) = 0, H(Z_{i,j}|B_j, S_i) = 0 \tag{1}$$

(1.1.b) For any $L \subseteq U$, $|L| \leq t$, and $U_i \notin L$, the uncertainty of the users in L to determining S_i is at least b bits($b > 0$). i.e.

$$H(S_i|\{S_l\}_{U_l \in L}, B_1, \ldots, B_m) \geq b \tag{2}$$

(1.1.c) What users $U_1, \ldots, U_n$ learn from B_j cannot be determined from broadcasts or personal keys alone. i.e.

$$H(Z_{i,j}|B_1, \ldots, B_m) = H(Z_{i,j}) = H(Z_{i,j}|S_1, \ldots, S_n) \tag{3}$$

1.2) D has t-revocation capability if given any $L \subseteq U$, where $|L| \leq t$, the group manager GM can generate a broadcast B_j, such that for all $U_i \notin L$, U_i can recover K_j, but the revoked users cannot. i.e.

$$H(K_j|B_j, S_i) = 0, H(K_j|B_j, \{S_l\}_{U_l \in L}) = H(K_j) \tag{4}$$

1.3) D is self-healing if the following is true for any j, $1 \leq j_1 < j < j_2 \leq m$: For any user U_i who is a valid member in session j_1 and j_2, the key K_j is determined by the set $\{Z_{i,j_1}, Z_{i,j_2}\}$. i.e.

$$H(K_j|Z_{i,j_1}, Z_{i,j_2}) = 0 \tag{5}$$

Definition 2. (t-wise forward and backward secrecy)
Let $t, i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.
2.1) A key distribution scheme D guarantees t-wise forward secrecy if for any set $L \subseteq \mathcal{U}(|L| \leq t)$, and all $U_l \in L$ are revoked before session j, the members in L together cannot get any information about K_j, even with the knowledge of group keys before session j. i.e.

$$H(K_j|B_1, \ldots, B_m, \{S_l\}_{U_l \in L}, K_1, \ldots, K_{j-1}) = H(K_j) \tag{6}$$

2.2) A session key distribution D guarantees t-wise backward secrecy if for any set $J \subseteq \mathcal{U}(|J| \leq t)$, and all $U_l \in J$ join after session j, the members in J together cannot get any information about K_j, even with the knowledge of group keys after session j. i.e.

$$H(K_j|B_1, \ldots, B_m, \{S_l\}_{U_l \in L}, K_{j+1}, \ldots, K_m) = H(K_j) \tag{7}$$

2.2 Protocol Requirements

Assume that there is a set which includes a group manager GM and n (fixed) users $\mathcal{U} = \{U_1, \ldots, U_n\}$. All of the operations in the protocol take place in a finite field F_q, where q is a large prime number ($q > n$). Let E_K, D_K denote respectively an encryption and corresponding decryption function under key $K \in F_q$. It also takes a random one way permutation f over F_q such that $f^i(u) \neq f^j(u)$ for all positive integers i, $j(i \neq j)$ and $u \in F_q(f^i$ means the permutation f is applied i times).

2.3 Self-healing Session Key Distribution

Setup: Let t be a positive integer. The group manager GM randomly chooses a t-degree polynomial $\psi(x) = a_0 + a_1x + a_2x^2 + \cdots + a_tx^t$ in $F_q[x]$ and a random initial session identifier $sid_0 \in F_q$. Each user U_i, for $1 \leq i \leq n$, stores $s_{i,0} = \{\psi(i), sid_0\}$ as its personal secret key. The personal secret key is received from the GM via the secure communication channel between GM and user. The GM randomly chooses a prime key $K_0 \in F_q$ that is kept secret to itself.

Broadcast: In the j-th session key distribution $j \geq 1$, the GM computes its j-th session identifier $sid_j = f(sid_{j-1})$. sid_0 is stored by users. For $j > 1$, it replaces the previous session identifier sid_{j-1} by the current value sid_j. Then GM chooses a random number $\beta_j \in F_q$ and computes the j-th session key $K_j = E_{\beta_j}(K_{j-1})$. β_j is as self-healing key of session j. Let $L_j = \{U_{l_1}, \ldots, U_{l_{w_j}}\} \subseteq \mathcal{U}(|L_j| = w_j \leq t)$ be the set of all revoked users for sessions in and before j. The GM broadcasts the following message:

$$B_j = L_j \cup \{\phi_j(x) = \Lambda_j(x)K_j + sid_j\psi(x)\} \cup \{E_{K_j}(\beta_1), \ldots, E_{K_j}(\beta_j)\}, \tag{8}$$

where $\Lambda_j(x) = (x - l_1)(x - l_2), \ldots, (x - l_{w_l})$. Here the polynomial $\Lambda_j(x)$ is called revocation polynomial and $\psi(x)$ performs the role of masking polynomial. Each user $U_i \in \mathcal{U}$ knows a single point, namely $(i, \psi(i))$ on the polynomial $\psi(x)$.

Session Key and Self-healing Key Recovery: When a non-revoked user U_i receives the j-th broadcast message B_j, it first computes the session identifier $sid_j = f(sid_{j-1})$ and replaces the previous session identifier sid_{j-1} by the current value sid_j for $j > 1$(in case $j = 1$, sid_0 is stored). Then U_i evaluates $\psi(i)$, $\phi_j(i)$ and $\Lambda_j(i)$. Finally, U_i computes the current session key

$$K_j = \frac{\phi_j(i) - sid_j\psi(i)}{\Lambda_j(i)}. \tag{9}$$

Note that from the set L_j in the broadcast message B_j, all users U_i can construct the polynomial $\Lambda_j(x)$, consequently, can evaluate the value $\Lambda_j(i)$. In particular, for $U_i \in L_j$, $\Lambda_j(i) = 0$.

Once a non-revoked user U_i recovers the current session key K_j, it can recover the self-healing keys $\beta_1, \ldots, \beta_j$. These self-healing keys enables the scheme to have self-healing capability as will be shown later.

We now explain self-healing mechanism: Let U_i be a group member that receives session key distribution messages B_{j_1} and B_{j_2} in sessions j_1 and j_2 respectively, where $1 \leq j_1 \leq j_2$, but not the session key distribution message B_j for session j, where $j_1 < j < j_2$. User U_i can still recover all the lost session keys K_j for $j_1 < j < j_2$ as follows:

(a) U_i first computes $sid_{j_1} = f^{j_1}(sid_0)$ and $\phi_{j_1}(i)$, $\Lambda_{j_1}(i)$ from the broadcast message B_{j_1} in session j_1. Then U_i recovers the j_1-th session key $K_{j_1} = \frac{\phi_{j_1}(i) - sid_{j_1}\psi(i)}{\Lambda_{j_1}(i)}$.

(b) U_i computes $sid_{j_2} = f^{j_2}(sid_0)$ and $\phi_{j_2}(i)$, $\Lambda_{j_2}(i)$ from the broadcast message B_{j_2} in session j_2. Then U_i recovers the j_2-th session key $K_{j_2} = \frac{\phi_{j_2}(i) - sid_{j_2}\psi(i)}{\Lambda_{j_2}(i)}$.

(c) U_i decrypts $E_{K_{j_2}}(\beta_1), \ldots, E_{K_{j_2}}(\beta_{j_2})$ using K_{j_2} to obtain the self-healing keys $\beta_1, \ldots, \beta_{j_1}, \beta_{j_1+1} \ldots \beta_{j_2}$.

(d) U_i then computes K_j for all $j = j_1+1, j_1+2, \ldots, j_2-1$ as: $K_j = E_{\beta_j}(K_{j-1})$.

3 Attack To Dutta's Scheme

In this section, we show how to attack Dutta's scheme and access to any other's secret keys and session keys which they should not know according to Dutta's scheme.
1) Suppose that there are two users U_A and U_B whose identity is A and B respectively, the secret key of U_A and U_B is $\psi(A)$ and $\psi(B)$ respectively. User U_B is not revoked in all of sessions. User U_A is not revoked until $(j+1)$-*th* session. So U_A can access to the j-*th* session key K_j but can not access the $(j+1)$-*th* session key K_{j+1} according to Dutta's scheme. Next, we show how U_A can access to the session key K_{j+1} and U_B's personal secret key $\psi(B)$.
2) U_A obtains broadcast polynomial ϕ_j and constructs revocation polynomial $\Lambda_j(x)$ according to the set of revoked users.
3) U_A also obtains the j-*th* session key K_j, since he is not revoked in j-*th* session. And U_A can compute the value of sid_j. Finally, since U_A knows ϕ_j, $\Lambda_j(x)$, sid_j and K_j, U_A can compute the polynomial

$$\psi(x) = \frac{\phi_j(x) - \Lambda_j(x)K_j}{sid_j}, \tag{10}$$

where $\psi(x)$ is used to compute the value of secret key.
4) Now, U_A evaluates the user U_B's secret key $\psi(B)$ by using user U_B's identity B and $\psi(x)$. Moreover, user U_A can access to any session key K_x by $\psi(B)$(x denotes any x-*th* session, user U_B is active in any session)

$$K_x = \frac{\phi_x(B) - sid_x\psi(B)}{\Lambda_x(B)}. \tag{11}$$

In the same way, U_A can compute any user's secret key. Moreover U_A can produce new secrets key by himself according to Dutta's scheme, because U_A knows the personal secret polynomial $\psi(x)$. Obviously, the above attack can be easily carried out.

To sum up, we point out that Dutta's scheme does not satisfy the requirement of $Definition 1.1.b$. For any set L and set $L_j \subseteq L_{j+1} \subseteq U$, $|L_j| \le |L_{j+1}| \le t$ and revoked user $U_l \in L_{j+1}$, $U_l \notin L_j$, the revoked user U_l can obtain any active user U_i's personal secret $S_i = \{s_{i,j}\} = \{\psi(i), sid_j\}$ according to the attack. Hence we have

$$\begin{aligned} &H(S_i|\{S_l\}_{U_l\in L}, B_1, \dots, B_m) \\ &= H(\{s_{i,j}\}|\{S_l\}_{U_l\in L}, B_1, \dots, B_m) \\ &= H(\{\psi(i), sid_j\}|\{\psi(l)\}_{U_l\in L}, B_1, \dots, B_m) \\ &= H(\{\psi(i)\}|\{\psi(l)\}_{U_l\in L}, B_1, \dots, B_m) \\ &\le H(\{\psi(i)\}|\{\psi(l)\}_{U_l\in L_{j+1}, U_l\notin L_j}, B_1, \dots, B_m) = 0 \end{aligned} \tag{12}$$

4 The Proposed Scheme 1

4.1 Security Model of the Proposed Scheme

We adopt the security model of $section 2.1$([1]).

4.2 The Proposed Scheme 1

We adopt the most of symbols which are similar to Dutta's scheme in order to compare with the schemes. Unlike Dutta's scheme, we make use of access polynomial [8] instead of revocation polynomial. In our setting, we use the notation $\mathcal{U}_{act_j}$ for denoting the set of active users in *j-th* session. Active user $U_{act_i} \in \mathcal{U}_{act_j}$ can access the *j-th* session key. The following steps, which slightly modify the Dutta's scheme, construct a scheme which is resistant to the attack described in *section*3. The scheme is never allowed that a revoked user to rejoin the group in a later session.

1) Initialization: Assume that there are a group manager GM and n users $\mathcal{U} = \{U_1, \ldots, U_n\}$. All of the operations take place in a finite field F_q, where q is a sufficiently large prime number. Let E_K, D_K denotes respectively an encryption and corresponding decryption function under key $K \in F_q$. f_1: $F_q \rightarrow F_q$ be cryptographically secure one-way function. h_{td}: $F_q \rightarrow F_q$ is cryptographically secure one-way trapdoor function where td is the trapdoor.

2) Setup: Let t be a positive integer. The GM chooses randomly a t-degree polynomial $\psi(x) = a_0 + a_1x + a_2x^2 + \cdots + a_tx^t$ in $F_q[x]$ and a random initial session identifier $sid_0 \in F_q$. The GM also selects random secret value $r_i \in F_q$ and computes the secret key $\psi(r_i)$ for user U_i. sid_0 and r_i should be different from each other. Each user $U_i (1 \leq i \leq n)$ stores secret value $s_{i,0} = \{r_i, \psi(r_i), sid_0\}$ as its personal secret received from the GM via the secure communication channel between them. The GM chooses randomly a prime key $K_0 \in F_q$ that is kept secret.

3) Broadcast: In the *j-th* session key distribution, $j \geq 1$, the GM computes its session identifier $sid_j = f_1(sid_{j-1})$. Note that initial value sid_0 as secret value is stored by user all the time. For $j > 1$, it replaces the previous session identifier sid_{j-1} by the current value sid_j. The GM chooses a random number $\beta_j \in F_q$ and computes the *j-th* session key $K_j = h_{\beta_j}(K_{j-1})$. β_j is as self-healing key of session j. Let $\mathcal{U}_{act_j} = \{U_{act_1}, \ldots, U_{act_{a_j}}\}$, $|\mathcal{U}_{act_j}| = a_j$, be the set of all active users for *j-th* session, where a_j is the number of active user in session j. Let $\mathcal{R}_{act_j} = \{r_{act_1}, \ldots, r_{act_{a_j}}\}$, $|\mathcal{R}_{act_j}| = a_j$, be the set of all active users' secret values for *j-th* session. The GM broadcasts the following message:

$$B_j = \{\phi_j(x) = A_j(x)K_j + sid_j\psi(x)\} \cup \{E_{K_j}(\beta_1), \ldots, E_{K_j}(\beta_j)\}, \tag{13}$$

where $A_j(x) = (x - BV_j)\prod_{i=1}^{a_j}(x - r_{act_i}) + 1$ is an access polynomial, using a_j active users' secret value r_{act_i}s. $(x - BV_j)$ is a blind value term and BV_j is different from all users' secret value r_i and randomly selected for each $A_j(x)$. $(x - BV_j)$ is used to make $A_j(x)$s different even they contain the same secret value r_{act_i} of active users. When an active user U_{act_i} receives the *j-th* broadcast message B_j, U_{act_i} can evaluate $A_j(r_{act_i})$ by using its secret value r_{act_i}, as $A_j(r_{act_i}) = 1$. However, for a revoked user $U_l \in L_j$, $A_j(r_l)$ is a random value. Note that none of $A_j(x)$ is sent to user and all users can not construct $A_j(x)$.

$\psi(x)$ is secret key polynomial. Each user U_i knows a single point $\{r_i, \psi(r_i)\}$ on the curve of polynomial $\psi(x)$.

4) Session Key and Self-healing Key Recovery: When an active user U_i receives the *j-th* broadcast message B_j, it first computes the session identifier $sid_j = f_1(sid_{j-1})$ and replaces the previous session identifier sid_{j-1} by the current value sid_j for $j > 1$(sid_0 is stored). Then U_i evaluates $\psi(r_i)$ and $\phi_j(r_i)$. Finally, according to $equation(13)$, U_i can compute the current session key

$$K_j = \phi_j(r_i) - sid_j\psi(r_i), \tag{14}$$

where $A_j(r_i) = 1$.

Now we describe our self-healing mechanism in this construction: Let U_i be an active user that receives session key distribution message B_{j_1} and B_{j_2} in session j_1 and $j_2(1 \leq j_1 < j_2)$ respectively. User U_i can still recover all the lost session keys K_j for $j_1 < j < j_2$ as follows:
At first, U_i computes $sid_{j_1} = f_1^{j_1}(sid_0)$ and $\phi_{j_1}(r_i)$ from the broadcast message B_{j_1} in session j_1 and U_i can recover the j_1-*th* session key $K_{j_1} = \phi_{j_1}(r_i) - sid_{j_1}\psi(r_i)$ according to $equation(14)$. Then U_i computes $sid_{j_2} = f^{j_2}(sid_0)$ and $\phi_{j_2}(r_i)$ from the broadcast message B_{j_2} in session j_2. Now U_i can recover the j_2-*th* session key $K_{j_2} = \phi_{j_2}(r_i) - sid_{j_2}\psi(r_i)$.

After that U_i decrypts $E_{K_{j_2}}(\beta_1), \ldots, E_{K_{j_2}}(\beta_{j_2})$ using K_{j_2} to obtain the self-healing keys $\beta_1, \ldots, \beta_{j_1}, \beta_{j_1+1}, \ldots, \beta_{j_2}$.
Finally, U_i can compute any session key K_j between session j_1 and session j_2 as $K_j = h_{\beta_j}(K_{j-1})$.

5) Add New users: When the GM adds a new user U_v started from session j, it just chooses an unique secret value $r_v \in F_q$, computes the personal secret key $\psi(r_v)$ and sid_j, and gives secret value $s_{v,j} = \{r_v, \psi(r_v), sid_j\}$ to U_v through the secure communication channel between them.

4.3 Security Analysis of Scheme 1

In this section, we prove that our scheme is resistant to the attack described in $section3$. Moreover, our scheme is a self-healing key distribution scheme with revocation capability. More precisely, we can prove the following results.

Result 1: $scheme1$ is resistant to the attack in $section3$.

Proof: We have the same assumption as the attack in $section3$ that user U_A is active in session j but revoked in session $j+1$ and user U_B is an active user in all of sessions. According to the proposed scheme, U_A can obtain the broadcast polynomial $\phi_j(x)$ and session identifier sid_j. Then U_A can evaluate the session key K_j according to $equation14$, since U_A is active in session j then U_A can get the value of polynomial $A_j(x)$ at $x = r_A$, i.e., $A_j(r_A) = 1$. After that, if A wants to compute the secret key polynomial $\psi(x)$ according to

$$\psi(x) = \frac{\phi_j(x) - A_j(x)K_j}{sid_j}, \tag{15}$$

U_A gets nothing but some random function. Since the set of active users is not sent to users in our scheme, so user can not construct the $A_j(x)$. Obviously, our scheme can successfully resist to this kind of attack.

Now we show that *scheme*1 satisfies the security model of *section*2.1. More precisely, we can prove the following results.

Result 2: The *scheme*1 is an unconditionally secure and self-healing session key distribution scheme with revocation capability in the security model of *section*2.1.

Proof: To prove this result is correct, we need to prove *scheme*2 satisfies all seven definitions in *section*2.

1.1.a) Session key recovery by a user U_i is described in step 4 of the construction. Therefore we have that $H(K_j|Z_{i,j}) = 0, H(Z_{i,j}|B_j, S_i) = 0 \quad (1)$.

1.1.b) For any set $L \subseteq \mathcal{U}$, $|L| \leq t$, t attackers $U_{l_1}, \ldots, U_{l_t} \in L$ and any active user $U_i \notin L$, if attacker want to get the secret value of any active user, they can take the two kinds of ways.

1. The first way is that every attacker $U_{l_1}, \ldots, U_{l_t} \in L$ attacks the system individually. According to our *Result*1, any user can not get any other use's secret information. So t attackers can not successfully attack the system individually.

2. The second way is that t attackers $U_{l_1}, \ldots, U_{l_t} \in L$ collude to attack the system. We show that the coalition of L gets nothing about the personal secret S_i of U_i(active user). For any session j, U_i's personal secret $\psi(r_i)$ is a value over a t-degree polynomial $\psi(x)$. Since the coalition gets at most t values over the t-degree polynomial $\psi(x)$, it is impossible for coalition L to learn $\psi(r_i)$. Since U_i's personal secret $\psi(r_i)$ is an element of F_q and $\psi(x)$ is picked randomly, we have $H(\{s_{i,j}\}) = H(\{\psi(r_i), sid_j\}) = H(\{\psi(r_i)\}) = \log q$. And the secret value r_i is a random value in F_q.

So according to (1) and (2) we have

$$\begin{aligned}
&H(S_i|\{S_l\}_{U_l \in L}, B_1, \ldots, B_m) \\
&= H(\{s_{i,j}\}|\{S_l\}_{U_l \in L}, B_1, \ldots, B_m) \\
&= H(\{\psi(r_i), sid_j\}|\{\psi(r_l)\}_{U_l \in L}, B_1, \ldots, B_m) \\
&= H(\{\psi(r_i), sid_j\}) \\
&= H(\{\psi(r_i)\}) = \log q.
\end{aligned} \tag{16}$$

1.1.c) Since the *j-th* session key $K_j = \mathcal{E}_{\beta_j}(K_{j-1})$ is independent of the personal secret $\psi(r_i)$ for $i = 1, \ldots, n$, and is generated by the self-healing key β_j which is a random number and the previous session key K_{j-1}, so the personal secret keys alone do not give any information about any session key. Since the self-healing key β_j, blind value BV_j and session key K_j is picked randomly, the set of session keys $K_1, \ldots, K_m$ can not determined only by broadcast messages. So $Z_{i,j} = K_j$ can not be determined by only personal key S_i or broadcast message B_j. And we have that $H(Z_{i,j}|B_1, \ldots, B_m) = H(Z_{i,j}) = H(Z_{i,j}|S_1, \ldots, S_n) \quad (3)$.

1.2) Assume that a collection L of t revoked users collude in *j-th* session. It is impossible for coalition L to learn the *j-th* session key K_j because knowledge of K_j implies the knowledge of the personal secret $\{\psi(r_i), r_i\}$ of user $U_i \notin L$ or the self-healing key β_j. This coalition L has no information on β_j or $\{\psi(i), r_i\}$

for $U_i \notin L$. L knows the points $\{r_i, \psi(r_i) : U_i \in L\}$. The size of the coalition L is at most t. Consequently, the colluding users only have at most t-points on the polynomial $\psi(x)$. But degree of the polynomial $\psi(x)$ is t. Hence the coalition L cannot recover $\psi(x)$. Moreover, secret value r_i is picked randomly. So they in turn make K_j appears to be randomly distributed to L. Therefore, K_j is completely safe. Hence, we have that
$H(K_j|B_j, S_i) = 0, H(K_j|B_j, \{S_l\}_{U_l \in L}) = H(K_j) \quad (4)$.

1.3) For any U_i that is an active user in session j_1 and j_2 $(1 \le j_1 \le j_2 \le m)$, he can recover K_{j_2} and hence he can obtain self-healing keys $\beta_1, \ldots, \beta_{j_2}$. And U_i also has the session key K_{j_1} which is recovered in j_1-th session. Hence, by the method of step 4 in $Scheme1$, U_i can subsequently recover the whole missed session keys. Hence we have that $H(K_j|Z_{i,j_1}, Z_{i,j_2}) = 0 \quad (5)$.

2) We can easily prove that $scheme1$ satisfy $Definition2$ following the same line of security analysis of [1], so $scheme1$ can satisfy t-wise forward and backward secrecy.

In conclusion, $scheme1$ is an unconditionally secure and self-healing session key distribution scheme with revocation capability.

5 The Proposed Scheme 2

In order to reduce the communication overhead, we construct the $scheme2$. The $scheme2$ focuses on computational secure and efficient key distribution scheme with self-healing property and revocation capability for large groups over insecure wireless sensor networks. In the application that sensor nodes have limited computing and energy resources, $scheme2$ is efficient in terms of communication complexity and computational complexity.

5.1 The Construction of Scheme 2

1) Initialization: Assume that $\mathcal{U} = \{U_1, \ldots, U_n\}$. The operations in $scheme2$ take place in a finite field F_q. It is never allowed that a revoked user to rejoin the group in a later session. Let f_1: $F_q \rightarrow F_q$ be cryptographically secure one-way function.

2) Setup: The group manager GM chooses at random a t-degree polynomial $\psi(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_t x^t$ in $F_q[x]$ and a random initial session identifier $sid_0 \in F_q$. And the GM selects random secret value $r_i \in F_q$ and computes the secret key $\psi(r_i)$ for each user $U_i (1 \le i \le n)$. The GM also chooses randomly two initial key seeds, the forward key seed $SD^F \in F_q$ and the backward key seed $SD^B \in F_q$. Each user U_i gets its personal key $s_{i,0} = \{r_i, \psi(r_i), SD^F, sid_0\}$ from the GM via the secure communication channel between them. GM repeatedly applies (in the pre-processing time) the one-way function f_1 on SD^B and computes the one-way key chain of length m: $K_j^B = f_1(K_{j-1}^B) = f_1^j(SD^B)$, $K_j^F = f_1(K_{j-1}^F) = f_1^j(SD^F)$ for $1 \le j \le m$. The j-th session key is computed as

$$K_j = K_j^F + K_{m-j+1}^B (in\ F_q) \tag{17}$$

Note that sid_0, SD^F and SD^B are all different from each other.

3) Broadcast: In the *j-th* session key distribution, $j \geq 1$, the GM computes user U_i's *j-th* session identifier $sid_j = f_1(sid_{j-1})$. Note that initial value sid_0 as secret value is stored by user all the time. For $j > 1$, it replaces the previous session identifier sid_{j-1} by the current value sid_j. $\mathcal{U}_{act_j} = \{U_{act_1}, \ldots, U_{act_{a_j}}\}(|\mathcal{U}_{act_j}| = a_j)$, be the set of all active users for *j-th* session, where a_j is the number of active user in session j. Let $\mathcal{R}_{act_j} = \{r_{act_1}, \ldots, r_{act_{a_j}}\}$, $|\mathcal{R}_{act_j}| = a_j$, be the set of all active users' secret values for *j-th* session. In the *j-th* session key distribution, GM locates the backward key K_{m-j+1}^B in the backward key chain and constructs the polynomial

$$B_j = \{\phi_j(x) = A_j(x) K_{m-j+1}^B + sid_j \psi(x)\}, \tag{18}$$

where $A_j(x) = (x - BV_j) \prod_{i=1}^{a_j} (x - r_{act_i}) + 1$ as access polynomial.

4) Session Key and Self-healing Key Recovery: At first, an active user U_i computes the session identifier $sid_j = f_1(sid_{j-1})$ and replaces the previous session identifier sid_{j-1} by the current value sid_j. After that, U_i evaluates $\psi(r_i)$ and $A_j(r_i)$. When an active user U_i receives the session *j-th* key distribution message B_j, it can evaluate $A_j(r_i)$ using its secret value r_i, it can get $A_j(r_i) = 1$. However, for a revoked user, it is a random value. Then U_i computes the current backward session key

$$K_{m-j+1}^B = \phi_j(r_i) - sid_j \psi(r_i). \tag{19}$$

Finally, U_i computes the *j-th* forward key $K_j^F = f_1(K_{j-1}^F) = f_1^{j-1}(SD^F)$ and evaluates the current session key $K_j = K_j^F + K_{m-j+1}^B$.

Now we explain our self-healing mechanism in above construction: Let U_i be an active user that receives session key distribution message B_{j_1} and B_{j_2} in session j_1 and $j_2 (1 \leq j_1 < j_2)$ respectively. User U_i can recover all the lost session keys K_j for $j_1 < j < j_2$ as follows: U_i recovers the backward key $K_{m-j_2+1}^B$ from the broadcast message B_{j_2} in session j_2, and then repeatedly apply the one-way function f_1 to compute the backward keys

$$K_{m-j+1}^B = f_1^{j_2-j}(K_{m-j_2+1}^B) = f_1^{j_2-j}(f_1^{m-j_2+1}(SD^B)), \tag{20}$$

for all j, $j_1 \leq j \leq j_2$. Then U_i can compute the forward keys

$$K_j^F = f_1^{j-j_1}(K_{j_1}^F) = f_1^{j-j_1}(f_1^{j_1}(SD^F)), \tag{21}$$

for all j, $j_1 \leq j \leq j_2$ by repeatedly applying f_1 on the forward seed SD^F or on the forward key $K_{j_1}^F$ of the j_1-*th* session. Finally, U_i can recover all the session keys $K_j = K_j^F + K_{m-j+1}^B$, for $j_1 \leq j \leq j_2$.

5) Add New Users: When the GM adds a new user U_v started from session j, it just chooses an unique secret value $r_v \in F_q$, and computes the personal secret key $\psi(r_v)$, sid_j and forward key K_j^F gives $s_{v,j} = \{r_v, \psi(r_v), K_j^F, sid_j\}$ to U_v through the secure communication channel between them.

5.2 The Security Analysis of Scheme 2

In this section, we show that our $scheme2$ is resistant to the attack which is proposed in $section3$. Moreover, the $scheme2$ is a self-healing key distribution scheme with revocation capability. More precisely, we can prove the following results.

Result 3: The $scheme2$ is resistant to the attack in section 3.

Proof: we can prove $Result3$ successfully following the same line of proving $Result1$.

Result 4: The $scheme2$ is secure self-healing session key distribution scheme with revocation capability.

Proof: 1). (self-healing property) As shown in step 4 of section 5.1, user U_i that is active user in session j_1 and $j_2 (1 \leq j_1 < j_2)$ receives the broadcast message B_{j_1} and B_{j_2}, can recover the session keys between j_1 and j_2. So $scheme2$ has self-healing capability.

2). (t-revocation property) We show that $scheme2$ is secure against coalitions of size at least t. Let $L \subseteq \mathcal{U}$ be the set of t revoked users colluding in session j. It is impossible for coalition L to get the j-th the session key K_j. Since the knowledge of K_j implies the knowledge of the backward key K_{m-j+1}^B according to $equation(17)$ where K_j^F can be computed by revoked users. And the knowledge of K_{m-j+1}^B implies the knowledge of the personal secret key $\psi(r_i)$ of user $U_i \notin L$. Then we show that colluding users can not get the personal secret key $\psi(r_i)$ of user $U_i \notin L$. The coalition L knows the values $\{\psi(r_i) : U_i \in L\}$. And the size of the coalition L is at most t. Consequently, the colluding users only have at most t-points on the curve of t-degree polynomial $\psi(x)$. Hence the coalition L can not recover $\psi(x)$, which in turn makes K_{m-j+1}^B appears random to L. Therefore, session key K_j is secure against the coalition and has revocation capability.

Result 5: The $scheme2$ achieves t-wise forward secrecy and t-wise backward secrecy.

Proof: 1). (t-wise forward secrecy) Let $L \subset \mathcal{U}$, where $|L| \leq t$ and all user $U_l \in L$ are revoked before the current session j. If the revoked users want to know the session K_j, they must know the backward key K_{m-j+1}^B according $equation(17)$. However the coalition L can not get any information about the current session key K_{m-j+1}^B even with the knowledge of group keys before session j. Since in order to know K_{m-j+1}^B, $U_l \in L$ needs to know at least $t+1$ values on the polynomial $\psi(x)$. Since the size of coalition L is at most t, the coalition L has at most t personal secret keys and gets t values on the polynomial $\psi(x)$. But at least $t+1$ values are needed on the polynomial $\psi(x)$ to recover the current backward session key K_{m-j+1}^B for any user $U_l \in R$. Moreover, it is computationally infeasible to compute $K_{j_1}^B$ from $K_{j_2}^B$ for $j_1 < j_2$. The user in L might know the

sequence of backward keys $K_m^B, \ldots, K_{m-j+2}^B$, but cannot compute K_{m-j+1}^B and consequently K_j from this sequence. Hence the *scheme*2 is t-wise forward secure.
2). (t-wise backward secrecy) Let $J \subseteq \mathcal{U}$, where $|J| \leq t$ and all user $U_l \in J$ join after the current session j. The coalition J can not get any information about any previous session key K_{j_1} for $j_1 < j$ even with the knowledge of group keys after session j. The reason is that in order to know K_{j_1}, $U_l \in J$ requires the knowledge of j_1-*th* forward key $K_{j_1}^F = f_1(K_{j_1-1}^F) = f_1^{j_1-1}(S^F)$. Now when a new user U_v joins the group starting from session $j+1$, the GM gives U_v's $(j+1)$-*th* forward key K_{j+1}^F instead of the initial forward key seed S^F, together with the personal secret key $\psi(r_v)$. Note that $K_{j+1}^F = f_1(K_j^F)$. Hence it is computationally infeasible for the new user U_v to compute the previous forward keys $K_{j_1}^F$ for $j_1 \leq j$ because of the one-way property of the function f_1. Hence, *scheme*2 is t-wise backward secure.

Remark: In *scheme*1 and *scheme*2, we does not consider the collusion between the revoked user and the joined newly user. Hence, we will improve this weakness as our future work.

6 Efficiency

In terms of storage complexity, *scheme*1 requires each user stores secret value r_i, personal secret key $\psi(r_i)$, session identifier sid_0 and j-*th* session identifier sid_j instead of $(j-1)$-*th* session identifier. Hence, the user's storage size is $4 \log q$, which is constant storage. And *scheme*1's communication complexity is $\max\{(t+j+1)\log q,\ (a_j+t+1)\log q\}$, where t is maximum number of revoked users and a_j is number of active users in the session j. *scheme*2 needs only one more forward key S_F than that of *scheme*1, so *scheme*2's storage complexity is $5 \log q$. In *scheme*2, the communication complexity in the j-*th* session is $\max\{(t+1)\log q,\ (a_j+1)\log q\}$. In the application environment requiring strong resistance to revokers' collusion attack or revoked users being more than the active users($t > a_j$), the advantage of our schemes is obvious. *scheme*1's

Table 1. Comparison among different self-healing key distribution schemes in j-*th* session

Schemes	Storage Overhead	Communication Overhead	Security
*scheme*3 of [2]	$(m-j+1)^2 \log q$	$(mt^2+2mt+m+t)\log q$	*Unconditional*
*scheme*2 of [4]	$(m-j+1)\log q$	$(2tj+j)\log q$	*Unconditional*
*scheme*2 of [5]	$(m-j+1)\log q$	$(tj+j)\log q$	*Unconditional*
scheme of [8]	$2\log q$	$(2tj)\log q$	*Unconditional*
scheme of [1]	$3\log q$	$(t+j+1)\log q$	*Insecure*
Our *scheme*1	$4\log q$	$max\{(t+j+1),(a_j+j+1)\}\log q$	*Unconditional*
Our *scheme*2	$5\log q$	$max\{(t+1),(a_j+1)\}\log q$	*Computational*

communication complexity is $(t + j + 1) \log q$ and *scheme*2's communication complexity is $(t + 1) \log q$.

*Table*1 compares storage overhead and communication complexity between the proposed schemes and previous schemes.

7 Conclusion

In this paper, we analyze the weakness of Dutta's scheme [1] and attack it successfully. And we slightly modify the Dutta's scheme and construct *scheme*1. *scheme*1 adopts the access polynomial in order to resist to the proposed attack. Moreover, *scheme*1 is an efficient unconditionally secure self-healing key distribution scheme without sacrificing the storage and communication complexity compared to Dutta's scheme. Finally, we use of the one way function to design *scheme*2 which is more efficient in communication complexity. And it focuses on computational secure. In the application environment requiring strong resistance to revoker's collusion attack or revoked user being more than the active users, the advantage of our schemes is obvious. They are scalable to large groups in wireless sensor network of very bad environment, where many users need to be revoked. In the future, we dedicate our work to overcome the weakness of the collusion between the revoked user and the joined newly user. [14] partly solves this problem by subset difference method. In the way of [14], we consider the idea of binding the time at which user joined the group with its ability to recover a previous group key in our scheme.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (Grant no. 60773035); the Key Projects Foundation of Ministry of Education of China (Grant no. 205136); The Foundation of Science and Technology Bureau of Sichuan Province, China (Grant no. 05JY029-131).

References

1. Ratna, D., Yongdong, W., Sourav, M.: Constant Storage Self-Healing Key Distribution with Revocation in Wireless Sensor Network. In: IEEE International Conference on Communications, ICC 2007, pp. 1323–1328 (2007)
2. Jessica, S., Sara, M., Matt, F., Dirk, B., Michael, M., Drew, D.: Self-healing key distribution with Revocation. In: Proceedings of IEEE Symposium on Security and Privacy 2002, pp. 224–240 (2002)
3. Donggang, L., Peng, N., Kun, S.: Efficient Self-healing Key Distribution with Revocation Capability. In: Proceedings of the 10th ACM CCS 2003, pp. 27–31 (2003)
4. Carlo, B., Paolo, D'A., Alfredo, D.S., Massimiliano, L.: Design of Self-healing Key Distribution Schemes. Design Codes and Cryptology, 15–44 (2004)
5. Dowon, H., Jusung, K.: An Efficient Key Distribution Scheme with Self-healing Property. In: IEEE Communication Letters 2005, vol. 9, pp. 759–761 (2005)

6. Biming, T., Mingxing, H.: A Self-healing Key Distribution Scheme with Novel Properties. International Journal of Network Security 7(1), 115–120 (2008)
7. Ratna, D., Sourav, M.: Improved Self-Healing Key Distribution with Revocation in Wireless Sensor Network. In: IEEE Wireless Communications and Networking Conference 2007, pp. 2965–2970 (2007)
8. Xukai, Z., Yuanshun, D.: A Robust and Stateless Self-Healing Group Key Management Scheme. In: International Conference on Communication Technology, ICCT 2006, vol. 28, pp. 455–459 (2006)
9. Sara, M.M., Michael, M., Jessica, S., Dirk, B.: Slidinig-window Self-healing Key Distribution with Revocation. In: ACM Workshop on Survivable and Self-regenerative Systems 2003, pp. 82–90 (2003)
10. German, S.: On Threshold Self-healing Key Distribution Schemes. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 340–354. Springer, Heidelberg (2005)
11. Carlo, B., Paolo, D'A., Alfredo, D.S.: On Self-Healing Key Distribution Scheme. IEEE Transactions on Information Theory 52, 5455–5467 (2006)
12. German, S.: Self-healing key distribution schemes with sponsorization. In: Dittmann, J., Katzenbeisser, S., Uhl, A. (eds.) CMS 2005. LNCS, vol. 3677, pp. 22–31. Springer, Heidelberg (2005)
13. Ratna, D., Ee-Chien, C., Sourav, M.: Efficient self-healing key distribution with revocation for wireless sensor networks using one way key chains. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 385–400. Springer, Heidelberg (2007)
14. Sencun, Z., Sanjeev, S., Sushil, J.: Adding Reliable and Self-healing Key Distribution to the Subset Difference Group Rekeying Method for Secure Multicast. Networked Group Communications, George Mason University Technical Report ISETR-03-02, Munich, Germany (2003)

Advances in Ultralightweight Cryptography for Low-Cost RFID Tags: Gossamer Protocol

Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M.E. Tapiador, and Arturo Ribagorda

Computer Science Department, Carlos III University of Madrid
{pperis,jcesar,jestevez,arturo}@inf.uc3m.es
http://www.lightweightcryptography.com

Abstract. The design of ultralightweight authentication protocols that conform to low-cost tag requirements is imperative. This paper analyses the most important proposals (except for those based in hard problems such as the HB [1–3] family) in the area [4–6] and identifies the common weaknesses that have left all of them open to various attacks [7–11]. Finally, we present Gossamer, a new protocol inspired by the recently published SASI scheme [13], that was lately also the subject of a disclosure attack by Hernandez-Castro et al. [14]. Specifically, this new protocol is designed to avoid the problems of the past, and we examine in some deep its security and performance.

1 Introduction

In a RFID system, objects are labeled with a tag. Each tag contains a microchip with a certain amount of computational and storage capabilities, and a coupling element. Such devices can be classified according to memory type and power source. Another relevant parameter is tag price[1], which creates a broad distinction between high-cost and low-cost RFID tags.

Each time a new protocol is defined, the tag's class for which it is envisioned should also be specified. We note that, depending on the class of the tag, the maximum security level that can be supported will also be very different. For example, the security level of a relatively high-cost tag as those used in e-passports should be much higher than that of a low-cost tag employed in supply chain management (i.e. tags compliant to EPC Class-1 Generation-2 specification).

In [13], Chien proposed a tag classification mainly based on which were the operations supported on-chip. High-cost tags are divided into two classes: "full-fledged" and "simple". Full-fledged tags support on-board conventional cryptography like symmetric encryption, cryptographic one-way functions and even public key cryptography. Simple tags can support random number generators and one-way hash functions. Likewise, there are two classes for low-cost RFID tags. "Lightweight" tags are those whose chip supports a random number generation and simple functions like a Cyclic Redundancy Code (CRC) checksum,

[1] The *rule of thumb* of gate cost says that every extra 1,000 gates increases chip price by 1 cent [15].

but not cryptographic hash function. “Ultralightweight” tags can only compute simple bitwise operations like XOR, AND, OR, etc. These ultralightweight tags represent the greatest challenge in terms of security, due to their expected wide deployment and very limited capabilities.

2 A Family of Ultralightweight Mutual Authentication Protocols

In 2006, Peris et al. proposed a family of Ultralightweight Mutual Authentication Protocols (henceforth referred to as the UMAP family of protocols). Chronologically, M^2AP [4] was the first proposal, followed by EMAP [5] and LMAP [6].

These protocols are based on the use of pseudonyms to guarantee tag anonymity. Specifically, an index-pseudonym is used by an authorized reader to retrieve the information associated with a tag (tag identification phase). Additionally, a key -divided in several subkeys- is shared between legitimate tags and readers (back-end database). Both readers and tags use these subkeys to build the messages exchanged in the mutual authentication phase.

In line with their real processing capabilities, tags only support on-board simple operations. Indeed, these protocols are based on bitwise XOR ($\oplus$), bitwise OR ($\vee$), bitwise AND ($\wedge$) and addition mod 2^m. By contrast, only readers need to generate pseudorandom numbers; tags only use them for creating fresh messages to the protocol.

In the UMAP family of protocols, the proposed scheme consists of three stages. First, the tag is identified by means of the index-pseudonym. Secondly, the reader and the tag are mutually authenticated. This phase is also used to transmit the static tag identifier (ID) securely. Finally, the index-pseudonym and keys are updated (the reader is referred to the original papers for more details).

2.1 Security Analysis of the UMAP Protocols

Since the publication of the UMAP family of protocols, their security has been analyzed in depth by the research community. In [7, 8] a desynchronization attack and a full disclosure attack are presented. These require an active attacker and several incomplete run executions of the protocol to disclose the secret information on the tag. Later, Chien et al. proposed -based on the same attack model- a far more efficient full-disclosure attack [9]. Additionally, Bárász et al. showed how a passive attacker (an attack model that may be, in certain scenarios, much more realistic) can find out the static identifier and on particular secrets shared by the reader and the tag after eavesdropping on a few consecutive protocol rounds [10, 11].

This leads us to the following conclusions: first, we must define what kind of attack scenarios are applicable. In our opinion, ultralightweight RFID tags have to be resistant to passive attacks but not necessarily to active attacks, because of their severe restrictions (storage, circuitry and power consumption). Regarding passive attacks, we can affirm the following:

- The UMAP family of protocols is based on the composition of simple operations like bitwise AND, XOR, OR and sum mod 2^m. Because all of these are triangular functions (T-functions) [16], the information does not propagate well from left to right. In other words, the bit in position i in the output only depends on bits j = 0,..., i of the input words.
- The use of the bitwise AND or OR operations to build public submessages is a weakness common to all these protocols. When a bitwise AND (OR) operation is computed even over random inputs, the probability of obtaining a one (zero) is $\frac{3}{4}$. In other words, the result is strongly biased. This poor characteristic is the basis of all the passive attacks proposed so far.

3 SASI Protocol

In 2007 Hung-Yu Chien proposed a very interesting ultralightweight authentication protocol providing Strong Authentication and Strong Integrity (SASI) for very low-cost RFID tags [13]. We briefly describe the messages exchanged between the reader (or back-end database) and the tag.

An index-pseudonym (IDS), the tag's private identification (ID), and two keys (k_1/k_2) are stored both on the tag and in the back-end database. Simple bitwise XOR ($\oplus$), bitwise AND ($\wedge$), bitwise OR ($\vee$), addition 2^m and left rotation (Rot(x,y)) are required on the tag. Additionally, random number generation (i.e. n_1 and n_2) is required on the reader. The protocol is divided into three states: tag identification, mutual authentication and updating phase. In the identification phase, the reader (R) sends a "hello" message to the tag (T), and the tag answers with its IDS. The reader then finds, in the back-end database, the information associated with the tag (ID and k_1/k_2), and the protocol continues to the mutual authentication phase. In this, the reader and the tag authenticate each other, and the index-pseudonym and keys are subsequently updated:

R $\rightarrow$ **T** : $A||B||C$
The reader generates nonces n_1 and n_2 to build the submessages as follows: $A = IDS \oplus k_1 \oplus n_1$; $B = (IDS \vee k_2) + n_2$; $C = (k_1 \oplus k_2^*) + (k_2 \oplus k_1^*)$; where $k_1^* = Rot(k_1 \oplus n_2, k_1)$; $k_2^* = Rot(k_2 \oplus n_1, k_2)$

Tag. From messages A and B, the tag can obtain values n_1 and n_2 respectively. Then it locally computes C' and checks if the result is equal to the received value. If this is the case, it sends D and updates the values of IDS, k_1 and k_2: $D = (k_2^* + ID) \oplus ((k_1 \oplus k_2) \vee k_1^*)$; $IDS^{next} = (IDS + ID) \oplus (n_2 \oplus k_1^*)$; $k_1^{next} = k_1^*$; $k_2^{next} = k_2^*$;

T $\rightarrow$ **R** : D

Reader. Verifies D and, if it is equal to the result of its local computation, updates IDS, k_1 and k_2 in the same way as the tag.

3.1 Vulnerability Analysis

From the analysis of the UMAP family of protocols, we conclude that it is necessary to incorporate a non-triangular function in order to increase the security of ultralightweight protocols. At first sight, the SASI protocol complies with this requirement as it includes the left rotation operation (which is non triangular). However, Hernandez-Castro et al. have recently showed that the protocol was not carefully designed [14]. Indeed, a passive attacker can obtain the secret static identifier of the tag (ID) after observing several consecutive authentication sessions. We now summarize the main weaknesses of the protocol (see the original paper for more details):

1. The second component of the IDS updating equation is dependent on the bitwise XOR between n_2 and k_1^*. This gives rise to poor statistical properties as k_1^* is also function of n_2.
2. The key updating equation has a kind of distributive operation that might be employed to attack the protocol, for example: $k_1^* = Rot(k_1 \oplus n_2, k_1) = Rot(k_1, k_1) \oplus Rot(n_2, k_1)$
3. As mentioned in *Section 2.1*, bitwise OR and bitwise AND should be used with extreme care. These operations result in a strongly biased output. For example, the nonce n_2 can be approximated with very good precision by simply computing $n_2 \simeq B - 1$. These operations might therefore be only employed in the inner parts of the protocol but should be avoided in the generation of public submessages (i.e. B and D submessages). In fact, all the exchanged messages should resemble random values as far as possible.

4 Gossamer Protocol

As a consequence of the above observations, we have derived a new protocol, called Gossamer[2], which is inspired by the SASI scheme but hopefully devoid of its weaknesses. Our main aim was to define a protocol with adequate security level and which can realistically be employed in ultralightweight RFID tags.

4.1 Model Suppositions

Each tag stores a static identifier (ID), an index-pseudonym (IDS) and two keys (k_1/k_2) in its memory. This information is also stored in the back-end database. The IDS is employed as a search index to allocate, in the database, all the information linked with each tag. These elements have a length of 96 bits, compatible with all the encoding schemes (i.e. GTIN, GRAI) defined by EPCGlobal. Additionally, tags are given the added requirement of storing the old and potential new values of the tuple (IDS, k_1, k_2), to avoid desynchronization attacks. In spite of this, resiliency against attacks which involve tag manipulation are not considered as these devices are not at all tamper-resistant.

For the implementation of the proposed protocol, only simple operations are available on tags, in accordance with their restrictions: specifically, bitwise XOR

[2] Gossamer: Noun describing a thin film of cobwebs floating in the air (this meaning dates from the 14th century) and an adjective meaning light, delicate, thin enough to let light through, nearly transparent.

($\oplus$), addition mod 2^m ($+$), and left rotation (Rot(x,y)). Rotation may be performed in several different ways. However, the original SASI scheme does not clearly specify the rotation method used. Sun et al., who recently published two desynchronization attacks on SASI, contacted the author to clarify the issue [17]. Chien asserted that $Rot(x, y)$ is a circular shift of x, $wht(y)$ positions to the left where $wht(y)$ denotes the Hamming weight of y. This is probably not optimal from the security point of view as the argument that determines the number of positions rotated is far from uniform. Indeed, this variable follows the following probability distribution:

$$Prob(wht(B) = k) = \frac{\binom{96}{k}}{2^{96}} \tag{1}$$

In our proposed scheme $Rot(x, y)$ is defined perform a circular shift on the value of x, (y mod N) positions to the left for a given value of N (in our case 96).

Random number generation, required in the protocol to supply freshness, is a costly operation, so it is performed by the reader. Moreover, random numbers cannot be indiscriminately employed because their use increases both memory requirements and message counts (which could be costly in certain applications). To significantly increase security, we have also added a specially designed and very lightweight function called $MixBits$. In [18], a detailed description of the methodology used -basically, to evolve compositions of extremely light operands by means of genetic programming, in order to obtain highly non-linear functions- is included. $MixBits$ has an extremely lightweight nature, as only bitwise right shift (>>) and additions are employed. Specifically,

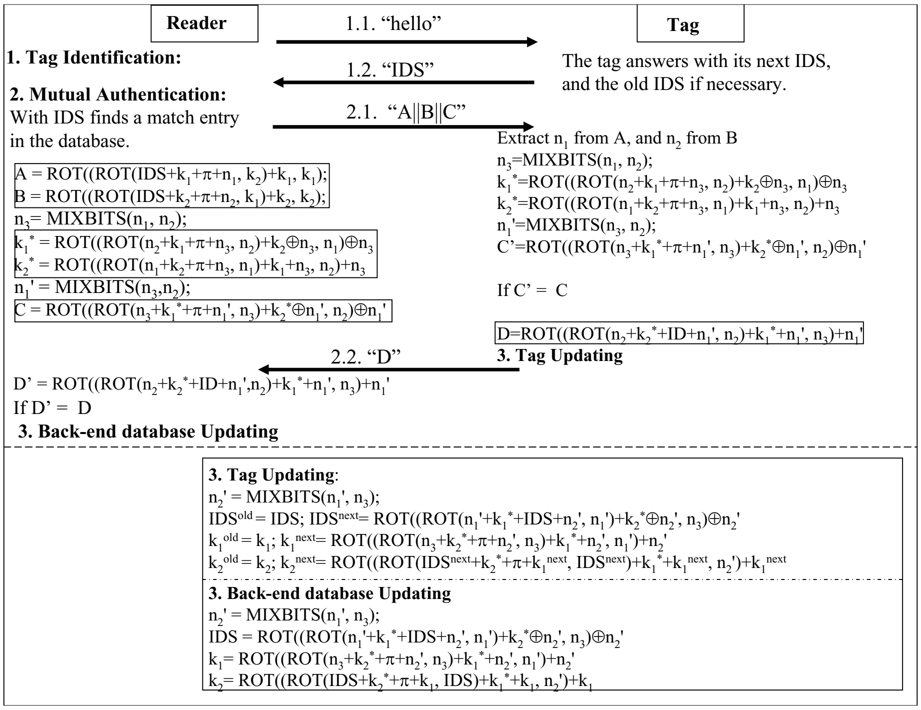
```
Z = MixBits(X,Y)
----------------------------
Z = X;
for(i=0; i<32; i++) {
Z = (Z>>1) + Z + Z + Y  ;}
----------------------------
```

Communication has to be initiated by readers, since tags are passive. The communication channel between the reader and the database is generally assumed to be secure, but the channel between the reader and the tag can be eavesdropped on. Attacks involving modification of the exchanged messages, the insertion of fraudulent new messages, or message blocking (active attacks), can be discounted.

4.2 The Protocol

The protocol comprises three stages: tag identification phase, mutual authentication phase, and updating phase. *Figure 1* shows the exchanged messages.

Tag Identification. The reader first sends a "hello" message to the tag, which answers with its potential next IDS. With it, the reader tries to find an identical entry in the database. If this search succeeds, the mutual authentication phase starts. Otherwise the identification is retried but with the old IDS, which is backscattered by the tag upon request.



Reader — **Tag**

1. Tag Identification:

1.1. "hello" (Reader → Tag)

1.2. "IDS" (Tag → Reader)

The tag answers with its next IDS, and the old IDS if necessary.

2. Mutual Authentication:
With IDS finds a match entry in the database.

$\boxed{A = ROT((ROT(IDS+k_1+\pi+n_1, k_2)+k_1, k_1);}$
$\boxed{B = ROT((ROT(IDS+k_2+\pi+n_2, k_1)+k_2, k_2);}$
$n_3 = MIXBITS(n_1, n_2);$
$\boxed{k_1^* = ROT((ROT(n_2+k_1+\pi+n_3, n_2)+k_2 \oplus n_3, n_1)\oplus n_3}$
$\boxed{k_2^* = ROT((ROT(n_1+k_2+\pi+n_3, n_1)+k_1+n_3, n_2)+n_3}$
$n_1' = MIXBITS(n_3, n_2);$
$\boxed{C = ROT((ROT(n_3+k_1^*+\pi+n_1', n_3)+k_2^* \oplus n_1', n_2)\oplus n_1'}$

2.1. "A||B||C" (Reader → Tag)

Extract n_1 from A, and n_2 from B
$n_3=MIXBITS(n_1, n_2);$
$k_1^*=ROT((ROT(n_2+k_1+\pi+n_3, n_2)+k_2\oplus n_3, n_1)\oplus n_3$
$k_2^*=ROT((ROT(n_1+k_2+\pi+n_3, n_1)+k_1+n_3, n_2)+n_3$
$n_1'=MIXBITS(n_3, n_2);$
$C'=ROT((ROT(n_3+k_1^*+\pi+n_1', n_3)+k_2^*\oplus n_1', n_2)\oplus n_1'$

If C' = C

$\boxed{D=ROT((ROT(n_2+k_2^*+ID+n_1', n_2)+k_1^*+n_1', n_3)+n_1'}$
3. Tag Updating

2.2. "D" (Tag → Reader)

$D' = ROT((ROT(n_2+k_2^*+ID+n_1',n_2)+k_1^*+n_1', n_3)+n_1'$
If D' = D
3. Back-end database Updating

3. Tag Updating:
$n_2' = MIXBITS(n_1', n_3);$
$IDS^{old} = IDS;\ IDS^{next} = ROT((ROT(n_1'+k_1^*+IDS+n_2', n_1')+k_2^*\oplus n_2', n_3)\oplus n_2'$
$k_1^{old} = k_1;\ k_1^{next} = ROT((ROT(n_3+k_2^*+\pi+n_2', n_3)+k_1^*+n_2', n_1')+n_2'$
$k_2^{old} = k_2;\ k_2^{next} = ROT((ROT(IDS^{next}+k_2^*+\pi+k_1^{next}, IDS^{next})+k_1^*+k_1^{next}, n_2')+k_1^{next}$

3. Back-end database Updating
$n_2' = MIXBITS(n_1', n_3);$
$IDS = ROT((ROT(n_1'+k_1^*+IDS+n_2', n_1')+k_2^*\oplus n_2', n_3)\oplus n_2'$
$k_1 = ROT((ROT(n_3+k_2^*+\pi+n_2', n_3)+k_1^*+n_2', n_1')+n_2'$
$k_2 = ROT((ROT(IDS+k_2^*+\pi+k_1, IDS)+k_1^*+k_1, n_2')+k_1$

† π = 0x3243F6A8885A308D313198A2 ($L = 96$ bits).

Fig. 1. Gossamer Protocol

Mutual Authentication. With *IDS*, the reader acquires the private information linked to the tag, identified from the database. Then the reader generates nonces n_1 and n_2 and builds and sends to the tag $A||B||C$ (see *Figure 1*). Note that the equations used in the generation of public messages, as do those used in the computation of internal values, generally follow the scheme below:

$$n_{i+2} = MIXBITS(n_i, n_{i+1}) \quad (2)$$

$$M_i = ROT((ROT(n_{i+1} + k_i + PI + n_{i+2}, n_{i+1}) + k_{i+1} \oplus n_{i+2}, n_i) \oplus n_{i+2} \quad (3)$$

$$M_{i+1} = ROT((ROT(n_i + k_{i+1} + PI + n_{i+2}, n_i) + k_i + n_{i+2}, n_{i+1}) + n_{i+2} \quad (4)$$

From submessages A and B, the tag extracts nonces n_1 and n_2. Then it computes n_3/n_1' and k_1^*/k_2^* and builds a local version of submessage C'. This is compared with the received value. If it is verified, the reader is authenticated. Finally, the tag sends message D to the reader. On receiving D, this value is compared with a computed local version. If comparison is successful, the tag is authenticated; otherwise the protocol is abandoned.

Index-Pseudonym and Key Updating. After successfully completing the mutual authentication phase between reader and tag, they locally update *IDS* and keys (k_1/k_2) as indicated in *Figure 1.* As we have just seen, submessages C/D allow reader/tag authentication, respectively. Moreover, the use of submessages C/D results in confirmation of synchronization for the internal secret values (n_3/n'_1 and k^*_1/k^*_2) used in the updating phase, preventing straightforward desynchronization attacks.

4.3 Security Analysis

We will now analyze the security of the proposed scheme against relevant attacks:

Data Confidentiality. All public messages are composed of at least three secret values shared only by legitimate readers and genuine tags. Note that we consider private information (ID, k_1, k_2), random numbers (n_1, n_2), and internal values (n_3, n'_1, n'_2, k^*_1, k^*_2) as secret values. The static identifier and the secret keys cannot, therefore, be easily obtained by an eavesdropper.

Tag anonymity. Each tag updates IDS and private keys (k_1, k_2) after successful authentication, and this update process involves random numbers (n_3, n'_1, n'_2). When the tag is interrogated again, a fresh IDS is backscattered. Additionally, all public submessages ($A||B||C||$ and D) are anonymized by the use of random numbers (n_1, n_2, n_3, n'_1). Tag anonymity is thus guaranteed, and location privacy of the tag owner is not compromised.

Mutual Authentication and Data Integrity. The protocol provides mutual authentication. Only a legitimate reader possessing keys (k_1, k_2), can build a valid message $A||B||C$. Similarly, only a genuine tag can derive nonces n_1, n_2 from $A||B||C$, and then compute message D.

Messages C and D, which involve the internal secret values (n_3, n'_1, k^*_1, k^*_2) and nonces (n_1, n_2), allow data integrity to be checked. Note that these values are included in the updating equations (potential next index-pseudonym and keys).

Replay attacks. An eavesdropper could store all the messages exchanged in a protocol run. To impersonate the tag, he could replay message D. However, this response would be invalid as different nonces are employed in each session -this will frustrate this naive attack. Additionally, the attacker could pretend that the reader has not accomplished the updating phase in the previous session. In this scenario, the tag is identified by the old index-pseudonym and the attacker may forward the eavesdropped values of $A||B||C$. Even if this is successful, no secret information is disclosed and the internal state is unchanged in the genuine tag, so all these attacks are unsuccessful.

Forward Security. Forward security is the property that guarantees the security of past communications even when a tag is compromised at a later stage. Imagine that a tag is exposed one day, making public its secret information (ID, k_1, k_2). The attacker still cannot infer any information from previous sessions as two unknown nonces (n_1, n_2) and five internal secret values (n_3, n'_1, n'_2, k^*_1, k^*_2) are involved in the message creation (mutual authentication phase). Additionally, these internal values are employed in the updating phase. Consequently, past communications cannot be easily jeopardized.

Updating Confirmation. The Gossamer protocol assumes that tags and readers share certain secret values. As these values are locally updated, synchronization is mandatory. Submessages C and D provide confirmation of the internal secret values (n_3, n'_1, k^*_1, k^*_2) and nonces (n_1, n_2). These values are employed in the updating stage. So the correct update of values IDS and keys (k_1, k_2) is implicitly ensured by submessages C and D.

Unintentional transmission errors can happen in the received messages since a radio link is used. This is an extremely serious issue for message D, since it can result in a loss of synchronization. However, the tuple (IDS, k_1, k_2) is stored twice in the tag memory -once with the old values, the other with the potential next values. With this mechanism, even in the event that message D is incorrectly received, the tag and the reader can still authenticate with the old values. So the reader and the tag will be able to recover their synchronized state.

4.4 Performance Analysis

Our proposed protocol is now examined from the point of view of computational cost, storage requirements and communication cost. Additionally, *Table 1* compares the most relevant ultralightweight protocol proposals (see *Section 1*) from a performance perspective.

Table 1. Performance Comparison of Ultralightweight Authentication Protocols

	U-MAP family [4–6]	SASI [13]	Gossamer
Resistance to Desynchronization Attacks	No	No	Yes
Resistance to Disclosure Attacks	No	No	Yes
Privacy and Anonymity	Yes	Yes	Yes
Mutual Authentication and Forward Security	Yes	Yes	Yes
Total Messages for Mutual Authentication	4-5L	4L	4L
Memory Size on Tag	6L	7L	7L
Memory Size for each Tag on Database	6L	4L	4L
Operation Types on Tag	$\oplus$, $\vee$, $\wedge$, $+$	$\oplus$, $\vee$, $\wedge$, $+$, Rot[2]	$\oplus$, $+$, Rot[3], *MixBits*

[1] L designates the bit length of variables used.
[2] $Rot(x, y) = x << wht(y)$, being $wht(y)$ the Hamming weight of vector y.
[3] $Rot(x, y) = x << (y \bmod L)$ for a given value of L -in our case $L = 96$.

Computational cost. The protocol we have proposed only requires simple bitwise XOR, addition 2^m, left rotation, and the *MixBits* function on tags. These operations are very low-cost and can be efficiently implemented in hardware.

When comparing Gossamer with the protocol SASI, we can observe that the bitwise AND and OR operations are eliminated, and the light *MixBits* operation is added for increased security. *MixBits* is very efficient from a hardware perspective. The number of iterations of this function is optimized to guarantee a good diffusion effect. Specifically, it consumes $32 \times 4 \times (96/m)$ clock cycles, m being the word length used to implement the protocol (i.e. m = 8, 16, 32, 64, 96). As this may have a cost impact on the temporal requirements, we have minimized the number of *MixBits* calls.

Storage requirement. Each tag stores its static identifier (ID) and two records of the tuple (IDS, k_1, k_2) -with old and potential new values. A 96-bit length is assumed for all elements in accordance with EPCGlobal. The ID is a static value, thus stored in ROM. The remaining values ($96 \times 6 = 576$ bits) are stored in a rewritable memory because they need to be updated.

In the protocol SASI, two temporal nonces are linked to each session. We include an additional value derived from the previous nonces ($n_{i+2} = MixBits(n_i, n_{i+1})$). As these nonces are updated three times in the internal steps of the protocol, our scheme is roughly equivalent to the use of five fresh random numbers. So, with the relatively light penalty of storing an extra nonce, the security level seems to be notably increased.

Communication cost. The proposed protocol performs mutual authentication and integrity protection with only four messages, so in this sense it is similar to the SASI scheme. In the identification phase, a "hello" and IDS message are sent over the channel. Messages $A||B||C$ and D are transmitted in the authentication phase. So a total of 424 bits are sent over the channel - considering 5 bytes for the "hello" message.

5 Conclusions

We now present some conclusions: firstly those related with RFID security in general, then specifically related to the security of ultralightweight protocols.

5.1 General Considerations

Price and operability are the main issues whenever a new technology appears (i.e. bluetooth, wireless, etc.), security frequently being only a side consideration. To avoid past errors, however, the use of secure solutions should be generalized. Otherwise, the massive deployment of RFID technology runs the risk of being significantly delayed. Since 2003, it seems that the general awareness on the security issues of RFID systems (notably privacy) has been considerably increased, as reflected by a steady increment in the number of research publications on the field. However, the majority of proposals to secure RFID tags make the same two errors:

Tag Class. The tag's class for which the proposed protocol should be intended is not clearly specified in most of the proposals. However, the number of available resources (memory, circuitry, power consumption, etc.) hugely varies from one to another. In other words, not all tags will support the same operation set. For example, public cryptography is applicable for the most expensive RFID tags [19, 20], but it clearly exceeds the capabilities of low-cost RFID tags.

Additionally, the same security level cannot be asked to each RFID class. It is not sensible for a low-cost RFID tag (eg. a tagged biscuit packet) to have the same security level as that of an e-passport.

Tag Resources. Most of the proposed schemes are not realistic with respect to tag resources. Many lightweight cryptographic primitives have been recently proposed, and significant progress is being made in each research area. Clearly, there have been great improvements in the design of lightweight stream/block ciphers [21–25], but the design of lightweight hash functions [26, 12] and PRNGs [27] remains a pending task.

Hash functions are considered a better choice within the RFID security community regarding implementation. As a result, most of the proposed protocols are based on the use of hash functions, and some of these also include a PRNG. In spite of this, many authors claim that the proposed schemes are appropriate for low-cost RFID tags (lightweight and ultralightweight). However, standard hash functions demand more than 5.5K gates (130 nm) [28] - 8K gates (350 nm) [29], which is over the maximum number of gates (3K - 4K gates) that can be devoted to security functions in this tags. Note that the additional resources, needed to support on-chip the PRNG, would increase the total number of logic gates required.

Regarding standardization, there was previously a clear lack of harmonization, and major RFID vendors offered proprietary systems in the earlier implementations. Fortunately, things are changing rapidly. One of the most important standards is the EPCglobal Class-1 Generation-2 RFID specification (known as Gen-2 for short) [30, 31]. Gen-2 specification represents a significant advance for the widespread introduction of this technology, but its security level is extremely low (i.e. privacy is compromised as the EPC is transmitted in clear on the channel). Some authors intending to increase its security level proposed slight modifications in this specification [32–35]. Despite the fact that standards are being increasingly adopted by many companies, other developers base the security of their tags on proprietary solutions. However, the use of proprietary solutions is not altogether bad if algorithms are published so they can be scrutinized by the research community. As time has shown, the security of an algorithm cannot reside in its obscurity. Good examples of this are Texas Instruments DST tags [36] and Philips Mifare cards [37–39]. Companies should learn from past errors and make their proprietary algorithms public.

5.2 Ultralightweight Protocols

In 2003, Vajda et al. published the first article proposing the use of lightweight cryptography [42]. The following year, Juels introduced the concept of minimalist cryptography [43]. In 2005, there was no proposal in this area, the majority of proposals being based on the use non-lightweight hash functions. The year after, Peris et al. proposed the UMAP family of protocols. From the aforementioned protocols, we can infer the following considerations:

Interest. The protocols arouse interest in the design of new ultralightweight protocols. Indeed, they have inspired the proposal of other protocols [13, 40, 41]. Additionally, as can be seen below, the security of the UMAP family of protocols has been carefully examined by the research community.

Security Weaknesses. The security of the UMAP family of protocols has been analyzed under different assumptions. First, security vulnerabilities were revealed under the hypothesis of an active attacker [7–9]. Secondly, Bárász et al. showed how a passive attacker can disclose part of the secret information stored in the tag's memory [10, 11].

As mentioned in *Section 4.1*, only attacks that do not alter or interfere with communications are considered a real threat in most scenarios. In other words, active attacks are discounted when designing a protocol to meet the requirements of ultralightweight RFID tags.

Operations. Only bitwise AND, XOR, OR and sum mod 2^m are required for the implementation of the UMAP protocol family. At first sight, the choice seems well-conceived as these operations can be efficiently implemented in hardware. However, they are all T-functions, which have a very poor diffusion effect; the information does not propagate well from left to right [16]. Also, as a consequence of the use of bitwise AND and OR operations in the generation of certain messages, the latter were highly biased. These two operands should therefore be avoided in messages passed on the channel, but may be used in inner parts of the protocol.

The protocol SASI was a step further towards a secure protocol compliant with real ultralightweight tag requirements. However, it recently came under attack when Hernandez-Castro et al. showed how a passive attacker can obtain the secret ID by observing several consecutive authentications sessions. Despite this, we consider that the protocol design shows some interesting new ideas (specifically, the inclusion of rotations). The analysis of SASI and the UMAP protocol family has led to the proposal of Gossamer, a new protocol inspired by SASI and examined here both from the security and performance perspective. Indeed, the resources needed for the implementation of Gossamer are very similar to those of SASI the scheme, but Gossamer seems to be considerably more secure because of the use of dual rotation and the $MixBits$ function. The price to be paid, of course, is the throughput (number of authenticated tags per second) of the Gossamer protocol. However, preliminary estimations seem to show that the commonly required figure of 100 responses per second is still achievable.

References

1. Weis, S.: Security parallels between people and pervasive devices. In: Proc. of PERSEC 2005, pp. 105–109. IEEE Computer Society Press, Los Alamitos (2005)
2. Piramuthu, S.: HB and related lightweight authentication protocols for secure RFID tag/reader authentication. In: Proc. of CollECTeR 2006 (2006)
3. Munilla, J., Peinado, A.: HB-MP: A further step in the HB-family of lightweight authentication protocols. Computer Networks 51(9), 2262–2267 (2007)
4. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: M2AP: A minimalist mutual-authentication protocol for low-cost RFID tags. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 912–923. Springer, Heidelberg (2006)
5. 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: Hand. of Workshop on RFID and Lightweight Crypto (2006)

6. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: EMAP: An efficient mutual authentication protocol for low-cost RFID tags. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4277, pp. 352–361. Springer, Heidelberg (2006)
7. Li, T., Deng, R.: Vulnerability analysis of EMAP - an efficient RFID mutual authentication protocol. In: Proc. of AReS 2007 (2007)
8. Li, T., Wang, G.: Security analysis of two ultra-lightweight RFID authentication protocols. In: Proc. of IFIP-SEC 2007 (2007)
9. Hung-Yu, C., Chen-Wei, H.: Security of ultra-lightweight RFID authentication protocols and its improvements. SIGOPS Oper. Syst. Rev. 41(4), 83–86 (2007)
10. Bárász, M., Boros, B., Ligeti, P., Lója, K., Nagy, D.: Breaking LMAP. In: Proc. of RFIDSec 2007 (2007)
11. Bárász, M., Boros, B., Ligeti, P., Lója, K., Nagy, D.: Passive Attack Against the M2AP Mutual Authentication Protocol for RFID Tags. In: Proc. of First International EURASIP Workshop on RFID Technology (2007)
12. Shamir, A.: SQUASH - A New MAC With Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 144–157. Springer, Heidelberg (2008)
13. Chien, H.-Y.: SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity. IEEE Transactions on Dependable and Secure Computing 4(4), 337–340 (2007)
14. Hernandez-Castro, J.C., Tapiador, J.M.E., Peris-Lopez, P., Quisquater, J.-J.: Cryptanalysis of the SASI Ultralightweight RFID Authentication Protocol. IEEE Transactions on Dependable and Secure Computing (submitted) (April 2008)
15. Weis, S.: Security and Privacy in Radio-Frequency Identification Devices. Master Thesis, MIT (2003)
16. Klimov, A., Shamir, A.: New Applications of T-functions in Block Ciphers and Hash Functions. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 18–31. Springer, Heidelberg (2005)
17. Sun, H.-M., Ting, W.-C., Wang, K.-H.: On the Security of Chien's Ultralightweight RFID Authentication Protocol. Cryptology ePrint Archive, http://eprint.iacr.org/2008/083
18. Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda-Garnacho, A., Ramos-Alvarez, B.: Wheedham: An automatically designed block cipher by means of genetic programming. In: Proc. of CEC 2006, pp. 192–199 (2006)
19. Batina, L., Guajardo, J., Kerins, T., Mentens, N., Tuyls, P., Verbauwhede, I.: Public-Key Cryptography for RFID-Tags. In: Proc. of PerCom 2007, pp. 217–222 (2007)
20. Kumar, S., Paar, C.: Are standards compliant elliptic curve cryptosystems feasible on RFID. In: Proc. of RFIDSec 2006 (2006)
21. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments, http://www.ecrypt.eu.org/stream/
22. Hell, M., Johansson, T., Meier, W.: A stream cipher proposal: Grain-128, http://www.ecrypt.eu.org/stream/
23. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
24. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES implementation on a grain of sand. In: Proc. on Information Security, vol. 152, pp. 13–20. IEEE Computer Society, Los Alamitos (2005)

25. Poschmann, A., Leander, G., Schramm, K., Paar, C.: New Light-Weight Crypto Algorithms for RFID. In: Proc. of ISCAS 2007, pp. 1843–1846 (2007)
26. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: An Efficient Authentication Protocol for RFID Systems Resistant to Active Attacks. In: Denko, M.K., Shih, C.-s., Li, K.-C., Tsao, S.-L., Zeng, Q.-A., Park, S.H., Ko, Y.-B., Hung, S.-H., Park, J.-H. (eds.) EUC-WS 2007. LNCS, vol. 4809, pp. 781–794. Springer, Heidelberg (2007)
27. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: LAMED – A PRNG for EPC Class-1 Generation-2 RFID specification. Journal of Computer Standards & Interfaces (2008), doi:10.1016/j.csi.2007.11.013
28. O'Neill, M. (McLoone): Low-Cost SHA-1 Hash Function Architecture for RFID Tags. In: Hand. of Conference on RFID Security (2008)
29. Feldhofer, M., Rechberger, C.: A case against currently used hash functions in RFID protocols. In: Hand. of Workshop on RFID and Lightweight Crypto (2006)
30. Class-1 Generation-2 UHF air interface protocol standard version 1.0.9: "Gen-2" (2005), http://www.epcglobalinc.org/standards/
31. ISO/IEC 18000-6:2004/Amd:2006 (2006), http://www.iso.org/
32. Duc, D.N., Park, J., Lee, H., Kim, K.: Enhancing security of EPCglobal Gen-2 RFID tag against traceability and cloning. In: The 2006 Symposium on Cryptography and Information Security (2006)
33. Chien, H.Y., Chen, C.H.: Mutual authentication protocol for RFID conforming to EPC Class-1 Generation-2 standards. Computer Standards & Interfaces 29(2), 254–259 (2007)
34. Konidala, D.M., Kim, K.: RFID Tag-Reader Mutual Authentication Scheme Utilizing Tag's Access Password. Auto-ID Labs White Paper WP-HARDWARE-033 (January 2007)
35. Burmester, M., de Medewiros, B.: The Security of EPCGen2 Anonymous compliant RFID Protocols. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 490–506. Springer, Heidelberg (2008)
36. Bono, S., Green, M., Stubblefield, A., Juels, A., Rubin, A., Szydlo, M.: Security analysis of a cryptographically-enabled RFID device. In: Proc. of 14th USENIX Security Symposium, pp. 1–16 (2005)
37. Garcia, F.D., de Koning Gans, G., Muijrers, R., van Rossum, P., Verdult, R., Wichers Schreur, R.: Dismantling MIFARE Classic. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283. Springer, Heidelberg (2008)
38. de Koning Gans, G., Hoepman, J.-H., Garcia, F.D.: A Practical Attack on the MIFARE Classic. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 267–282. Springer, Heidelberg (2008)
39. Karten, N., Plotz, H.: Mifare little security, despite obscurity (2007), http://events.ccc.de/congress/2007/Fahrplan/events/2378.en.html
40. Li, T., Wang, G.: SLMAP-A Secure ultra-Lightweight RFID Mutual Authentication Protocol. In: Proc. of Chinacrypt 2007 (2007)
41. Lo, N.-W., Shie, H.-S., Yeh, K.-H.: A Design of RFID Mutual Authentication Protocol Using Lightweight Bitwise Operations. In: Proc. of JWIS 2008 (2008)
42. Vajda, I., Buttyán, L.: Lightweight authentication protocols for low-cost RFID tags. In: Dey, A.K., Schmidt, A., McCarthy, J.F. (eds.) UbiComp 2003. LNCS, vol. 2864. Springer, Heidelberg (2003)
43. Juels, A.: Minimalist cryptography for low-cost RFID tags. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 149–164. Springer, Heidelberg (2005)

Securing Layer-2 Path Selection in Wireless Mesh Networks*

Md. Shariful Islam, Md. Abdul Hamid,
Byung Goo Choi, and Choong Seon Hong**

Department of Computer Engineering, Kyung Hee University, Republic of Korea
{sharif,hamid}@networking.khu.ac.kr,
{bgchoi,cshong}@khu.ac.kr

Abstract. The current draft standard of 802.11s has defined routing for Wireless Mesh Networks (WMNs) in layer-2 and to differentiate from layer-3 routing, it termed layer-2 routing as path selection. The layer-2 path selection (LPS) mechanism is fully specified in the draft of IEEE 802.11s for WMNs. However, routing with security provision is not specified in the standard. Our study identifies that the current path selection mechanism is vulnerable to various types of routing attacks like flooding, route re-direction, spoofing etc. In this paper, we develop a novel Secure Layer-2 Path Selection (SLPS) mechanism that uses cryptographic extensions to provide authenticity and integrity of routing messages. Particularly, the proposed SLPS prevents unauthorized manipulation of mutable fields in the routing messages. Results from analysis and simulation demonstrate that SLPS protocol is robust against identified attacks and provides higher packet delivery ratio, requires no extra communication cost and incurs little path acquisition delay, computational and storage overhead to accomplish secure path selection.

Keywords: Security, Merkle Tree-based Authentication, Layer-2 Routing, Wireless Mesh Networks.

1 Introduction

The area of wireless networks has gained increased importance and development during the past decade. Wireless Mesh Networks (WMN) have emerged as a key technology to support a numerous number of application scenarios like broadband home networking, community and neighborhood networking, enterprise networking, metropolitan area networking, etc. It is a paradigm shift from conventional 802.11 WLAN for its unique characteristics of self-configuring capability, easy network maintenance, lower cost and robustness [1]. Wireless mesh network's infrastructure is, in effect, a router network minus the cabling between nodes. It is built of peer radio devices that do not have to be cabled to

* This research was supported by the MKE under the ITRC support program supervised by the IITA" (IITA-2008-(C1090-0801-0016)).
** Corresponding author.

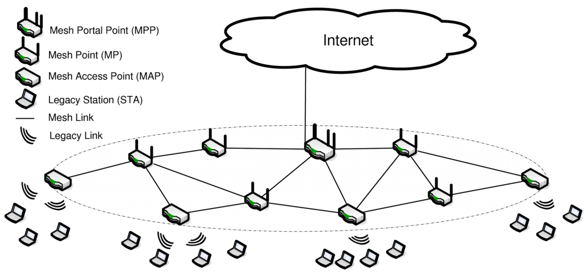


Fig. 1. Network architecture: Wireless Mesh Networks

a wired port like traditional WLAN access points (AP) do and such architecture provides high bandwidth, spectral efficiency, and economic advantage over the coverage area. As a result, the increased interest in WMN has demanded a standard named IEEE 802.11s. The current draft D1.06 [2] of 802.11s is the first to introduce the concept of embedding routing in layer-2. The main motivation that drives incorporating routing in MAC layer is the interoperability between devices of different vendors.

The network architecture of a 802.11s WMN is depicted in Fig. 1. A mesh point (MP) is an IEEE 802.11s entity that can support WLAN mesh services. A mesh access point (MAP) is an MP but can also work as an access point. A mesh portal (MPP) is a logical point that connects the mesh network to other networks such as a traditional 802.11 WLAN or a non-802.11 network. The current 802.11s standard defines secure links between MPs, but it does not provide end-to-end security [3]. Also, the security in routing or forwarding functionality is not specified in 802.11s standard. Routing information elements are transferred in plain text and are prone to various types of routing attacks like flooding, route re-direction, spoofing, etc. The main reason is that intermediate nodes need to modify mutable fields (i.e., hop count, TTL, metric, etc.) in the routing element before forwarding and re-broadcasting them. Since other nodes will act upon those added information, these must also be protected somehow from being forged or modified. However, only source authentication does not solve this problem, because the information is added or modified in intermediate nodes. This motivates us to include hop-by-hop authentication in our proposal. More specifically, each node that adds information in the control packet should authenticate the added information in such a way that each other node acts upon that information should be able to verify its authenticity.

We develop a Secure Layer-2 Path Selection (SLPS) protocol for wireless mesh networks. SLPS takes into consideration the existing key hierarchy of 802.11s, identifies the mutable and non-mutable fields in the routing message, protects the non-mutable part using symmetric encryption and authenticates mutable information using Merkle tree [4]. Particularly, the proposed SLPS prevents unauthorized manipulation of mutable fields in the routing messages. Results from analysis and simulation demonstrate that SLPS protocol is robust against

Table 1. Notations and abbreviations used in this paper

Notation	Description
PREQ	Path request message
PREP	Path reply message
$PREQ/PREP - MF$	Path request or reply message excluding the mutable fields
RANN	Root announcement message
GTK	Group transient key
PTK	Pairwise transient key
MAC	Message authentication code
$MAC_k root(x)$	MAC created on the root of the Merkle Tree using key k by the node x
$h(v)$	A hash created on the mutable field v
$\|\|$	Concatenation of messages
$authpath(v)$	Authentication path for the mutable field v

the identified attacks, provides higher packet delivery ratio, requires no extra communication cost and incurs little computational and storage overhead. Table 1 depicts the notations and abbreviations used throughout the paper.

The rest of the paper is organized as follows. Section 2 discusses related works. Section 3 briefly introduces existing L-2 path selection (LPS) mechanism in 802.11s. Security vulnerabilities of existing path selection mechanism are identified in Section 4. We describe our proposed secure routing (SLPS) in Section 5. Security analysis and performance evaluation are carried out in Section 6 and 7, respectively. Finally, we conclude the paper in Section 8.

2 Related Works

Security is a critical step to deploy and manage WMNs. Since the access to any deployed wireless mesh network is possible for any wireless enabled device, it should be ensured that only authorized users are granted network access. As of now, there is no state-of-the-art solution exists in the literature for securing L2 routing in 802.11s. In [5], the authors have just summarized the proposed routing from 802.11s draft. Ref [6] describes just an update of layer-2 routing in the current draft. A framework for 802.11s and research challenges are summarized in [3]. A hybrid centralized routing protocol is presented in [7] that incorporates tree-based routing with a root-driven routing protocol. Apart from these, there has been some research on securing layer-3 routing. Ariande in [8] ensures a secure on-demand source routing. Authentication is done using TESLA [9], digital signatures and standard MAC. However, as the route request is not authenticated until it reaches the destination, an adversary can initiate route request flooding attack. A variant of Ariande [8] named endairA is proposed in [10] with the exception that instead of signing a route request, intermediate nodes sign the route reply. However, endairA is still vulnerable to malicious route request flooding attack. SAODV [11] is a secure variant of AODV in which operations are similar to AODV, but uses cryptographic extensions to provide authenticity and integrity of routing message. It uses hash chains in order to prevent manipulation of hop count field. However, an adversary may always increase the hop count. Another secure on-demand distant vector protocol, ARAN (Authenticated Routing for Ad hoc Networks), is presented in [12]. Just like SAODV, ARAN uses public key cryptography to ensure integrity of routing message. However, a major drawback of ARAN is that it requires extensive signature generation and verification during the route request flooding which may result in denial-of-service attack.

Our secure path selection scheme employs only symmetric cryptographic primitives and does not assume the existence of pairwise shared key for source-destination. Rather, a Merkle tree-based hop-by-hop authentication mechanism is devised exploiting existing keying hierarchy of 802.11s standard.

3 Path Selection Mechanism in 802.11s

The layer-2 path selection (LPS) mechanism defined in 802.11s is a combination of both reactive and proactive strategy by employing both on-demand path selection mode and proactive tree building mode [2] [5] [6]. The mandatory routing metric used is the airtime cost metric [2] that measures the amount of channel resource consumed by transmitting a frame over a particular link. In LPS mechanism, both on demand and proactive mode can be used simultaneously. In the On-demand mode, a source MP broadcasts *path request* (PREQ) message requesting a route to the destination. The PREQ is processed and forwarded by all intermediate MPs and set up the reverse path from the destination to the source of route discovery. The destination MP or any intermediate MP with a path to the destination may unicast a *path reply* (PREP) to the source MP that creates the forward path to the destination.

Proactive Tree Building mode builds a tree-topology network when a root in the WMN is configured. This topology tree formation begins when the root starts to periodically broadcast a root announcement (RANN) message which propagates the metric information across the network. Upon reception of a RANN message, an MP that wants to create or refresh a path to the root MP sends a unicast PREQ to the root MP. The root MP then unicasts a PREP in response to each PREQ. The unicast PREQ creates the reverse path from the root MP to the originating MP, while the PREP creates the forward path from the MP to the root MP. A root MP may also proactively disseminate a PREQ message to all the MPs in the networks with the intention to establish a path. An MP, after receiving a proactive PREQ, creates or updates its path to the root MP by unicasting a *Proactive* PREP.

Path selection mechanism also allows both on-demand and proactive mode to work simultaneously. This hybrid mode is used in situations where a root MP is configured and a mesh point S wants to send data to another mesh point D but has no path to D in its routing table. Instead of initiating on demand mode, S may send data to the root MP, which in turns delivers the data to D informing that both S and D are in the same mesh. This will trigger an on-demand route discovery between S and D and subsequent data will be forwarded using the new path.

4 Possible Attacks

In this section, we show that LPS, in its normal operation, is vulnerable to the following attacks.

4.1 Flooding Attack

It is very easy for a malicious node to flood the network with a PREQ messages destined to an address which is not present in the network. As the destination node is not present in the network, every intermediate node will keep forwarding the PREQ messages. As a result, a large number of PREQ messages in a short period will consume the network bandwidth and can degrade the overall throughput.

4.2 Route Re-direction Attack

A malicious node M may divert traffic to itself by advertising a route to a destination with a destination sequence number (DSN) greater than the one it received from the destination. For example, the malicious node M in Fig. 2a receives a PREQ from A which was originated from S for a route to node D. As LPS allows intermediate PREP, M may unicast a PREP to A with a higher DSN than the value last advertised by D. So, A will re-direct all subsequent traffic destined for D to the malicious node M.

Route re-direction attack can also be launched by modifying the mutable metric field used in the LPS's PREQ messages. A malicious node can modify the mutable metric field to zero to announce a better path to a destination. As depicted in Fig. 2b, M can decrease the metric field in the PREQ to zero and

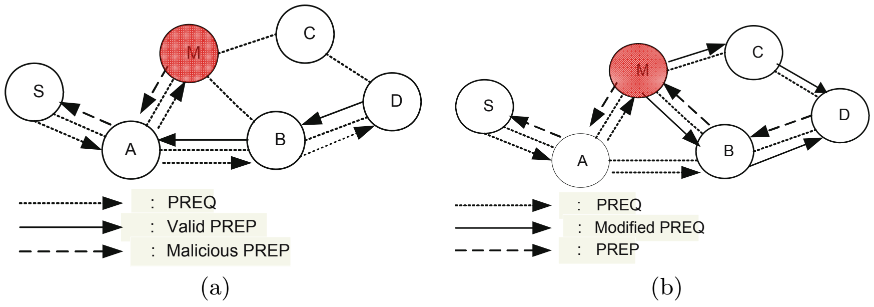


Fig. 2. Route re-direction attack. (a) Increasing DSN. (b) Decreasing metric.

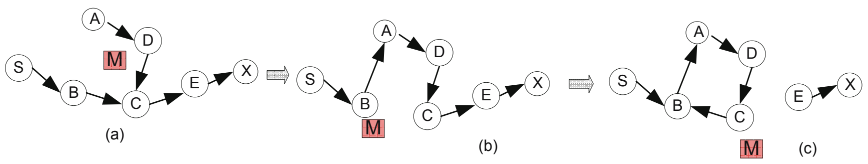


Fig. 3. Routing loops formation

re-broadcasts it to the network. So, the reverse path created should go through the malicious node M. As a result, all traffics to the destination D will be passed through the attacker.

4.3 Routing Loops

A malicious node may create routing loops [12] in a mesh network by spoofing MAC addresses and modifying the value of the metric field. Consider the following network scenarios (Fig. 3) where a path exists between the source S and destination X that goes through node B and C. Also, there exists a path from A to C through D.

Assume that a malicious node M, as shown in Fig. 3a, is present in the vicinity where it can listen to the PREQ/PREP messages that pass through A, B, C and D during route discovery process. It may create a routing loop among the nodes A, B, C and D by impersonation combined with modification of metric field in PREP message. First, it impersonates node A's MAC address and moved out of the reach of node A and closer to node B. And then it sends a PREP message to node B indicating a better metric value then that of the value received from C. So, node B now re-establishes its route to X that should go through A as shown in Fig 3b. At this point, the malicious node impersonates node B and moves closer to node C and sends a PREP to node C indicating a better metric then the one received from E. So, node C will now choose B as the next hop for its route to the destination X as shown in Fig 3c. Thus a loop has been formed and the destination X is unreachable from all the four nodes.

5 Proposed Secure Layer-2 Path Selection (SLPS) Protocol

SLPS, proposed in this section is a secure extension of LPS. As specified in [2], LPS routing information elements have a mutable and a non-mutable part. We exploit these existing mutable and non-mutable fields to design a secure layer-2 path selection. Particularly, SLPS mechanism has four components: (1) key establishment, (2) identifying the mutable and non-mutable fields, (3) showing that the mutable fields can be authenticated in a hop-by-hop fashion using the concept of Merkle tree, and finally (4) protection of non-mutable fields is achieved with the use of symmetric encryption. We present our approach in details in the following subsections.

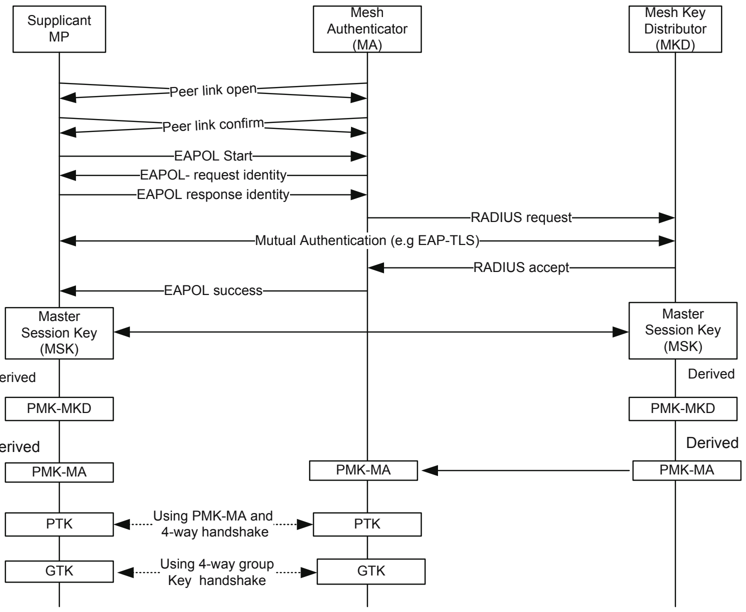


Fig. 4. Key distribution and authentication in 802.11s

5.1 Key Establishment

Entities in a WMN can act both as supplicant MP or mesh authenticator (MA). Before initiating a route discovery process, all the MPs authenticate its neighboring MPs, establish pairwise transient key (PTK), and send the group transient key (GTK) through key distribution and authentication process of 802.11s as depicted in Fig 4. We use this GTK for securing broadcast messages (e.g., PREQ, RANN) and PTK for securing unicast messages (e.g., PREP, proactive PREP).

5.2 Identifying Mutable and Non-mutable Fields

The information elements in the LPS contain mutable fields that are modified in the intermediate routers and non-mutable fields that are not modified in the intermediate routers. The identified mutable fields in the PREQ (Fig. 5) element are:

Hop count field: Provides information on the number of links in the path, incremented by each intermediate node, but it is not used for routing decision.

TTL field: The time-to-live field defines the scope of the PREQ in number of hops. TTL value is decremented by 1 at each intermediate node.

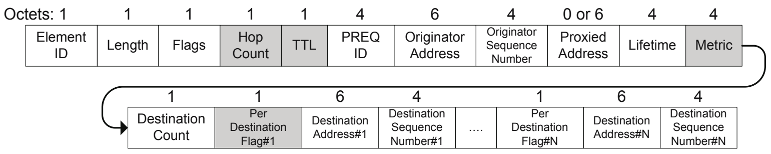


Fig. 5. PREQ message format

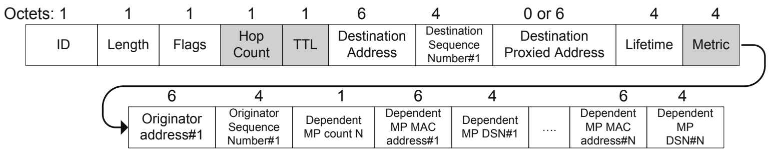


Fig. 6. PREP message format

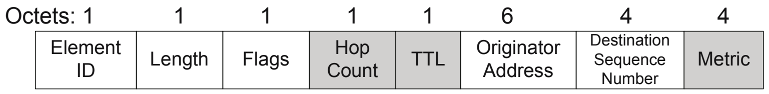


Fig. 7. RANN message format

Metric field: Unlike AODV, LPS uses an airtime link metric instead of hop count metric to take a decision on path selection. Whenever an intermediate node receives a PREQ that is to be forwarded, it calculates the airtime cost in the current path and adds the value to the existing metric field.

Per destination flag: The Destination Only (DO) and Reply and Forward (RF) Flag determine whether the route-reply message (RREP) will be sent by intermediate node or only by the destination. If DO flag is not set and RF flag is set, the first intermediate node that has a path to the destination sends PREP and forwards the PREQ by setting the DO flag to avoid all intermediate MPs sending a PREP. In this case, per destination flag field is also a mutable field. Fig. 6 and Fig. 7 show the format of a PREP and RANN information element. In both the cases, the mutable fields are hop-count, TTL and metric indicated by the shadowed boxes.

5.3 Secure Route Discovery

Securing On demand mode: This mode is used when there is no root MP configured or a root MP is configured, but on demand mode can provide a better path to another MP. Consider a source MP S in Fig. 8 that wants to communicate with a destination MP X.

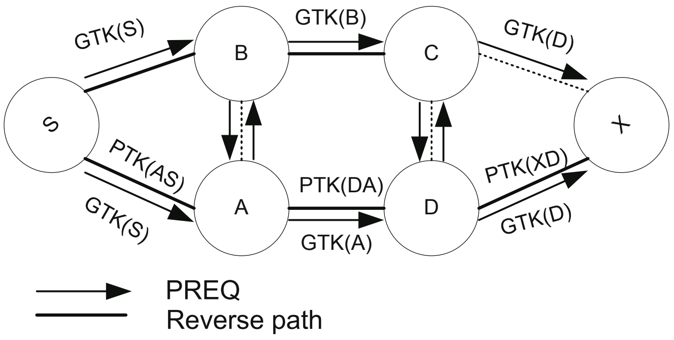


Fig. 8. Secure on-demand path selection

In order to establish a secure route, source node S, destination node X and set of intermediate nodes F_1 that includes $\{A, B\}$ and F_2 that includes $\{C, D\}$ executes the route discovery process in the following way:

$$S \rightarrow \star : MAC_{GTK} root(S), \{v_i, authpath(v_i)\}, \{PREQ - MF\}_{GTK} \quad (1)$$
$$F_1 \rightarrow \star : MAC_{GTK} root(F_1), \{v_i, authpath(v_i)\}, \{PREQ - MF\}_{GTK} \quad (2)$$
$$F_2 \rightarrow \star : MAC_{GTK} root(F_2), \{v_i, authpath(v_i)\}, \{PREQ - MF\}_{GTK} \quad (3)$$
$$X \rightarrow F_2 : MAC_{PTK}^{X,F_2} root(X), \{v_i, authpath(v_i)\}, \{PREP - MF\}_{PTK}^{X,F_2} \quad (4)$$
$$F_2 \rightarrow F_1 : MAC_{PTK}^{F_2,F_1} root(F_2), \{v_i, authpath(v_i)\}, \{PREP - MF\}_{PTK}^{F_2,F_1} \quad (5)$$
$$F_1 \rightarrow S : MAC_{PTK}^{F_1,S} root(F_1), \{v_i, authpath(v_i)\}, \{PREP - MF\}_{PTK}^{F_1,S} \quad (6)$$

The key management of 802.11s ensures that node S is equipped with one GTK that it shares with its neighbors and set of PTKs for communicating with each neighbor individually. Before broadcasting the PREQ, it first creates a Merkle tree with the leaves being the hash of mutable fields of PREQ message. S then creates a MAC on the root of the Merkle tree it just created. Then, S broadcasts message (1) which includes the MAC of the root created using the GTK, mutable fields v_i that need to be authenticated along with the values of its authentication path $authpath(v_i)$ and encrypted PREQ message excluding the mutable fields.

Any of the neighboring nodes of S, after receiving the PREQ, tries to authenticate the mutable fields by hashing the values received in an ordered way, create a MAC on it using the shared GTK and comparing that with the received MAC value of the root. If the two values match, the intermediate MP is ascertain that the values are authentic and come from the same source that created the tree.

Let us consider, for example, that B receives a PREQ from its neighboring node S and wants to authenticate the value of the metric field M as shown in Fig. 9. According to our protocol, B and C should receive the value M along with the values of the authentication path of M in the Merkle tree such as U_H and U_{TF}. B and C can now verify the authenticity of M by computing $h(h(h(M)||U_H)||U_{TF})$ and a MAC on this value using the key GTK. It then

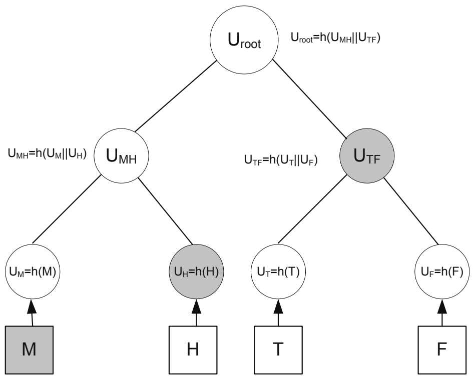


Fig. 9. Authentication path for the metric field in a Merkle Tree

compares the received MAC value with the new one, if it found a match, then it can assure that the value M is authentic and came from the same entity that has created the tree and computed the MAC on the U_{root}.

The intermediate nodes then update the values of the mutable fields (e.g., hop count, metric and TTL) and create Merkle trees from the modified fields. They also decrypt the non-mutable part of the PREQ message and re-encrypt it with their own broadcast key and re-broadcast (as shown in (2) and (3)) the PREQ message following the same principle. After receiving the PREQ, the destination MP updates the mutable fields; creates its own Merkle tree and unicasts a PREP message as in (4) using the same principle but this time it uses PTK instead of GTK. The PREP is propagated as (5) and (6) to the source MP in the reverse path created using PREQ and thus a secure forward path from the source to the destination is established.

Securing Proactive Mode: To accomplish security in proactive mode, we need to employ security in both Proactive RANN and Proactive PREQ mechanism.

In the *Proactive RANN* mode, the RANN message is broadcasted using the group transient key as shown in Eq. (7) - (9) to protect the non-mutable fields and authenticate the mutable fields (hop count, TTL and metric) using the Merkle tree approach. As there are only three mutable fields in the RANN message, a node requires generating a random number to construct the Merkle tree. After receiving the RANN message an MP that needs to setup a path to the root MP unicasts a PREQ to the root MP as per Eq. (10) - (12). Upon receiving each PREQ, the root MP replies with a unicast PREP to that node as described in Eq. (13) - (15).

$$R \rightarrow \star : MAC_{GTK}root(R), \{v_i, authpath(v_i)\}, \{RANN - MF\}_{GTK} \quad (7)$$

$$F_1 \rightarrow \star : MAC_{GTK}root(F_1), \{v_i, authpath(v_i)\}, \{RANN - MF\}_{GTK} \quad (8)$$

$$F_2 \rightarrow \star : MAC_{GTK}root(F_2), \{v_i, authpath(v_i)\}, \{RANN - MF\}_{GTK} \quad (9)$$

$$D \rightarrow F_2 : MAC^{D,F_2}_{PTK}root(D), \{v_i, authpath(v_i)\}, \{PREQ - MF\}^{D,F_2}_{PTK} \quad (10)$$

$$F_2 \rightarrow F_1 : MAC_{PTK}^{F_2,F_1} root(F_2), \{v_i, authpath(v_i)\}, \{PREQ - MF\}_{PTK}^{F_2,F_1} \quad (11)$$
$$F_1 \rightarrow R : MAC_{PTK}^{F_1,R} root(F_1), \{v_i, authpath(v_i)\}, \{PREQ - MF\}_{PTK}^{F_1,R} \quad (12)$$
$$R \rightarrow F_1 : MAC_{PTK}^{R,F_1} root(R), \{v_i, authpath(v_i)\}, \{PREP - MF\}_{PTK}^{R,F_1} \quad (13)$$
$$F_1 \rightarrow F_2 : MAC_{PTK}^{F_1,F_2} root(F_1), \{v_i, authpath(v_i)\}, \{PREP - MF\}_{PTK}^{F_1,F_2} \quad (14)$$
$$F_2 \rightarrow D : MAC_{PTK}^{F_2,D} root(F_2), \{v_i, authpath(v_i)\}, \{PREP - MF\}_{PTK}^{F_2,D} \quad (15)$$

Notations used in Eq. (7) - (15) are as follows. R is considered as the root MP and D is the MP that needs to setup a path to R. F_1 and F_2 are the intermediate nodes in the path. $MAC_k root(X)$ represents the MAC of the Merkle tree's root created by the node X using a shared key k. $RANN/PREQ/PREP - MF$ represents the routing information elements without the mutable fields. v_i and $authpath(v_i)$ denote the fields needed to be authenticated and the values assigned to the authentication path from v_i to the root of the tree, respectively.

Proactive PREQ mechanism is used to create paths between the root MP and the remaining MPs in the network proactively. Only the MP that is configured as a root MP would send proactive PREQ messages periodically. The proactive PREQ is also needed to be broadcasted to the MPs attached to the root MP encrypted using GTK of the root MP, the mutable fields (TTL, hop count, metric and per destination flag) of this PREQ should be authenticated in the intermediate MPs in the same way as discussed earlier. If an MP needs to update its path to the root MP, it unicasts a proactive PREP to the root MP. PREP is transmitted securely as in the case of PREP in on-demand mode.

6 Security and Overhead Analyses

In this section, we will analyze the proposed SLPS in terms of robustness against the attacks presented in Section 4 and also the overhead required for ensuring secure routing.

6.1 Security Analysis

Preventing Flooding Attack: In the proposed SLPS, a node can participate in the route discovery process only if it has successfully established a GTK and PTK through key distribution and authentication mechanism of 802.11s. Thus, it will not be possible for a malicious node to initiate a route discovery process with a destination address that is not in the network. Again, as the PREQ message is encrypted during transmission, a malicious node cannot insert new destination address.

Preventing Route Re-direction Attacks: The root cause of route re-direction attacks are modification of mutable fields in routing messages. These mutable fields are authenticated in each hop. If any malicious node modifies the value of a field in transit, it will be readily detected by the next hop while comparing the new MAC with the received one. It will find a miss-match in comparing the MACs and modified packet will be discarded.

Avoiding Formation of Routing Loops: Formation of routing loops requires gaining information regarding network topology, spoofing and alteration of routing message. As all the routing information is encrypted between nodes, an adversary will be unable to learn network topology by overhearing routing messages. Spoofing will not benefit the adversary as it will require authentication and key establishment to transmit a message with spoofed MAC. Moreover, fabrication of routing messages is detected by integrity check. So, the proposed mechanism ensures that routing loops cannot be formed.

6.2 Overhead Analysis

Computational Overhead: The computational overhead of a sender and a receiver can be given by:

$$\langle k \times h \rangle + m + e \text{ (sender)} \tag{16}$$

$$\langle a + 1 \rangle h + m + d \text{ (receiver)} \tag{17}$$

where, k is the number of hash operations required to form a Merkle tree. Cost of computing a hash function is defined by h, m is the cost involved in computing the MAC of the root, whereas e and d are encryption and decryption cost. To authenticate a particular value, a receiver need to compute the root by calculating $(a + 1)$ hash operations, where a is the number of nodes in the authentication path.

Communication Overhead: It is defined by the number of routing messages required to establish a secure path and given by Eq. (18), (19) and (20),

$$(n-1) \times \mathit{broadcast} + h \times \mathit{unicast} \text{ (on-demand)} \tag{18}$$

$$n \times \mathit{broadcast} + h \times \mathit{unicast} \text{ (proactive PREQ)} \tag{19}$$

$$n \times \mathit{broadcast} + 2h \times \mathit{unicast} \text{ (proactive RANN)} \tag{20}$$

where, n is the number of nodes in the network, h is the number of hops in the shortest path. The number of messages required for establishing a path in LPS is same as our proposed one. So, our protocol does not incur any extra communication overhead.

Storage Requirements: A node needs to store the number of fields that need to be authenticated, hashed values of the Merkle tree and the MAC of the root value. So, storage requirement of a node is given by Eq. (21).

Table 2. Overhead Comparison

	Computation			Communication		Storage (bytes)
	Hash	MAC	Enc/Dec	Unicast	Broadcast	
LPS	0	0	0	h	n	20
SLPS	k	1	1	h	n	47

$$\sum_{i=1}^{n} d_i + (k \times l) + S_M \tag{21}$$

where, d_i is the size of a mutable field, k is the number of hashes in the Merkle tree, l is the size of a hashed value and S_M is the size of the MAC.

Table 2 summarizes the computation, communication and storage overheads required by a particular sender/receiver for both LPS and SLPS schemes. It shows that though the computation and storage requirements for the secure mechanism are slightly higher than the non-secure LPS scheme, the proposed SLPS scheme does not incur any extra communication overhead.

7 Performance Evaluation

In this section, we evaluate the performance of the proposed SLPS scheme. We use ns-2 [13] to simulate our proposed secure path selection approach and compare that with existing LPS. We have simulated 50 nodes in a 1500 x1500 m^2 area. We use 5 to 10 distinct source destinations pairs that are selected randomly. Traffic source are CBR (constant bit-rate). Each source sends data packets of 512 bytes at the rate of four packets per second during the simulation period of 900 seconds. The Performance metrics that considered are: i) **Packet delivery ratio**: Ratio of the number of data packets received at the destinations to the number of data packets generated by the CBR source, ii) **Control overhead (in bytes)**: Ratio of the control overhead to the delivered data, iii) **end-to-end delay** for data packets, and iv) **path acquisition delay**, which is the time requires to establish a route for a source-destination pair.

We assume that there are 10 nodes that are misbehaving and take part in the route discovery process and drop packets that they should forward. Since, in our secure routing approach, misbehaving nodes can not participate in the route

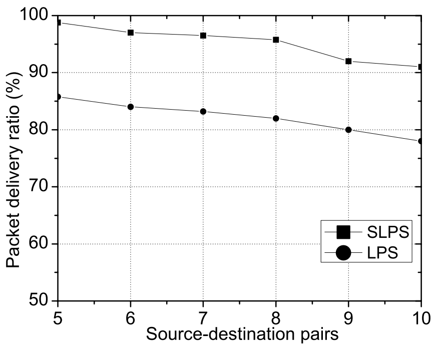


Fig. 10. Packet delivery ratio

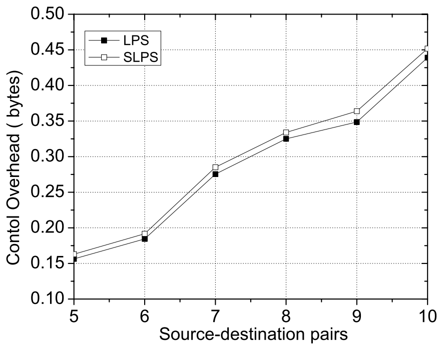


Fig. 11. Control packet overhead

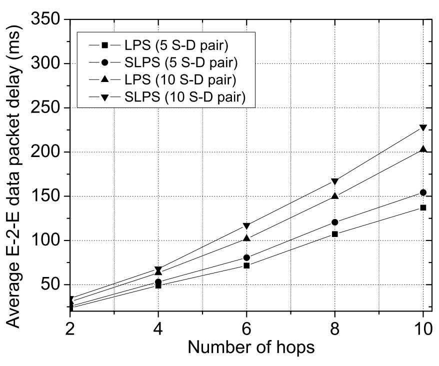


Fig. 12. End-to-end delay for data

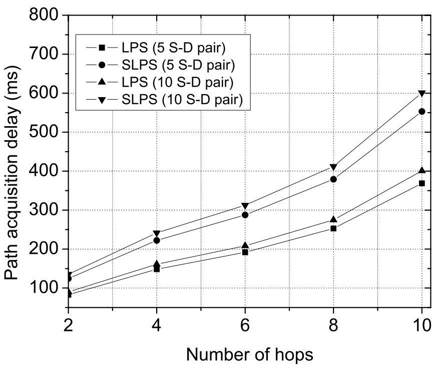


Fig. 13. Path Acquisition Delay

discovery process and as a result it gives a better packet delivery ratio as shown in Fig. 10. On the other hand, though the number of control messages required to transmit for a route establishment is roughly the same, due to the addition of a MAC value, control packet overhead of the proposed scheme is slightly higher than the LPS scheme as shown in Fig. 11. But, this control overhead ensures higher security.

Fig. 12 depicts that the average end-to-end delay of data packets for both protocols are almost equal. So, it is also evident that the effect of route acquisition delay on average end-to-end delay is not significant. Average route acquisition delay for the proposed SLPS scheme is much higher than that of the LPS mechanism as shown in Fig. 13. Because, in addition to normal routing operation of LPS, the proposed SLPS scheme requires computing hash and MACs values which require extra processing delay.

8 Conclusions

In this paper, we devise a secure path selection mechanism for wireless mesh networks that is gradually maturing to a point where it cannot be ignored when considering various wireless networking technologies for deployment. We have proposed SLPS, a secure extension of layer-2 routing specified in 802.11s. Our proposed mechanism takes into consideration the existing key hierarchy of 802.11s (so, there is no extra keying burden), identifies the mutable and non-mutable fields in the routing message, protects the non-mutable part using symmetric encryption and uses Merkle-tree approach to authenticate mutable information. We have shown that our protocol is robust against identified attacks and computationally efficient as it uses only symmetric key operations.

References

1. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. Computer Networks 47(4), 445–487 (2005)
2. IEEE 802.11s Task Group, Draft Amendment to Standard for Information technology-Telecommunications and Information Exchange Between Systems–Local and metropolitan area networks-Specific requirements - Part 11: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment IEEE p802.11s/d1.06: Mesh Networking (July 2007)
3. Wang, X., Lim, A.O.: IEEE 802.11s wireless mesh networks: Framework and challenges. Ad Hoc Networks 6, 970–984 (2008)
4. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
5. Bahr, M.: Proposed routing for IEEE 802.11s wlan mesh networks. In: WICON 2006: Proceedings of the 2nd annual international workshop on Wireless internet, p. 5. ACM, New York (2006)
6. Bahr, M.: Update on the hybrid wireless mesh protocol of IEEE 802.11s. In: IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, MASS 2007, pp. 1–6 (2007)
7. Lim, A.O., Wang, X., Kado, Y., Zhang, B.: A hybrid centralized routing protocol for 802.11s wmns. Mob. Netw. Appl. 13(1), 117–131 (2008)
8. Hu, Y.C., Perrig, A., Johnson, D.B.: Ariadne: a secure on-demand routing protocol for ad hoc networks. Wirel. Netw. 11(1-2), 21–38 (2005)
9. Perrig, A., Tygar, J.D., Song, D., Canetti, R.: Efficient authentication and signing of multicast streams over lossy channels. In: SP 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy, p. 56. IEEE Computer Society, Washington (2000)
10. Ács, G., Buttyán, L., Vajda, I.: Provably secure on-demand source routing in mobile ad hoc networks. IEEE Trans. Mob. Comput. 5(11), 1533–1546 (2006)
11. Zapata, M.G., Asokan, N.: Securing ad hoc routing protocols. In: WiSE 2002: Proceedings of the 1st ACM workshop on Wireless security, pp. 1–10. ACM, New York (2002)
12. Sanzgiri, K., Dahill, B., Levine, B.N., Shields, C., Belding-Royer, E.M.: A secure routing protocol for ad hoc networks. In: ICNP 2002: Proceedings of the 10th IEEE International Conference on Network Protocols, pp. 78–89. IEEE Computer Society, Washington (2002)
13. Information Sciences Institute: NS-2 network simulator. Software Package (2003), http://www.isi.edu/nsnam/ns/

Public Key Authentication with Memory Tokens

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

[1] Hitachi, Ltd., Systems Development Laboratory
{camille.vuillaume.ch,katsuyuki.okeya.ue}@hitachi.com
[2] Technische Universität Darmstadt
{dahmen,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. We propose a very low-cost authentication scheme based on Merkle signatures, which does not require any computation on the prover side, but instead, has moderate memory requirements. Our technique is particularly attractive on platforms where memory is already available, since it can be implemented at practically no cost, without any CPU, and with an extremely simple memory access control mechanism.

Keywords: Merkle signatures, authentication, low-cost implementation.

1 Introduction

One of the major contributions of cryptography is its potential for proving our identity, for example during electronic transactions. Many solutions are known to this problem: using symmetric key cryptography like HMAC, or using public key cryptography such as the Fiat-Shamir identification scheme [1] or any digital signature, RSA for instance. All known authentication techniques share a common characteristic: in order to issue a proof of his identity, the prover has perform some calculations. In lightweight schemes, these calculations might be a hash computation or a single modular multiplication, but in other cases, amount to a costly exponentiation over a finite field. Intuitively, it seems that such computation is required, because in a challenge-response protocol, the prover has to construct the proof of his identity adaptively, according to a challenge provided by the verifier. In this paper, we attempt to answer the following question: is it possible to design an authentication (or better, an identification) scheme using simple memory manipulations such as read and write accesses? This is motivated by the fact that (1) the prices of memory are in free fall, (2) in many cases, memory is available but computational resources are not. Think for example of RFID tags, which only accommodate for memory operations, including read, write and password-protected access [2], USB and flash memory cards, or game cartridges based on solid state technology to cite a few. In these cases, computational resources are at best extremely limited or even nonexistent, and yet authentication mechanisms are highly desirable.

Our contribution in this paper is a public key authentication scheme, with all its usual benefits, such as flexibility, scalability, decentralized architecture and revocation mechanisms, and with the additional feature that the prover does not perform any computation but merely data selection. As a result, the prover can be perfectly instantiated by a simple memory token. Verification does involve the calculation of a few hashes, but its computational cost is order of magnitudes smaller than typical public key cryptosystems.

Our scheme is based on the observation that in some settings of the Merkle signature scheme [3], signing boils down to data selection; to the best of our knowledge, this fact is not mentioned in the literature, perhaps because the resulting memory consumption makes the scheme less attractive than the case where computations are actually performed. We argue that with low prices of flash memory, things are not that clear, and further mitigate the issue of memory consumption by limiting the scope of our scheme to a challenge-response authentication protocol, where we allow short challenges and use hash functions with a small output size. The security assumptions are extremely weak since we only need a secure hash function: no random oracle or number-theoretic assumptions are required. Collision resistance yields provable security but in practice we suggest that second-preimage resistance alone is enough.

Unlike typical cryptographic implementations, our scheme does not need any tamper-resistant calculation unit, and as a consequence, is naturally immune to environmental attacks such as side-channel attacks [4]. However, tamper-resistant memory is required for storing secret data; more precisely, security is guaranteed as long as at most half of the secret values are revealed. Tamper-resistance can be implemented in the shape of protective coating and a memory controller that allows only specific memory accesses; destructive reading will satisfy our security requirements. Unlike cryptographic engines which occupy a large silicon area, our proposed design for the memory controller is extremely simple. Thanks to its very low hardware cost, our scheme is especially meaningful when memory is already available; in this case, cryptographic functionalities can be provided for almost free.

2 Authentication Techniques

First, we describe authentication techniques and Merkle signatures, and explain how they can be useful for authentication.

2.1 Approaches for Authentication

Symmetric Key Authentication. There exist low-cost chips capable of performing symmetric key authentication using onboard cryptographic resources such as block ciphers, stream ciphers or hash functions. In order to decrease hardware costs, these dedicated cryptographic units often use custom cryptographic algorithms, sometimes with disastrous consequences for security [5]. In addition to possible cryptographic flaws, the use of symmetric key primitives is inherently limited in terms of system architecture. For instance, if all devices share the same secret key, reverse-engineering one device will result in a system-wide security breach. Alternatively, one might attempt to use different keys, but the verification system still has to store all keys. Putting scalability issues aside, again, this is a single point of failure: if one verification device is compromised, so is the whole system.

One possible way to mitigate this problem is to store all keys in a remote database for verification. We emphasize that the database is still a single point of failure, and merely changing the location of the keys does not really solve the problem. In addition, in a large system, such database should have the ability to process a huge number of authentications in a very short time: think for example of ticket gates in a train station, where passengers expect to go through the gates without even slowing down their pace. As a consequence, the database should not only be reliable, constantly available and secure, but queries should be processed almost instantly. Naturally, one cannot expect such system to be cheap, and perhaps not well-suited for low-cost transactions such as transportation tickets or small retail.

Public Key Cryptography. On the other hand, with a public key-based authentication scheme (or identification scheme following the taxonomy introduced by Fiat and Shamir [1]), it is easy to have different keys for all authentication tokens and yet no connection to the network is necessary. More precisely, we may assume that tokens store their private key, public key and a certificate thereof. As a consequence, the verification system only needs to know the public key of the certification authority; everything can be done offline. Finally, in the event where one authentication token is compromised, its public key can be broadcasted to all verification systems in a black list, thereby limiting the impact of compromised systems. It is noteworthy that this scenario does not assume a persistent network connection, but only periodic updates.

So why is it that public key cryptosystems are not used in practice for low-cost authentication? The reason is quite simple: they are too expensive. For instance, using advanced software engineering techniques on an ATMega128 chip running at 8MHz, ECC scalar multiplication has been reported to take 0.8s and RSA signatures 11s; timings are even slower on a 8051-based CPU running at 14.7456MHz [6]. It can be seen that even with a full-featured CPU that occupies much more silicon area than a dedicated symmetric key cryptographic engine, and therefore for a higher price, timings are still too slow for many applications where users are not willing to wait. Therefore, the only way to have a practical public key cryptography engine is to use a CPU with enough horsepower or select a platform with hardware acceleration for public key operations, at the expense of production cost.

2.2 Merkle Signatures

One-Time Signatures. Essentially, Merkle signatures transform the Lamport one-time signature scheme [7] in a multi-time multi-signature scheme by providing an authentication path from one-time public keys to a single “master” public key. First, we describe the Lamport one-time signature scheme in the context of

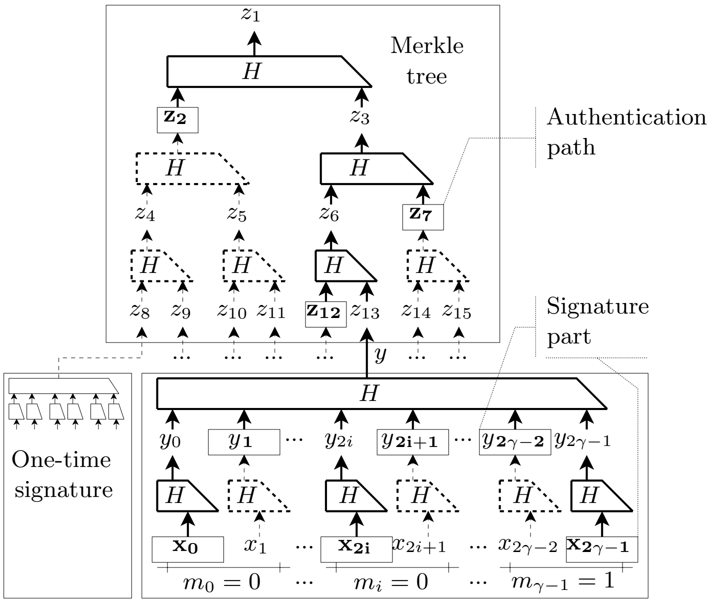

z_1
Merkle tree
H
$\mathbf{z_2}$
z_3
Authentication path
z_4 z_5 z_6 $\mathbf{z_7}$
z_8 z_9 z_{10} z_{11} $\mathbf{z_{12}}$ z_{13} z_{14} z_{15}
Signature part
y
One-time signature
y_0 $\mathbf{y_1}$ y_{2i} $\mathbf{y_{2i+1}}$ $\mathbf{y_{2\gamma-2}}$ $y_{2\gamma-1}$
$\mathbf{x_0}$ x_1 $\mathbf{x_{2i}}$ x_{2i+1} $x_{2\gamma-2}$ $\mathbf{x_{2\gamma-1}}$
$m_0 = 0$ $m_i = 0$ $m_{\gamma-1} = 1$

Fig. 1. Lamport one-time signature combined with Merkle tree

a 128-bit hash function H and a message size of γ bits. The scheme has a 128-bit public key y and 2γ secret values $x_0, \ldots, x_{2\gamma-1}$ of 128 bits each. To generate the keys, the 2γ secret values are randomly generated, and their images $y_i = H(x_i)$ are computed using the hash function H. Next, the 2γ images are concatenated and hashed again; the resulting value is the public key y.

To sign one message bit m_i, one proceeds as follows. If $m_i = 0$, the signature s_{2i}, s_{2i+1} is x_{2i}, y_{2i+1}, whereas if $m_i = 1$, the signature s_{2i}, s_{2i+1} is y_{2i}, x_{2i+1}. Since in any case, only one of the two secrets x_{2i}, x_{2i+1} is revealed, forgeries require inverting the hash function H in order to find the missing secret value, or finding new secret values mapping to the same public key y. In the first case, the one-wayness property of H is violated, and in the second case, the collision-resistance of H is violated [8].

A signature is verified by re-constructing the one-time public key y from the signature data. More precisely, given the signature s_{2i}, s_{2i+1} of message bit m_i, one proceeds as follows. If $m_i = 0$, the two values y'_{2i}, y'_{2i+1} are $H(s_{2i}), s_{2i+1}$, whereas if $m_i = 1$, the two values y'_{2i}, y'_{2i+1} are $s_{2i}, H(s_{2i+1})$. After repeating this operation for all message bits, the obtained values are concatenated and hashed; if the final hash value y' matches the one-time public key y, the signature is accepted.

Merkle Tree. Although Lamport signatures are attractive in terms of efficiency and security because they rely on hash functions only, they have a major limitation which makes them impractical: their security is guaranteed as long as one keypair is used for signing one message. Merkle signatures address this problem using an *authentication tree*, that is, a binary hash tree mapping a sequence of σ one-time public keys to a single "master" public key, root of the hash tree. As a consequence, the scheme can sign up to σ messages.

The upper part of Figure 1 illustrates how the Merkle authentication tree works in the case where eight signatures can be signed. First, eight one-time keypairs with respective public keys z_8, z_9, $\ldots$, z_{15} are generated. Next, pairs of consecutive one-time public keys are concatenated and hashed, obtaining the four parent nodes z_4, z_5, z_6 and z_7. The same operation is repeated until only one node remains, namely z_1, root of the binary tree and public key of the system. Note that with this numbering, the children of node z_i are z_{2i} and z_{2i+1}.

During the verification of the third signature, using the one-time signature scheme with public key z_{13}, node z_{13} is re-calculated in the process. The role of the Merkle authentication tree is to "connect" node z_{13} with the root z_1. This is realized by providing an *authentication path* in the signature. For example, in the case of z_{13}, the authentication path consists of z_{12}, z_7 and z_2. Indeed, from z_{12} and z_{13}, one obtains $z_6 = H(z_{12}||z_{13})$. Similarly, z_6 can be combined with the authentication node z_7 in order to calculate $z_3 = H(z_6\|z_7)$, and finally, from z_2 and z_3, the root $z_1 = H(z_2\|z_3)$ can be re-computed.

In terms of security, a forger against the Merkle signature scheme either forges a one-time signature, or finds a collision in the Merkle tree for the hash function H [8]. Again, this violates security assumptions on the hash function (preimage resistance and collision resistance), and therefore the scheme is secure (in fact, existentially unforgeable under adaptive chosen message attacks [8]).

Authentication with Merkle Signatures: a First Attempt. Merkle signatures could be used for realizing a public key authentication scheme, where the prover signs a random challenge generated by the verifier. At first sight, it appears that such authentication scheme has the same hardware requirements as a symmetric key authentication scheme such as HMAC. However, this is not true because the generation of one-time signatures and the calculation of authentication paths involves hundreds of hash computations and requires a relatively large workspace, even with advanced techniques [3,9]. Therefore, in practice, Merkle signatures suffer from the same limitations as usual public key cryptosystems when it comes to low-cost authentication.

3 Authentication without Calculations

Next, we describe our authentication scheme based on Merkle signatures, including its properties and advantages over existing techniques. After that, we will discuss its security in view of cryptographic and physical attacks.

3.1 Generating Signatures without Calculations

The Lamport one-time signature scheme [7] has one property that is often ignored: with adequate settings, the signing step does not involve *any* calculations.

Recall that if $m_i = 0$, the signature is x_{2i}, y_{2i+1}, and if $m_i = 1$, the signature is y_{2i}, x_{2i+1}; signing boils down to data selection. Also, the *only* secret data in the Lamport scheme is the list of $x_0, \dots x_{2\gamma-1}$ since the hash values $y_0, \dots, y_{2\gamma-1}$ are all re-computed during the verification step.

From the above remarks, we proposed the following design for the Lamport scheme. On the one hand, secret values $x_0, \dots, x_{2\gamma-1}$ are stored in tamper-resistant memory; access to this memory area is limited as follows: only one element in the pair x_{2i}, x_{2i+1} can be accessed. For instance, a memory controller, taking one message bit m_i as input and storing a counter i, could return x_{2i+m_i}, increment i and erase both of x_{2i} and x_{2i+1}, thereby preventing illegal accesses at later time. On the other hand, the hash values $y_0, \dots y_{2\gamma-1}$ are public and can be freely accessed (the memory does not need to be tamper-resistant).

Of course, the same idea can be applied to the Merkle signature scheme. In addition to the secret values and hash values, the Merkle signature scheme requires the storage of all nodes of the binary hash tree $z_1, \dots, z_{2\sigma-1}$, which are all public, just the like hash values y_i. Therefore, for a 128-bit hash function, γ-bit messages and a total of σ signatures, memory requirements are $128 * 2\gamma * \sigma$ bits of tamper-resistant memory with access control for secret values, $128 * 2\gamma * \sigma$ bits (of possibly insecure memory) for their hash values and $128 * (2\sigma - 1)$ bits (of possibly insecure memory) for tree values. In total, a little bit more than $128 * 4 * \gamma * \sigma$ bits of memory are required for signing σ messages of γ bits each.

Signing and Verifying. As explained above, on input m_i, the tamper-resistant memory returns x_{2i+m_i}, increments the locally stored counter i and optionally, erase both of x_{2i} and x_{2i+1} in order to prevent future illegal accesses. In addition to x_{2i+m_i}, the hash value $y_{2i+\bar{m}_i}$ must be sent as well; however, since hash values are not secret, the verifier may freely access the memory area where hash values are stored and get the required data by himself. This procedure is repeated for all message bits $m_0, \dots, m_{\gamma-1}$.

After that, the authentication path must be provided as well. In the usual Merkle signature scheme, the one-time public key of the signature with index j is stored in node z_k with $k = \sigma + j$. If k is even, the first authentication node has index $k + 1$, otherwise it has index $k - 1$. The next authentication node is the

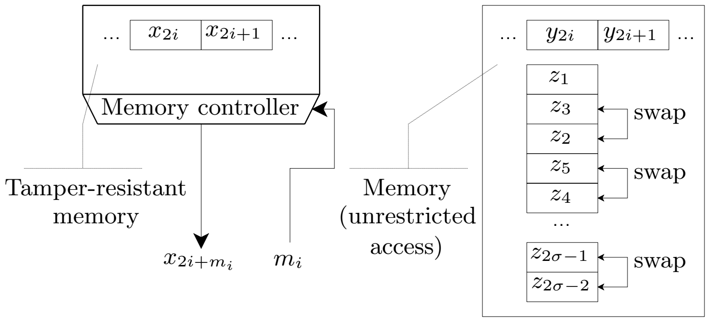


Fig. 2. Memory Units for Merkle Signatures without Calculations

sibling of the parent node, which has index $\lfloor k/2 \rfloor$, and again, the index of the sibling depends on the parity of $\lfloor k/2 \rfloor$. This procedure is iterated until index 1 (that is, root of the tree) is reached after successive divisions by two. Although quite simple, this algorithm can be further simplified if elements in the tree table are swapped as depicted in Figure 2. Indeed, in a table with swapped elements, the first authentication node is z_k with $k = \sigma + j$, the next one has index $\lfloor k/2 \rfloor$ (where the division by two is a simple right shift), and so on until $k = 1$. Just like in the case of hash values, authentication nodes are not secret and can be freely accessed by the verifier, who could be responsible for extracting the correct node from the table. Alternatively, and especially if table elements are swapped, a memory controller could select and send authentication nodes to the verifier.

Unlike signature generation, verification is not free since it involves (a small number of) hash computations: γ hashes for computing the missing hash values y_{2i+m_i} from signature blocks x_{2i+m_i}, one hash computation (over 2γ blocks) for re-computing the one-time public key and another $\lceil \log_2 \sigma \rceil$ hashes for recomputing the root z_1 from authentication nodes. Assuming an iterated hash function with 128-bit block size, this corresponds to $3\gamma + \lceil \log_2 \sigma \rceil$ hash iterations in total.

3.2 Authentication Protocol

Compared to authentication systems based on symmetric key primitives, public key authentication with Merkle signatures has the important advantage that different provers may have different secret keys and yet any verifier can authenticate any prover using the public key of a certificate authority. In particular, a global database is unnecessary. In addition, the prover does not perform any computation, and as a consequence our protocol is well-suited for low-cost and low-power platforms such as RFID tags. The drawback of this design is that moderate storage capacities are required.

Figure 3 illustrates the authentication protocol. The prover sends its public key, the certificate for its public key and an index for the signature scheme selected among indices that have not been used before. The verifier checks the validity of the public key with the certificate and sends a random challenge. The

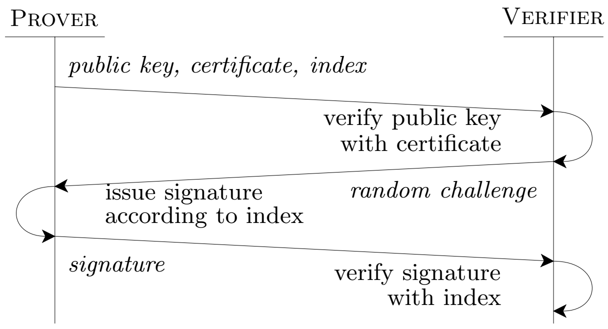

PROVER
VERIFIER
public key, certificate, index
verify public key with certificate
random challenge
issue signature according to index
signature
verify signature with index

Fig. 3. Authentication Protocol

prover generates a Merkle signature for this challenge, which is verified in the last step using the index selected in the first step.

Attack Model. In our attack model, a successful forger may output *any* valid signature, including a signature that has been queried before. The only constraint is that the attacker is not allowed to query a valid system in the runtime; i.e. he can perform queries and interact with legitimate provers, but once the authentication protocol is started, no further query is allowed. This scenario is sound in an authentication system, where attackers can obviously authenticate if they have access to a legitimate authentication token. For example, in anti-counterfeiting systems, one expects the counterfeit to be sold at a lower price, and therefore cannot include (i.e. query) the legitimate system.

Security. If the forger outputs a signature that has *not* been queried, this signature is an existential forgery. This case is ruled by the fact that if the underlying hash function is secure, Merkle signatures are existentially unforgeable under adaptive chosen message attack [8]. On the other hand, it might be that by chance, the forger has already queried the same challenge to a legitimate system. It remains to upper-bound the probability of this case. Since the forger can query a particular system with a given index only once (because the one-time signature scheme associated with the index can be used only once), for this particular index and this particular public key, the forger knows the signature of one challenge only. The security of our scheme stems from the fact that in the authentication protocol, the prover has to commit to the index and the public key *before* knowing the challenge, and as a consequence, for a γ-bit challenge, the probability that the forger knows the corresponding signature is $2^{-\gamma}$.

At this point, it is necessary to put things in their context. Although when γ is small, say 8 or 16 bits, the probability $2^{-\gamma}$ seems to be high from a cryptographic perspective, the probability remains the same independently from how many queries the attacker performs, and independently from how much computational power he has (that is, unless the attacker can break the hash function, which we assume to be intractable). In many practical cases, one can live with the fact that a lucky attacker will be successful from time to time. For instance, verifiers could run the protocol only once in a specified time frame, allow only a certain number of failures from a given prover and black-list suspicious provers. A similar argument was given by Fiat and Shamir, who suggest that a security level of 2^{-20} (corresponding to 20-bit challenges) is enough for authentication [1]; however, we believe that in some applications, and especially when the goal is simply to give a hard time to cheaters, 16 or even 8 bits of security are sufficient. Indeed, security is always about trade-offs: despite the fact that one can win at Russian roulette with the significant probability of 5/6, no sane person would play the game. Similarly, a short challenge size is perfectly reasonable in many (but not all) cases. Therefore, we argue that the size of the challenge should be chosen according to what security level is acceptable in practice, but also to hardware requirements.

Hash Function Security. It has been established that a forgery of the Merkle signature scheme implies either a collision or a preimage for the underlying hash function [8], which means that breaking the Merkle signature scheme is harder than finding a collision. On the one hand, several attacks against the collision resistance property of popular hash functions have been published [10]. On the other hand, it seems unlikely that breaking the Merkle signature scheme is exactly as hard as finding a collision, and it has been suggested (without security proof) that second-preimage resistance is relevant to the security of the scheme [11]. Indeed, a collision finding algorithm seems useless for forging signatures; in fact, even a second-preimage does not seem to be sufficient, but a preimage certainly is. Moreover, there exist alternative approaches for building provably secure Merkle signature schemes that do not require a collision resistant hash function, but instead, a target-collision resistant [12] or even a second-preimage resistant hash function [13]. These are the reason why throughout this paper, we assume a 128-bit hash function such as MD5 (for which no second-preimage or preimage attack is known) or AES in a suitable PGV mode.

Physical Security. We previously wrote that tamper-resistant memory is required (for storing secret values only). This is in stark contrast with other cryptosystems where computations must be secure. In our case, there are no computations, but access to memory must be restricted. Such tamper-resistant memory should be easier to produce than a tamper-resistant CPU or a cryptographic engine (which include tamper-resistant memory anyway); in particular, side-channel attacks [4] are out of scope.

3.3 Comparison

Using our design, one can benefit from a public key authentication scheme where no calculations are performed by the prover. Compared to the naive approach of storing the output values of a one-way function, which has been a commonly used method for authenticating users based on one-time passwords, Merkle signatures have several important advantages. First, Merkle signatures belong to the realm of public key cryptography, which implies that, among others, any verifier can authenticate any prover without sharing any secret. Second, when an authentication token stores a sequence of one-time passwords, such token can easily cloned by querying all possible passwords, whereas this approach is impossible with Merkle signatures: even after querying many signatures, the probability to forge authentication remains unaffected. Alternatively, the token may store different possible one-time passwords, reveal just one of them during one session and delete the others. Although this approach might look similar to ours at first sight, its storage requirements are much higher. Assume for instance that there are $2^8 = 256$ possible passwords of 128 bits each per session; the verifier sends a random 8-bit number to the prover, who returns the corresponding password and delete the others. In this case, for each authentication, $16 * 2^8 = 4$ KBytes are required, but our Merkle scheme stores only $16 * 4 * 8 = 1$ KByte of data. More generally, the naive approach of storing passwords requires $2^\gamma * 16$

bytes where we need only $4 * \gamma * 16$ bytes and enjoy the benefits of a public key authentication protocol.

4 Implementation and Applications

When memory is taken into account, our approach requires a high number of gates when implemented in hardware. However, we emphasize two important facts. First, many platforms *already* have large amounts of memory but little or no computational power; in particular all flash-based memory drives. In this context, it is perfectly feasible to use a small part of their available memory to implement cryptographic functions, but it is more difficult to add a large unit dedicated to cryptography. Second, the price per megabyte of solid-state memory has drastically dropped; for instance, the price per megabyte for flash memory is now close to one cent. This makes a flash-based solution very attractive compared to a pure ASIC design implementing a symmetric or asymmetric cryptographic primitives.

4.1 Memory and Resource Requirements

Shorter challenges result in shorter signatures and decreased storage requirements for the prover. More precisely, the memory requirements are $(64\gamma + 32)\sigma$ bytes[1] , where the challenge has γ bits and σ signatures can be signed. Per signature, there are 32γ bytes for secrets, 32γ bytes for images and 32 bytes for tree data.

Table 1. Memory requirements for the prover using Lamport OTS

	$\sigma = 2^{10} \approx 1,000$	$\sigma = 2^{15} \approx 32,000$	$\sigma = 2^{20} \approx 1,000,000$
$\gamma = 8$ bits	557 KBytes	17.8 MBytes	570 MBytes
$\gamma = 16$ bits	1.08 MByte	34.6 MBytes	1.11 GByte
$\gamma = 64$ bits	4.23 MBytes	135 MBytes	4.33 GBytes

Table 1 illustrates the memory requirements of our scheme in different settings. Note that Table 1 does not include memory for storing the public key (16 bytes) and its certificate (for example 256 bytes in the case of a 2048-bit RSA signature). If less than 1,000 signatures must be signed, and a security level of 8 bits is sufficient, only 557 KBytes are required: 256 KBytes for storing the secret values and 301 KBytes for the rest of the data (images of the secret values and nodes of the tree). With a 16-bit challenge, we have a scheme which offers a reasonable security level (forgery probability of 1/65,536), and 32,000 signatures is more than enough for most applications. In these settings, 35 MBytes are sufficient for storing data. Assuming that the cost per megabyte of the chosen memory type is one cent, in the first case (8-bit challenges), our signature

[1] Using Merkle one-time signatures [3] instead of Lamport one-time signature, one can slightly decrease storage requirements to $32(\gamma + \lceil \log_2 \gamma \rceil + 1)\sigma$.

Table 2. Signature size, transmission time at 9.6Kbps and verification cost

	$\gamma = 8,\ \sigma \approx 1,000$	$\gamma = 16,\ \sigma \approx 32,000$	$\gamma = 64,\ \sigma \approx 1,000,000$
Signature size	416 Bytes	752 Bytes	2.4 KBytes
Transmission	0.35s	0.63s	2s
Verification cost	34 hashes	63 hashes	212 hashes

scheme costs 0.5 cents, and in the second case (16-bit challenges) 35 cents. Let us now consider the extreme case from Table 1: if the challenge has 64 bits, we are already in the realm of cryptographic strength, and one million signatures should satisfy all reasonable use cases (for example, the typical endurance of commercial flash memory is about one million cycles as well). In this scenario, about 4 GBytes are required; although this is a large amount of memory, this is perfectly feasible on recent SSD drives.

Table 2 describes other important features of our scheme: signature size, transmission time and verification cost. One drawback of the Merkle signature scheme is its large signature size, but this drawback is largely compensated by the fact that our scheme utilizes relatively short challenges. As a consequence, the signature size of our scheme remains reasonable, and data transmission can be done even with a low-speed interface, for example with a bandwidth of 9600 bps as illustrated in Table 2. In fact, modern wired or wireless communication interfaces have bandwidths orders of magnitude higher than 9600 bps. As mentioned earlier, signing is essentially free since there are no calculations. And even though verifying is not free, it can be seen in Table 2 that the verification cost is extremely low, and much lower than other public key cryptosystems including elliptic curves and even RSA with a small public exponent.

4.2 Hardware Implementation

Nevertheless, some kind of access control is required in order to prevent attackers from revealing secret information stored in memory. Recall that in the Merkle scheme, the x_i values *only* are secret; the rest of the data, including the images y_i and the tree data z_i can be freely accessed by anyone. Therefore, access control mechanisms should aim at protecting the x_i values only. More precisely, given one challenge bit m_i, one and only one of the two secret values x_{2i} and x_{2i+1} should be revealed.

Hardware Implementation. Figure 4 should convince the reader of the simplicity of such access control mechanism. The memory area to read is selected using the index of the signature and i, a counter going from 0 to $\gamma - 1$. Since there are 2γ secrets of 128 bits (2^4 bytes) per signature, if they are stored sequentially in a table, the data related to the j-th signature starts from byte $2^5 * \gamma * j$. In addition, there are two 128-bit elements per signature element, therefore the i-th signature element x_{2i}, x_{2i+1} starts from the $2^5 i$-th byte in the signature data segment. Upon reading x_{2i}, x_{2i+1}, data is destroyed in the secret table. Note that 64 bytes are read at a time from the secret table, which is the usual data path

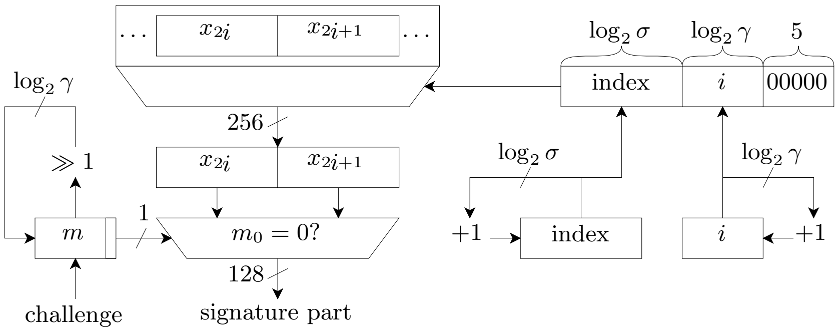


Fig. 4. Hardware implementation of access control

for block transfer of flash memory. The index consists of non-volatile memory, and should be incremented for every new signature. Challenge bits are scanned from left to right, and the challenge is stored in a register m, which is shifted by one bit to the right after each iteration. Thus, at iteration i, the least significant bit m_0 stores the i-th challenge bit. This challenge bit is used to select either x_{2i} or x_{2i+1}, sent to the verifier as part of the signature. The task of constructing the signature, and especially reading the appropriate images y_i and tree data z_i can be left to the verifier since this data is public anyway. Alternatively, the signature could be assembled by the prover since the corresponding circuitry is equally simple.

Tampering with Counters. Although not secret, the values of the index and the counter i play an important role in the security of the scheme. In particular, they should *always* be incremented; if their value is tampered with, attackers might be able to extract additional secret data, which can be used for forging signatures. This is the reason why we suggest that after reading x_{2i} and/or x_{2i+1}, both should be erased from memory. This ensures that even if the value of the counters is tampered with, no secret data can be extracted. This "one-time" read access makes ferroelectric RAM (FeRAM), a type of memory which is used for some RFID tags, especially well-suited for the scheme since it has destructive reading. Of course, usual flash memory may be used as well, provided that data is "manually" erased after reading. In Figure 4, both of x_{2i} and x_{2i+1} are read and stored in two registers. Before starting the transmission of x_{2i+m_i}, both of x_{2i} and x_{2i+1} are erased from the main memory. Following this procedure, it is guaranteed that attackers will recover at most one of the two secrets.

4.3 Applications

Authentication for Removable Storage. The widespread piracy of video games or other digital contents stored on removable storage such as memory cards is a serious problem for the industry. For example, it is known that in some countries, the sale figures of consoles are larger than those of games, which

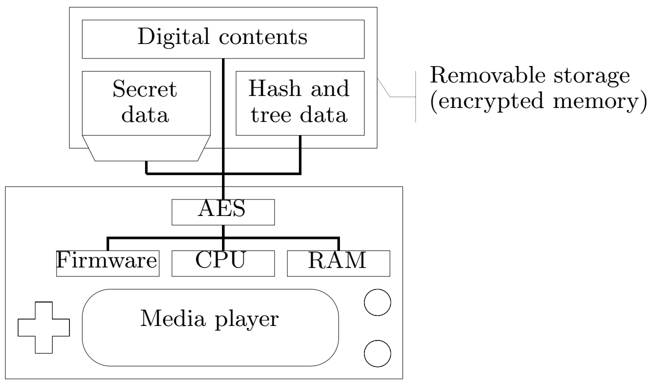


Fig. 5. Authentication for removable storage

implies a widespread use of illegal copies. The usual DRM approach is to store encrypted data with a block cipher such as AES, using a symmetric key which can be derived from secret material stored partially on the player and partially on the storage medium. On the one hand, this approach prevents users from playing digital contents on unauthorized players. On the other hand, it does not address bit-by-bit copy at all. Note that a full memory dump of digital contents is always possible since they must be played in the first place. Even recent commercial copy-protection mechanisms for removable storage do not solve this problem because they use ID numbers: original, read-only media and recordable media are assumed to have different IDs, and the player will reject original contents stored on recordable medium. However, it is technically easy to produce recordable media with forged IDs.

Using authentication would make bit-by-bit copy much more difficult: unlike an ID, which is transmitted in clear to the media player, in an authentication based on a challenge-response protocol, using responses from previous sessions to forge authentication is doomed to fail. Although one could use a different cryptographic scheme implemented on a tamper-resistant chip in order to achieve more or less the same results, our method has the important advantage that little modifications would be required to the existing architectures of storage media, and as a consequence, their production price would be essentially unaffected. On the other hand, the media player is usually an expensive item, and therefore we suggest embedding a dedicated cryptographic engine, for instance AES. This AES chip could transparently decrypt digital contents and forward decrypted data to the main CPU, store sensitive key material and check the integrity of the firmware when booting, but also verify authentication responses if the underlying hash function is based on AES. Public key authentication combines especially well with standard memory encryption, since it prevents bit-by-bit copy (assuming that the contents of the tamper-resistant memory cannot be obtained in a memory dump) and allows revocation mechanisms for *individual* storage items. In addition, since digital contents are encrypted, if the AES chip is incapacitated in order to skip the authentication step, the decryption step cannot be performed anymore, rendering the media player useless. Finally, we argue that an 8-bit challenge is enough: even if an attacker collects sufficiently many challenges and responses, after 256 trials, his probability of successfully authenticating is only 63%.

Authentication for RFID Tickets. Next, we describe another interesting use-case: RFID tags. Although EPCgen2-compliant tags have memory access control mechanisms [2], they typically do not have any full-featured CPU in order to keep prices as low as possible, let alone any onboard cryptographic engine. More and more paper tickets are equipped with RFID tags, which makes touch-and-go access control possible. However, the use of RFID tags does not fundamentally improve security, which almost entirely relies on the back-end system. In fact, the only way to provide real onboard security is to use a true authentication protocol. Although one obvious solution is to add a cryptographic chip to the tag, this will considerably increase the cost of an item which is often designed to be disposable. On the other hand, for a one-time ticket, a (single) one-time signature seems to be the perfect tool: here, the fact that the number of signatures is limited clearly plays in favor of our proposal. In particular, unlike standard signature schemes, it is relatively easy to identify fraudsters who are likely to re-use the same compromised one-time signature. Using a 64-bit hash function with an 8-bit challenge, the required memory for our scheme is only 2112 bits: 1024 bits of secret data, 1024 bits of hash data and 64 bits for the one-time public key. In the spirit of the EPCgen2 specifications [2], memory could be organized in 128-bit segments, where each segment has a read and kill passwords, and where the read password for the segment storing x_{2i} is the same as the kill password for the segment storing x_{2i+1}, thereby providing a mechanism for revealing only one of the two secret values.

5 Conclusion

Our Merkle authentication scheme is the only known public key technique where the prover does not perform *any* computations, since the step of signing boils down to data selection. As a consequence, hardware requirements are extremely low, provided that memory is already available, which is the case in many flash-based platforms. In this scenario, public key authentication functionalities can be implemented essentially for free. Although the verification step consists of a few hash computations, its cost comes close to that of a symmetric key authentication scheme. In addition, security requirements are extremely weak, since all that is needed is a secure hash function. Finally, the fact that no computations are performed makes tamper-resistance easier to achieve; in particular, side-channel attacks are out of scope.

References

1. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
2. EPCglobal: Class 1 generation 2 UHF air interface protocol standard (EPCgen2), http://www.epcglobalinc.com/
3. Merkle, R.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
4. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
5. Indesteege, S., Keller, N., Biham, E., Dunkelman, O., Preneel, B.: A practical attack on KeeLoq. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 1–18. Springer, Heidelberg (2008)
6. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 119–132. Springer, Heidelberg (2004)
7. Lamport, L.: Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (1979)
8. Coronado García, L.C.: On the security and the efficiency of the Merkle signature scheme. Cryptology ePrint Archive, Report 2005/192 (2005)
9. Szydlo, M.: Merkle tree traversal in log space and time. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 541–554. Springer, Heidelberg (2004)
10. 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)
11. Naor, D., Shenhav, A., Wool, A.: One-time signatures revisited: Have they become practical. Cryptology ePrint Archive, Report 2005/442 (2005)
12. Rohatgi, P.: A compact and fast hybrid signature scheme for multicast packet authentication. In: Proceedings of the 6th ACM Conference on Computer and Communications Security - CCS 1999, pp. 93–100. ACM Press, New York (1999)
13. Dahmen, E., Okeya, K., Takagi, T., Vuillaume, C.: Digital signatures out of second-preimage resistant hash functions. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 109–123. Springer, Heidelberg (2008)

Certificate-Based Signatures: New Definitions and a Generic Construction from Certificateless Signatures

Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang[⋆]

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering
University of Wollongong, Australia
{ww986,ymu,wsusilo,xh068}@uow.edu.au

Abstract. Certificate-based encryption was introduced in Eurocrypt'03 to solve the certificate management problem in public key encryption. Recently, this idea has been extended to certificate-based signatures. To date, several new schemes and security models of certificate-based signatures have been proposed. In this paper, we first introduce a new security model of certificate-based signatures. Our model is not only more elaborated when compared with the existing ones, but also defines several new types of adversaries in certificate-based signatures. We then investigate the relationship between certificate-based signatures and certificateless signatures, by proposing a generic construction of certificate-based signatures from certificateless signatures. Our generic construction is secure (in the random oracle model) under the security model defined in this paper, assuming the underlying certificateless signatures satisfying certain security notions.

Keywords: Certificate-based, Certificateless, Security Models, Generic Construction.

1 Introduction

In a public-key cryptography, each user has a pair of keys: public key and private key. The public key is always published and publicly accessible, while the private key is kept secret by the owner. The central problem in a public key system is to prove that a public key is genuine and authentic, and has not been tampered with or replaced by a malicious third party. The usual approach to ensure the authenticity of a public key is to use a certificate. A (digital) certificate is a signature of a trusted certificate authority (CA) that binds together the identity of an entity A, its public key PK and other information. This kind of system is referred as public key infrastructure (traditional PKI). The traditional PKI is generally considered to be costly to use and manage.

[⋆] Supported by National Science Foundation of China (NSFC 60673070).

Shamir [16] introduced the concept of identity-based public key cryptography (or, ID-PKC for short), whose original motivation is to ease the certificate management in the e-mail system. However, key escrow is an inherent problem in this kind of ID-PKC (e.g., [16,5]), as the PKG knows any user's private key. The escrow problem could be partially solved by the introduction of multiple PKGs and the use of threshold techniques, which requires extra communications and infrastructures.

Al-Riyami and Paterson proposed a new paradigm called certificateless public key cryptography [4] (or, CL-PKC for short), whose original motivation is to find a public key system that does not require the use of certificates and does not have the key escrow problem. Each entity in CL-PKC holds two secrets: a secret value and a partial private key. The secret value SV is generated by the entity itself, while a third party Key Generating Center (KGC), holding a master key, generates the partial private key PPK from the user's identity information[1]. The entity's actual private key is the output of some function with the input SV and PPK. This way, KGC does not know the actual private key and the key escrow problem is eliminated. The entity can use the actual private key to generate the public key, which is no longer only computed from the identity. This makes the certificateless system non-identity-based. The entity's public key could be available to other entities by transmitting it along with messages (for example, in a signing application) or by placing it in a public directory (this would be more appropriate for an encryption setting). However, there is no certificate to ensure the authenticity of the entity's public key in CL-PKC. Therefore, it is necessary to assume that an adversary is able to replace the entity's public key with a false key of its choice, which is also known as *key replacement attack*. One assumption is that KGC never mounts the key replacement attack.

In Eurocrypt 2003, Gentry [7] introduced the notion of certificate-based encryption. As in the traditional PKI, each client generates its own public/private key pair and requests a certificate from the CA. The difference is that, a certificate in the certificate-based cryptography, or, more generally, a signature from the third party who acts not only as a certificate (as in the traditional PKI) but also as a decryption key (as in ID-PKC and CL-PKC). The sender can encrypt a message without obtaining explicit information other than the receiver's public key and the parameters of CA. To decrypt a message, a keyholder needs both its secret key and an up-to-date certificate from its CA (or a signature from an authority). Therefore, CA does not need to make the certificate status information available among the whole system, and instead only needs to contact the certificate holder for revocation and update. As the sender is not required to check the certificate of the signer's public key, the sender could be duped to encrypt messages with an uncertified public key. This could be due to the reason that the receiver has not yet got his/her public key certified, or the encryption

[1] In Section 5.1 of [4], the authors sketched an alternative partial private key generation technique. In this paper, when we mention a cryptographic protocol in CL-PKC, we mean it is a protocol with the classic private key generation technique used in Section 4.1 of [4], which has been adopted by most researchers in CL-PKC.

key that the sender has is not the receiver's authentic public key. In this sense, certificate-based encryption works is similar to certificateless encryption, but the difference is that there are certificates in certificate-based encryption.

1.1 Certificate-Based Signatures: Pros and Cons

Certificate-based cryptography was introduced to solve the certificate management problem in the traditional PKI, but only in the scenario of encryption. Later, the notion of certificate-based encryption is extended to certificate-based signature [14,15]. However, as mentioned in [7], if we only consider signing and verification signatures in public key system, then the certificate management problem is not as challenging as in the scenario of encryption and decryption. For example, the signer can send its public key and the proof of certificate status to the verifier simultaneously with its signature, thus the verifier can obtain the certificate without referring to a public directory or issuing a third-party query to CA. In this section, we make a brief comparison of signature schemes in traditional PKI, identity-based cryptography, certificateless cryptography and certificate-based cryptography.

In public key systems like ID-PKC [16,5] and CL-PKC [4], one can directly use the entity A's public key to verify signatures, without checking the certificate of A's public key. Such public key systems can eliminate the certificate management problems in traditional PKI. However, this is achieved at the cost of assuming certain trust on the authority, who is able to impersonate any user in an undetectable way. For example, the PKG in ID-PKC knows any user's private key, and the KGC in CL-PKC is able to replace any user's public key and thus has the knowledge of the private key.

In certificate-based cryptography, one needs two secret information to generate valid signatures of a user with the identity information ID and public key PK: the certificate of (ID, PK) and the secret key of PK. If one replaces PK with PK' and generates a valid signature under ID and PK', she/he must have a certificate of (ID, PK'). This can prove that the third party CA is dishonest, as there is only one party with the ability to generate certificates. Therefore, the third party in certificate-based signatures has the Trust Level 3 in the definition in [9], which is similar as CA in the traditional PKI and few constructions of identity-based signatures [6,8]. To avoid the certificate management problem in the traditional PKI, one must send its certificate to the verifier simultaneously with the signature, instead of only sending the signature. Similar techniques have also been adopted in [6,8]. It is clear to see that such techniques will require more bandwidth for signature transmitting. On the contrary, in certificate-based cryptography, one actually uses the certificate to generate signatures, and does not need send the certificate simultaneously with signatures. The verifier can ensure the existence of the certificate by verifying the validity of signatures. However, certificate-based signatures could require more operation cost, as the signature generation uses both private key and certificate, and the verification of each signature implies the verification of the certificate.

To summarize, (1) The authority in certificate-based signatures and traditional-PKI-based signatures is at Trust Level 3 in the definition given in [9], which is higher than the authority in the ID-PKC and the CL-PKC, and (2) To avoid the problem of certificate management, certificate-based signatures consume (in general) less bandwidth in signature transmitting but might require more computational cost than traditional-PKI-based signatures.

1.2 Related Works about Certificate-Based Signatures

Kang, Park and Hahn [14] proposed the notion and construction of certificate-based signature, by extending the idea of Gentry's [7] certificate-based encryption. That is, to generate a valid signature under the public key PK, the entity needs to know both the corresponding private key SK and the up-to-date certificate of PK. To verify a claimed signature, one only needs the signer's public key and the parameter of CA (particularly, no need to check the certificate of that public key). As the verifier is not required to check the certificate about a claimed public key, key replacement attacks also exist in certificate-based cryptography. Key replacement attacks in certificate-based signatures were first addressed in [14] and formally defined by Li *et al.* [15]. As introduced in [15], adversaries in certificate-based signature can be roughly divided into two types: **CB-**$\mathcal{A}_I$ and **CB-**$\mathcal{A}_{II}$. **CB-**$\mathcal{A}_I$ adversary can replace any entity's public key PK with a new public key PK' chosen by itself, and is trying to forge a valid signature under PK' whose certificate is not available to **CB-**$\mathcal{A}_I$. **CB-**$\mathcal{A}_{II}$ has the knowledge of CA's master key and thus can generate the certificate for any user. **CB-**$\mathcal{A}_{II}$ is trying to forge a valid signature under an entity's authentic public key PK (that is, PK is chosen by that entity), whose private key is not available to **CB-**$\mathcal{A}_{II}$. In addition to the security models, a certificate-based signature scheme secure against key replacement attacks was also proposed in [15]. Very recently, Liu *et al.* proposed two new certificate-based signature schemes [12]. The first one does not require any pairing operation and the security of their second scheme can be proved without random oracles. Some variants of certificate-based signatures (e.g., certificate-based proxy signature [14] and certificate-based linkable ring signature [2]) have also been proposed.

1.3 Motivations and Contributions

As mentioned in [4], certificate-based cryptography and certificateless cryptography is quite similar and there could be a possible method to convert a certificateless cryptographical protocol to a certificate-based cryptographical protocol. This paper is motivated by the question *how to construct a certificate-based signature scheme from a certificateless signature scheme.* The contribution of this paper also includes *new security models of certificate-based signatures*, which we believe is of independent interest.

1. *New Security Models of Certificate-based Signatures*
A reasonable security model is necessary for constructing cryptographic protocols. In this paper, we provide elaborated definitions of certificate-based signatures.

Although the security of certificate-based signatures has been investigated in [14,15], their security definitions are not satisfactory, especially in the exact meaning of key replacement attacks in certificate-based signatures. In our definition, we further divide the potential adversaries into three types: normal adversary, strong adversary and super adversary. The adversary is divided by the attacking power, especially by the information that the adversary can obtain after key replacement attacks. Based on such divisions, we provide a more reasonable and more precise security definition of certificate-based signatures.

2. *Generic Construction of Certificate-based Signatures from Certificateless Signatures*
After giving new security models of certificate-based signatures, we propose a generic construction of certificate-based signatures which is secure in the new proposed models. We show how to build a certificate-based signature scheme from a certificateless signature scheme, by using a hash function as a (one-way) bridge to connect those two primitives. Our method can be used to build certificate-based signature schemes secure against any type of adversaries defined in this paper, assuming that the underlying certificateless signature schemes satisfy certain security notions and the hash function is viewed as the random oracle. Prior to our work, the generic construction of certificate-based encryption from certificateless encryption has been proposed in [3], but a recent work [13] shows the flaw in the security proof of [3].

Organization of Our Paper
The outline of certificate-based signatures (CBS) is presented in next section. We then redefine the security of CBS against different types of attacks in Section 3. In Section 4, we propose a generic construction of certificate-based signatures from certificateless signatures with formal security analysis. Finally, Section 5 concludes the paper.

2 Certificate-Based Signatures

In this section, we will first review the definitions of certificate-based signatures. Then, we will describe oracles used in our security model.

2.1 Syntax of Certificate-Based Signatures

A certificate-based signature (CBS) scheme consists of the following five algorithms:

1. $\mathsf{CB\text{-}Setup}(1^k) \rightarrow (\mathit{CB\text{-}msk}, \mathit{CB\text{-}mpk}, \mathit{CB\text{-}params})$.
   By taking as input a security parameter 1^k, the certifier runs the algorithm $\mathsf{CB\text{-}Setup}$ to generate the certifier's master secret key $\mathit{CB\text{-}msk}$, master public key $\mathit{CB\text{-}mpk}$ and the system parameter $\mathit{CB\text{-}params}$. $\mathit{CB\text{-}params}$ includes the description of a string space Γ, which can be any subset of $\{0,1\}^*$. $\mathit{CB\text{-}msk}$ will be kept as secret by the certifier, while $\mathit{CB\text{-}mpk}$ and $\mathit{CB\text{-}params}$ are publicly accessible by all the users in the system.

For the convenience of using the expression, in the following of the paper, we still use $\mathsf{ID} \in \Gamma$ to denote a user with the identity information ID. But actually, ID contains more information other than the identity.

2. $\mathsf{CB\text{-}UserKeyGen}(CB\text{-}mpk, CB\text{-}params, \mathsf{ID}) \rightarrow (CB\text{-}SK_{\mathsf{ID}}, CB\text{-}PK_{\mathsf{ID}})$.
   The user with the identity information ID runs the algorithm $\mathsf{CB\text{-}UserKeyGen}$ to generate the user ID's secret/public key pair $(CB\text{-}SK_{\mathsf{ID}}, CB\text{-}PK_{\mathsf{ID}}) \in \mathcal{SK}^{\mathcal{CB}} \times \mathcal{PK}^{\mathcal{CB}}$, by taking as input $CB\text{-}mpk$ and $CB\text{-}params$. Here, $\mathcal{SK}^{\mathcal{CB}}$ denotes the set of valid secret key values and $\mathcal{PK}^{\mathcal{CB}}$ denotes the set of valid public key values. The descriptions of $\mathcal{SK}^{\mathcal{CB}}$ and $\mathcal{PK}^{\mathcal{CB}}$ are included in $CB\text{-}params$.
3. $\mathsf{CB\text{-}CertGen}(CB\text{-}msk, CB\text{-}mpk, CB\text{-}params, \mathsf{ID}, CB\text{-}PK_{\mathsf{ID}}) \rightarrow Cert_{\mathsf{ID}}$.
   The certifier runs the algorithm $\mathsf{CB\text{-}CertGen}$ to generate the certificate $Cert_{\mathsf{ID}}$, by taking as input $CB\text{-}msk$, $CB\text{-}mpk$, $CB\text{-}params$, ID and its public key $CB\text{-}PK_{\mathsf{ID}}$.
4. $\mathsf{CB\text{-}Sign}(m, CB\text{-}params, CB\text{-}mpk, \mathsf{ID}, Cert_{\mathsf{ID}}, CB\text{-}SK_{\mathsf{ID}}, CB\text{-}PK_{\mathsf{ID}}) \rightarrow CB\text{-}\sigma$.
   The prospective signer runs the algorithm $\mathsf{CB\text{-}Sign}$ to generate the signature $CB\text{-}\sigma$, by taking as input a message m to be signed, $CB\text{-}params$, $CB\text{-}mpk$, the user's identity ID, its $Cert_{\mathsf{ID}}$ and key pair $(CB\text{-}SK_{\mathsf{ID}}, CB\text{-}PK_{\mathsf{ID}})$.
5. $\mathsf{CB\text{-}Verify}(CB\text{-}mpk, CB\text{-}params, \mathsf{ID}, CB\text{-}PK_{\mathsf{ID}}, (m, CB\text{-}\sigma)) \rightarrow \{true, false\}$.
   Anyone can run the algorithm $\mathsf{CB\text{-}Verify}$ to check the validity of the signature. By taking as input a message/signature pair $(m, CB\text{-}\sigma)$, ID, $CB\text{-}PK_{\mathsf{ID}}$, $CB\text{-}mpk$, $CB\text{-}params$, this algorithm outputs $true$ if $CB\text{-}\sigma$ is ID's valid signature on m. Otherwise, outputs $false$.

Remark. When a user ID requests a certificate of its public key $CB\text{-}PK_{\mathsf{ID}}$, it must prove the certifier his possession of $CB\text{-}SK_{\mathsf{ID}}$. The certifier must also check other information of the user. This can be done as the same way in traditional public key system.

Correctness. Signatures generated by the algorithm $\mathsf{CB\text{-}Sign}$ can pass through the verification in $\mathsf{CB\text{-}Verify}$. That is,
$\mathsf{CB\text{-}Verify}(CB\text{-}mpk, CB\text{-}params, \mathsf{ID}, CB\text{-}PK_{\mathsf{ID}}, (m, \mathsf{CB\text{-}Sign}(m, CB\text{-}params, CB\text{-}mpk, \mathsf{ID}, Cert_{\mathsf{ID}}, CB\text{-}SK_{\mathsf{ID}}, CB\text{-}PK_{\mathsf{ID}}))) \rightarrow true$.

2.2 Adversaries and Oracles

In this section, we will describe the oracles which will be used in the security model of certificate-based signatures. We first give a brief description of adversaries in certificate-based signatures. Formal definitions of these adversaries are given in Section 3.

The security of a certificate-based signature scheme requires that one can generate a valid signature under the public key $CB\text{-}PK_{\mathsf{ID}}$ *if and only if* he has both secrets $Cert_{\mathsf{ID}}$ and $CB\text{-}SK_{\mathsf{ID}}$. In other words, one cannot generate a valid signature with only $Cert_{\mathsf{ID}}$ or $CB\text{-}SK_{\mathsf{ID}}$. As introduced in [15], adversaries in certificate-based signature can be divided into two types: $\mathbf{CB}\text{-}\mathcal{A}_I$ and $\mathbf{CB}\text{-}\mathcal{A}_{II}$. Type I adversary $\mathbf{CB}\text{-}\mathcal{A}_I$ simulates the scenario where the adversary (anyone except the certifier) can replace the public keys of entities at will, but is not allowed to obtain the target user's certificate $Cert_{\mathsf{ID}}$. Key replacement attacks exist in certificate-based signatures as the verifier is not required to check the correctness of a given public key. Type II adversary $\mathbf{CB}\text{-}\mathcal{A}_{II}$ simulates a malicious certifier who can produce certificates but cannot replace the target user's public key. We will use the following oracles to simulate potential attacking scenarios. In the remainder of this paper, we write $\alpha \leftarrow \beta$ to denote the algorithmic action of assigning the value of β to the value α.

1. $\mathcal{O}^{\mathsf{CB-UserCreate}}$: This oracle receives an input $\mathsf{ID} \in \Gamma$ and outputs the public key of user ID. This oracle maintains two lists $L1^{PK}$ and $L2^{PK}$, which are initially empty and used to record the information for each user ID. Both lists $L1^{PK}$ and $L2^{PK}$ are accessible and writable (if needed) by all the other oracles which will be defined shortly. $L1^{PK}$=$\{(\mathsf{ID}, CB\text{-}SK_{\mathsf{ID}}, CB\text{-}PK_{\mathsf{ID}})\}$ provides the information about user ID's secret key and the public key when it is created. $L2^{PK}$=$\{(\mathsf{ID}, CB\text{-}\overline{PK}_{\mathsf{ID}})\}$ provides the information of ID's current public key, which is denoted as $CB\text{-}\overline{PK}_{\mathsf{ID}}$ and might not be the one generated by this oracle.
   (a) For a fresh input ID, the oracle runs the algorithms $\mathsf{CB\text{-}UserKeyGen}$ to obtain the secret key $CB\text{-}SK_{\mathsf{ID}}$ and public key $CB\text{-}PK_{\mathsf{ID}}$. Then it adds $(\mathsf{ID}, CB\text{-}SK_{\mathsf{ID}}, CB\text{-}PK_{\mathsf{ID}})$ to $L1^{PK}$ and $(\mathsf{ID}, CB\text{-}\overline{PK}_{\mathsf{ID}})$ to $L2^{PK}$ where $CB\text{-}\overline{PK}_{\mathsf{ID}} \leftarrow CB\text{-}PK_{\mathsf{ID}}$. After that, it outputs $CB\text{-}PK_{\mathsf{ID}}$. In this case, ID is said to be *created*. Here we assume that other oracles (which will be defined later) only respond to the identity which has been created.
   (b) Otherwise, ID has already been created. The oracle simply finds ID in $L1^{PK}$ and returns $CB\text{-}PK_{\mathsf{ID}}$ as the output.
2. $\mathcal{O}^{\mathsf{CB-PKReplace}}$: This oracle is employed to simulate the scenario that the adversary can replace any user's public key with the public key chosen by itself.

   For a public key replacement query $(\mathsf{ID}, \mathsf{CB\text{-}PK}) \in \Gamma \times \mathcal{CB}^{\mathcal{PK}}$, this oracle finds the user ID in the list $L2^{PK}$, sets $CB\text{-}\overline{PK}_{\mathsf{ID}} \leftarrow \mathsf{CB\text{-}PK}$ and updates the corresponding information as $(\mathsf{ID}, CB\text{-}\overline{PK}_{\mathsf{ID}})$. Note that the adversary is not required to provide the secret key $\mathsf{CB\text{-}SK}$ corresponding to $\mathsf{CB\text{-}PK}$. For a created user ID, the adversary can replace the public key repeatedly.
3. $\mathcal{O}^{\mathsf{CB-Corruption}}$: This oracle takes as input a query ID. It browses the list $L1^{PK}$ and returns the secret key $CB\text{-}SK_{\mathsf{ID}}$. Note that the secret key output by this oracle is the one corresponding to ID's original public key $CB\text{-}PK_{\mathsf{ID}}$ returned by $\mathcal{O}^{\mathsf{CB-UserCreate}}$.
4. $\mathcal{O}^{\mathsf{CB-CertGen}}$: For a certificate request for $(\mathsf{ID}, \mathsf{CB\text{-}PK}) \in \Gamma \times \mathcal{CB}^{\mathcal{PK}}$, this oracle runs the algorithm $\mathsf{CB\text{-}CertGen}$ and returns the certificate $Cert_{\mathsf{ID}}$ for $(\mathsf{ID}, \mathsf{CB\text{-}PK})$. Again, the adversary is not required to provide the corresponding secret key $\mathsf{CB\text{-}SK}$.
5. $\mathcal{O}^{\mathsf{CB-Sign}}$: Due to different levels of signing power the challenger has, this oracle can be further divided into the following three types:

(a) $\mathcal{O}^{\mathsf{CB-NormalSign}}$: This oracle takes as input a query (ID, m), where m denotes the message to be signed. It outputs a signature $CB\text{-}\sigma$ such that $true \leftarrow \mathsf{CB-Verify}(m,\ CB\text{-}\sigma,\ CB\text{-}params,\ \mathsf{ID},\ CB\text{-}PK_{\mathsf{ID}},\ CB\text{-}mpk)$. Here $CB\text{-}PK_{\mathsf{ID}}$ is ID's public key in the list $L1^{PK}$. That is, this oracle only generates ID's signatures which are valid under the public key generated by the oracle $\mathcal{O}^{\mathsf{CB-UserCreate}}$.

(b) $\mathcal{O}^{\mathsf{CB-StrongSign}}$: This oracle takes as input a query $(\mathsf{ID}, m, coin)$, where m denotes the message to be signed, and $coin \in \{1, 2\}$. It acts differently according to $coin$:
   - If $coin = 1$, this oracle works the same as $\mathcal{O}^{\mathsf{CB-NormalSign}}$.
   - Otherwise $coin = 2$, this oracle first checks the list $L1^{PK}$ and $L2^{PK}$ to obtain ID's original public key $CB\text{-}PK_{\mathsf{ID}}$ and ID's current public key $CB\text{-}\overline{PK}_{\mathsf{ID}}$. If $CB\text{-}\overline{PK}_{\mathsf{ID}} = CB\text{-}PK_{\mathsf{ID}}$, this oracle works the same as $\mathcal{O}^{\mathsf{CB-NormalSign}}$. Otherwise $CB\text{-}\overline{PK}_{\mathsf{ID}} \neq CB\text{-}PK_{\mathsf{ID}}$, which means ID's public key has been replaced by the adversary. In this case, $\mathcal{O}^{\mathsf{CB-StrongSign}}$ will ask the adversary to supply the secret key $CB\text{-}\overline{SK}_{\mathsf{ID}} \in \mathcal{SK}^{\mathcal{CB}}$ corresponding to $CB\text{-}\overline{PK}_{\mathsf{ID}}$. After that, this oracle uses $CB\text{-}\overline{SK}_{\mathsf{ID}}$ and the certificate for $(\mathsf{ID}, CB\text{-}\overline{PK}_{\mathsf{ID}})$ to generate the signature $CB\text{-}\sigma$, which will be returned as the answer.

(c) $\mathcal{O}^{\mathsf{CB-SuperSign}}$: This oracle takes as input a query (ID, m), where m denotes the message to be signed. It first finds ID's current public key $CB\text{-}\overline{PK}_{\mathsf{ID}}$ in $L2^{PK}$. This oracle then outputs a signature σ such that $true \leftarrow \mathsf{CB-Verify}(m, \sigma,\ CB\text{-}params, \mathsf{ID},\ CB\text{-}\overline{PK}_{\mathsf{ID}},\ CB\text{-}mpk)$. Note that $\mathcal{O}^{\mathsf{CB-SuperSign}}$ does not require the adversary to supply the secret key $CB\text{-}\overline{SK}_{\mathsf{ID}}$ corresponding to the current public key $CB\text{-}\overline{PK}_{\mathsf{ID}}$ in $L2^{PK}$.

Remark. A Type II adversary **CB-**$\mathcal{A}_{II}$, who simulates the malicious certificate, is not allowed to make any requests to $\mathcal{O}^{\mathsf{CB-CertGen}}$.

3 Security Models

In this section, we will define security models of certificate-based signatures. Our models follow the standard methods: each security notion is defined by the game between the adversary and the challenger, which consists of several oracles defined in Section 2.2. For each security notion, we will first give a brief description of the attack scenario that we are concerning about, and then provide its formal definition. In the definition of the security models, we will use the notation: $\{Q_1, Q_2, \cdots, Q_n\} \nrightarrow \{\mathcal{O}^1, \mathcal{O}^2, \cdots, \mathcal{O}^n\}$ which denotes that "No query $Q \in \{Q_1, Q_2, \cdots, Q_n\}$ can be submitted to any oracle $\mathcal{O} \in \{\mathcal{O}^1, \mathcal{O}^2, \cdots, \mathcal{O}^n\}$. By using this notation, the security models can be described by just pointing out its aim while hiding all details.

Inspired by the security models in certificateless signatures [11], we divide the potential adversaries in certificate-based signatures according to their attack power. They are Normal Adversary, Strong Adversary and Super Adversary. Combined with the known type I adversary and type II adversary in certificate-based system, we now define the security of certificate-based signatures in different attack scenarios.

3.1 Security against Normal Type I Adversary

In this section, we are concerning about the Normal Type I adversary: **Normal-CB-$\mathcal{A}_I$**. Informally, we want to capture the attack scenarios as follows:

1. $\mathcal{A}_I$ can see some message/signature pairs $(m_i, \mathit{CB}\text{-}\sigma_i)$ which are generated by the target user ID by using this ID's secret key $\mathit{CB}\text{-}\mathit{SK}_{\mathsf{ID}}$ and the certificate $\mathit{Cert}_{\mathsf{ID}}$.
2. $\mathcal{A}_I$ is not allowed to obtain the target user ID's secret information. That is, $\mathcal{A}_I$ does not know $\mathit{CB}\text{-}\mathit{SK}_{\mathsf{ID}}$ or $\mathit{Cert}_{\mathsf{ID}}$.
3. $\mathcal{A}_I$ can replace the target user ID's public key with $\mathit{CB}\text{-}\mathit{PK}'_{\mathsf{ID}}$ which is chosen by himself. He can also dupe any other third party to verify user ID's signatures using the false $\mathit{CB}\text{-}\mathit{PK}'_{\mathsf{ID}}$.

The security of certificate-based signature against a **Normal-CB-$\mathcal{A}_I$** is defined by the game described below:

Initial: The challenger runs the algorithm $\mathsf{CB\text{-}Setup}$, returns $\mathit{CB}\text{-}\mathit{params}$ and $\mathit{CB}\text{-}\mathit{mpk}$ to $\mathcal{A}_I$.

Queries: In this phase, $\mathcal{A}_I$ can adaptively access oracles $\mathcal{O}^{\mathsf{CB-UserCreate}}$, $\mathcal{O}^{\mathsf{CB-PKReplace}}$, $\mathcal{O}^{\mathsf{CB-Corruption}}$, $\mathcal{O}^{\mathsf{CB-CertGen}}$, $\mathcal{O}^{\mathsf{CB-NormalSign}}$.

Output: After all the queries, $\mathcal{A}_I$ outputs a forgery $(m^*, \mathit{CB}\text{-}\sigma^*, \mathsf{ID}^*)$. Let $\mathit{CB}\text{-}\overline{\mathit{PK}}_{\mathsf{ID}^*}$ be the current public key of the user ID^* in the list $L2^{PK}$.

Restrictions: We say $\mathcal{A}_I$ wins the game if the forgery satisfies the following requirements:

1. $\mathit{true} \leftarrow \mathsf{CB{-}Verify}(m^*, \mathit{CB}\text{-}\sigma^*, \mathit{CB}\text{-}\mathit{params}, \mathsf{ID}^*, \mathit{CB}\text{-}\overline{\mathit{PK}}_{\mathsf{ID}^*}, \mathit{CB}\text{-}\mathit{mpk})$;
2. $(\mathsf{ID}^*, m^*) \nrightarrow \mathcal{O}^{\mathsf{CB-NormalSign}}$; $(\mathsf{ID}^*, \mathit{CB}\text{-}\overline{\mathit{PK}}_{\mathsf{ID}^*}) \nrightarrow \mathcal{O}^{\mathsf{CB-CertGen}}$; $\mathsf{ID}^* \nrightarrow \mathcal{O}^{\mathsf{CB-Corruption}}$.

The success probability that an adaptive chosen message and chosen identity adversary **Normal-CB-$\mathcal{A}_I$** wins the above game is defined as $\mathit{Succ}^{cma,cida}_{\mathcal{A}_I,normal}$.

Definition 1. *[Security against Normal Type I Adversary] We say a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_{CG}, q_{NS})$ adaptive chosen message and chosen identity adversary* ***Normal-CB-$\mathcal{A}_I$****, if $\mathcal{A}_I$ runs in polynomial time t, makes at most q_{UC} queries to the oracle $\mathcal{O}^{\mathsf{CB-UserCreate}}$, q_{PKR} queries to the oracle $\mathcal{O}^{\mathsf{CB-PKReplace}}$, q_C queries to the oracle $\mathcal{O}^{\mathsf{CB-Corruption}}$, q_{CG} queries to the oracle $\mathcal{O}^{\mathsf{CB-CertGen}}$, q_{NS} queries to the oracle $\mathcal{O}^{\mathsf{CB-NormalSign}}$ and $\mathit{Succ}^{cma,cida}_{\mathcal{A}_I,normal}$ is negligible.*

Remark: The Normal Type I adversary is similar to the one defined in [15]. However, there are two improvements. Firstly, we allow the adversary to replace any user's public key, while the adversary in [15] can only replace the target user's public key. The other improvement in our model is that the adversary can obtain certificates of $(\mathit{ID},\mathit{CB}\text{-}\mathit{PK})$ chosen by itself, but the adversary in [15] can only obtain certificates of the user with original public keys generated by the challenger.

3.2 Security against Strong Type I Adversary

In this section, we upgrade the attack power of Normal Type I adversary and define the Strong Type I adversary: **Strong-CB-$\mathcal{A}_I$**. **Strong-CB-$\mathcal{A}_I$** is more powerful than **Normal-CB-$\mathcal{A}_I$** in the sense that **Strong-CB-$\mathcal{A}_I$** can access the oracle $\mathcal{O}^{\mathsf{CB-StrongSign}}$. In other words, a **Strong-CB-$\mathcal{A}_I$** can obtain a valid signature under the public key replaced by himself, with the restriction that he can supply the corresponding secret key. In addition, **Strong-CB-$\mathcal{A}_I$** can also corrupt the target user ID^*'s secret key $CB\text{-}SK_{\mathsf{ID}^*}$.

Initial: The challenger runs the algorithm $\mathsf{CB\text{-}Setup}$, returns $CB\text{-}params$ and $CB\text{-}mpk$ to $\mathcal{A}_I$.

Queries: In this phase, $\mathcal{A}_I$ can adaptively access the oracles $\mathcal{O}^{\mathsf{CB-UserCreate}}$, $\mathcal{O}^{\mathsf{CB-PKReplace}}$, $\mathcal{O}^{\mathsf{CB-Corruption}}$, $\mathcal{O}^{\mathsf{CB-CertGen}}$, $\mathcal{O}^{\mathsf{CB-StrongSign}}$.

Output: After all the queries, $\mathcal{A}_I$ outputs a forgery $(m^*, CB\text{-}\sigma^*, \mathsf{ID}^*)$. Let $CB\text{-}\overline{PK}_{\mathsf{ID}^*}$ be the current public key of the user ID^* in the list $L2^{PK}$.

Restrictions: We say $\mathcal{A}_I$ wins the game if the forgery satisfies the following requirements:

1. $true \leftarrow \mathsf{CB{-}Verify}(m^*, CB\text{-}\sigma^*, CB\text{-}params, \mathsf{ID}^*, CB\text{-}\overline{PK}_{\mathsf{ID}^*}, CB\text{-}mpk)$;
2. $(\mathsf{ID}^*, m^*) \nrightarrow \mathcal{O}^{\mathsf{CB-StrongSign}}$; $(\mathsf{ID}^*, CB\text{-}\overline{PK}_{\mathsf{ID}^*}) \nrightarrow \mathcal{O}^{\mathsf{CB-CertGen}}$.

The success probability that an adaptive chosen message and chosen identity adversary **Strong-CB-$\mathcal{A}_I$** wins the above game is defined as $Succ^{cma,cida}_{\mathcal{A}_I,strong}$.

Definition 2 (Security against Strong Type I Adversary). *We say a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_{CG}, q_{SS})$ adaptive chosen message and chosen identity adversary **Strong-CB-$\mathcal{A}_I$**, if $\mathcal{A}_I$ runs in polynomial time t, makes at most q_{UC} queries to the oracle $\mathcal{O}^{\mathsf{CB-UserCreate}}$, q_{PKR} queries to the oracle $\mathcal{O}^{\mathsf{CB-PKReplace}}$, q_C queries to the oracle $\mathcal{O}^{\mathsf{CB-Corruption}}$, q_{CG} queries to the oracle $\mathcal{O}^{\mathsf{CB-CertGen}}$, q_{SS} queries to the oracle $\mathcal{O}^{\mathsf{CB-StrongSign}}$ and $Succ^{cma,cida}_{\mathcal{A}_I,strong}$ is negligible.*

If a certificate-based signature scheme is secure against Strong Type I adversary as defined in Def. 2, it is also secure against a Normal Type I adversary as defined in Def. 1.

3.3 Security against Super Type I Adversary

In this section, we will define Super Type I adversary **Super-CB-$\mathcal{A}_I$**. **Super-CB-$\mathcal{A}_I$** is more powerful than **Strong-CB-$\mathcal{A}_I$** (and hence, more powerful than **Normal-CB-$\mathcal{A}_I$**) in the sense that **Super-CB-$\mathcal{A}_I$** can access the oracle $\mathcal{O}^{\mathsf{CB-SuperSign}}$. That is, **Super-CB-$\mathcal{A}_I$** can obtain a valid signature under the public key chosen by himself without providing the corresponding secret key. Obviously, **Super-CB-$\mathcal{A}_I$** is the strongest among all the adversaries.

Initial: The challenger runs the algorithm $\mathsf{CB\text{-}Setup}$, returns $CB\text{-}params$ and $CB\text{-}mpk$ to $\mathcal{A}_I$.

Queries: In this phase, $\mathcal{A}_I$ can adaptively access the oracles $\mathcal{O}^{\mathsf{CB-UserCreate}}$, $\mathcal{O}^{\mathsf{CB-PKReplace}}$, $\mathcal{O}^{\mathsf{CB-Corruption}}$, $\mathcal{O}^{\mathsf{CB-CertGen}}$, $\mathcal{O}^{\mathsf{CB-SuperSign}}$.

Output: After all the queries, $\mathcal{A}_I$ outputs a forgery $(m^*,\ CB\text{-}\sigma^*, \mathsf{ID}^*)$. Let $CB\text{-}\overline{PK}_{\mathsf{ID}^*}$ be the current public key of the user ID^* in the list $L2^{PK}$.

Restrictions: We say $\mathcal{A}_I$ wins the game if the forgery satisfies the following requirements:

1. $true \leftarrow \mathsf{CB{-}Verify}(m^*, CB\text{-}\sigma^*, CB\text{-}params, \mathsf{ID}^*, CB\text{-}\overline{PK}_{\mathsf{ID}^*}, CB\text{-}mpk)$;
2. $(\mathsf{ID}^*, m^*) \nrightarrow \mathcal{O}^{\mathsf{CB-SuperSign}}$; $(\mathsf{ID}^*, CB\text{-}\overline{PK}_{\mathsf{ID}^*}) \nrightarrow \mathcal{O}^{\mathsf{CB-CertGen}}$.

The success probability of an adaptively chosen message and chosen identity adversary **Super-CB-$\mathcal{A}_I$** wins the above game is defined as $Succ^{cma,cida}_{\mathcal{A}_I,super}$.

Definition 3 (Security against Super Type I Adversary). *We say a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_{CG}, q_{SS})$ adaptive chosen message and chosen identity adversary* ***Super-CB-$\mathcal{A}_I$****, if $\mathcal{A}_I$ runs in polynomial time t, makes at most q_{UC} queries to the oracle $\mathcal{O}^{\mathsf{CB-UserCreate}}$, q_{PKR} queries to the oracle $\mathcal{O}^{\mathsf{CB-PKReplace}}$, q_C queries to the oracle $\mathcal{O}^{\mathsf{CB-Corruption}}$, q_{CG} queries to the oracle $\mathcal{O}^{\mathsf{CB-CertGen}}$, q_{SS} queries to the oracle $\mathcal{O}^{\mathsf{CB-SuperSign}}$ and $Succ^{cma,cida}_{\mathcal{A}_I,super}$ is negligible.*

If a certificate-based signature scheme is secure against Super Type I adversary as defined in Def. 3, it is also secure against a Strong Type I adversary as defined in Def. 2, and hence secure against a Normal Type I adversary as defined in Def. 1.

3.4 Security against Type II Adversary

In certificate-based signatures, a type II adversary CB-$\mathcal{A}_{II}$ simulates the certifier who is equipped with the master secret key and might engage in other adversarial activities, such as eavesdropping on signatures and making signing queries. Similar to type I adversary, type II adversary **CB-$\mathcal{A}_{II}$** could be also divided into **Normal-CB-$\mathcal{A}_{II}$**, **Strong-CB-$\mathcal{A}_{II}$**, **Super-CB-$\mathcal{A}_{II}$**, which has the access to $\mathcal{O}^{\mathsf{CB-NormalSign}}$, $\mathcal{O}^{\mathsf{CB-StrongSign}}$, $\mathcal{O}^{\mathsf{CB-SuperSign}}$, respectively. However, there is no need to particularly define **Strong-CB-$\mathcal{A}_{II}$**. $\mathcal{O}^{\mathsf{CB-StrongSign}}$ can answer the queries either by using the $\mathcal{O}^{\mathsf{CB-NormalSign}}$ (then the $\mathcal{O}^{\mathsf{CB-StrongSign}}$ is exactly the same as $\mathcal{O}^{\mathsf{CB-NormalSign}}$), or signing the message after getting the corresponding secret key provided by the adversary. Note that, what the **CB-$\mathcal{A}_{II}$** initially has obtained is the master secret key, namely, he can calculate any user's certificate by himself. If he knows the secret key as well, he can generate the signature by himself and $\mathcal{O}^{\mathsf{CB-StrongSign}}$ becomes useless. Therefore, for a type II adversary CB-$\mathcal{A}_{II}$, it is sufficient for us to only define two types of adversaries **Normal-CB-$\mathcal{A}_{II}$** and **Super-CB-$\mathcal{A}_{II}$**.

Initial: The challenger runs the algorithm $\mathsf{CB\text{-}Setup}$ and returns the system parameters *CB-params*, master secret key *CB-msk* and master public key *CB-mpk* to $\mathcal{A}_{II}$.

Queries: $\mathcal{A}_{II}$ can adaptively access oracles $\mathcal{O}^{\mathsf{CB-UserCreate}}$, $\mathcal{O}^{\mathsf{CB-PKReplace}}$, $\mathcal{O}^{\mathsf{CB-Corruption}}$ and $\mathcal{O}^{\mathsf{CB-Sign}}$, where $\mathcal{O}^{\mathsf{CB-Sign}} \in \{\mathcal{O}^{\mathsf{CB-NormalSign}}, \mathcal{O}^{\mathsf{CB-SuperSign}}\}$. All these oracles are the same as defined in Section 2.2.

Output: After all the queries, $\mathcal{A}_{II}$ outputs a forgery (m^*, $CB\text{-}\sigma^*, \mathsf{ID}^*$).

Restrictions: We say $\mathcal{A}_{II}$ wins the game if the forgery satisfies the requirements as following:

1. $true \leftarrow \mathsf{CB-Verify}(m, CB\text{-}\sigma^*, CB\text{-}params, \mathsf{ID}^*, CB\text{-}PK_{\mathsf{ID}^*}, CB\text{-}mpk)$. Here $CB\text{-}PK_{\mathsf{ID}^*}$ is the original public key in the list $L1^{PK}$;
2. $(\mathsf{ID}^*, m^*) \nrightarrow \mathcal{O}^{\mathsf{CB-Sign}}$; $\mathsf{ID}^* \nrightarrow \mathcal{O}^{\mathsf{CB-Corruption}}$.

The success probability that an adaptive chosen message and chosen identity adversary **CB-**$\mathcal{A}_{II}$ wins the above game is defined as $Succ_{\mathcal{A}_{II}}^{cma,cida}$.

Definition 4 (Security against Type II Adversary). *We say a certificate-based signature scheme is secure against a* $(t, q_{UC}, q_{PKR}, q_C, q_S)$ *adaptive chosen message and chosen identity Type II adversary* ***CB-***$\mathcal{A}_{II}$*, if* $\mathcal{A}_{II}$ *runs in polynomial time* t*, makes at most* q_{UC} *queries to the oracle* $\mathcal{O}^{\mathsf{CB-UserCreate}}$*,* q_{PKR} *queries to the oracle* $\mathcal{O}^{\mathsf{CB-PKReplace}}$*,* q_C *queries to the oracle* $\mathcal{O}^{\mathsf{CB-Corruption}}$*,* q_S *queries to the oracle* $\mathcal{O}^{\mathsf{CB-Sign}}$ *and* $Succ_{\mathcal{A}_{II}}^{cma,cida}$ *is negligible.*

Similarly, we can define the security against malicious-but-passive Type II adversary which is first introduced to certificateless cryptography in [1].

4 Generic Construction of Certificate-Based Signatures

In this section, we will introduce a generic method to construct certificate-based signatures. Our construction is based on certificateless signatures whose description is as below.

4.1 Syntax of Certificateless Signatures

A certificateless signature (CLS) scheme is defined by six algorithms: CL-Setup, CL-PPKExtract(Partial Private Key Extract), CL-SSValue(Set Secret Value), CL-SPKey(Set Public Key), CL-Sign and CL-Verify. To distinguish from the identity information in the certificate-based system (which is denoted as ID), we use the notion ID to denote the identity information in the certificateless system. For other information, we put the prefix "*CL-*" to specify that this is in the certificateless system. Details of the description is given in Appendix A.

4.2 Generic Construction: CLS-2-CBS

In this section, we show how to convert a certificateless signature scheme into a certificate-based signature scheme. In our construction, we need a hash function $H : \Gamma \times \mathcal{PK}^{\mathcal{CB}} \rightarrow \mathcal{ID}^{\mathcal{CL}}$ where Γ is the set of identity information in the certificate-based system, $\mathcal{PK}^{\mathcal{CB}}$ is public key space in certificate-based system and $\mathcal{ID}^{\mathcal{CL}}$ denotes the set of identities in the certificateless cryptosystem[2];.

Let CLS be the certificateless signature scheme defined in Section 4.1. We now describe the generic construction CLS-2-CBS.

1. CB-Setup$(1^k) \rightarrow (CB\text{-}msk,\ CB\text{-}mpk,\ CB\text{-}params)$.
   (a) Run algorithm CL-Setup(1^k) of CLS to obtain $CL\text{-}params$, $CL\text{-}msk$ and $CL\text{-}mpk$;
   (b) Set $CB\text{-}params$ by extending $CL\text{-}params$ to include the description of Γ;
   (c) $(CB\text{-}msk,\ CB\text{-}mpk) \leftarrow (CL\text{-}msk,\ CL\text{-}mpk)$.
2. CB-UserCreate$(CB\text{-}mpk,\ CB\text{-}params, \mathsf{ID} \in \Gamma) \rightarrow (CB\text{-}SK_{\mathsf{ID}},\ CB\text{-}PK_{\mathsf{ID}})$.
   (a) $CL\text{-}mpk \leftarrow CB\text{-}mpk$;
   (b) Extract $CL\text{-}params$ from $CB\text{-}params$;
   (c) $CB\text{-}SK_{\mathsf{ID}} \leftarrow$ CL-SSValue$(CL\text{-}mpk,\ CL\text{-}params)$;
   (d) $CB\text{-}PK_{\mathsf{ID}} \leftarrow$ CL-SPKey$(CL\text{-}mpk,\ CL\text{-}params,\ CB\text{-}SK_{\mathsf{ID}})$.
3. CB-CertGen$(CB\text{-}msk,\ CB\text{-}mpk,\ CB\text{-}params,\ \mathsf{ID} \in \Gamma,\ CB\text{-}PK_{\mathsf{ID}}) \rightarrow Cert_{\mathsf{ID}}$.
   (a) $(CL\text{-}msk,\ CL\text{-}mpk) \leftarrow (CB\text{-}msk,\ CB\text{-}mpk)$;
   (b) Extract $CL\text{-}params$ from $CB\text{-}params$;
   (c) $H(\mathsf{ID},\ CB\text{-}PK_{\mathsf{ID}}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
   (d) $Cert_{\mathsf{ID}} \leftarrow$ CL-PPKExtract$(CL\text{-}msk,\ CL\text{-}mpk,\ CL\text{-}params, ID)$.
4. CB-Sign$(m,\ CB\text{-}params,\ CB\text{-}mpk,\ \mathsf{ID}, Cert_{\mathsf{ID}},\ CB\text{-}SK_{\mathsf{ID}},\ CB\text{-}PK_{\mathsf{ID}}) \rightarrow CB\text{-}\sigma$.
   (a) Extract $CL\text{-}params$ from $CB\text{-}params$;
   (b) $CL\text{-}mpk \leftarrow CB\text{-}mpk$;
   (c) $H(\mathsf{ID},\ CB\text{-}PK_{\mathsf{ID}}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
   (d) $(CL\text{-}SV_{ID},\ CL\text{-}PK_{ID}) \leftarrow (CB\text{-}SK_{\mathsf{ID}},\ CB\text{-}PK_{\mathsf{ID}})$, $CL\text{-}PPK_{ID} \leftarrow Cert_{\mathsf{ID}}$;
   (e) $CB\text{-}\sigma \leftarrow$ CL-Sign$(m,\ CL\text{-}params,\ CL\text{-}mpk,\ ID,\ CL\text{-}SV_{ID},\ CL\text{-}PK_{ID},\ CL\text{-}PPK_{ID})$.
5. CB-Verify$(CB\text{-}params,\ CB\text{-}mpk,\ \mathsf{ID},\ CB\text{-}PK_{\mathsf{ID}},\ (m,\ CB\text{-}\sigma)) \rightarrow \{true,\ false\}$.
   (a) Extract $CL\text{-}params$ from $CB\text{-}params$;
   (b) $CL\text{-}mpk \leftarrow CB\text{-}mpk$;
   (c) $H(\mathsf{ID},\ CB\text{-}PK_{\mathsf{ID}}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
   (d) $CL\text{-}PK_{ID} \leftarrow CB\text{-}PK_{\mathsf{ID}}$;
   (e) $CL\text{-}\sigma \leftarrow CB\text{-}\sigma$.
   (f) Output CL-Verify$(CL\text{-}mpk,\ CL\text{-}params, ID,\ CL\text{-}PK_{ID},\ (m,\ CL\text{-}\sigma))$.

[2] Here, we use the hash function H to "equal" two identities in certificate-based signatures and certificateless signatures. This is different from the technique in the generic construction of certificate-based encryption proposed in [3]. By viewing H as the random oracle, our generic construction does not have the flaw of the security proof of [3] shown in [13].

Correctness. We show that any certificate-based signature output by CB-Sign will pass through the check in CB-Verify.

In our construction, a certificate-based signature is the output of the algorithm CL-Sign in the certificateless system, and algorithm CB-Verify also employs the verification algorithm CL-Verify in the certificateless system. To show the correctness of our construction, it suffices to show that under the same $CL\text{-}params$ and $CL\text{-}mpk$, a certificateless signature produced by using the secret value $CL\text{-}SV_{ID}$ and the partial private key $CL\text{-}PPK_{ID}$ will pass through the check using the corresponding identity ID and its public key $CL\text{-}PK_{ID}$. This is guaranteed by the correctness of the underlying certificateless signature scheme, that is, for any signature $CL\text{-}\sigma$ produced by CL-Sign(m, $CL\text{-}params$, $CL\text{-}mpk$, ID, $CL\text{-}SV_{ID}$, $CL\text{-}PK_{ID}$, $CL\text{-}PPK_{ID}$), CL-Verify($CL\text{-}mpk$, $CL\text{-}params$, ID, $CL\text{-}PK_{ID}$, (m, $CL\text{-}\sigma$)) will output *true*. Therefore, for any signature output by CB-Sign defined in our construction, the algorithm CB-Verify will output *true*.

Security Analysis

Theorem 1. *[Security of CLS-2-CBS]* CLS-2-CBS *is secure (in the random oracle model) against the adversaries defined in Section. 3, assuming the underling certificateless signature scheme* CLS *satisfying certain security requirement.*

Proof. Please refer to the full version.

5 Conclusion

The focus of this paper was on certificate-based signatures. We demonstrated the pros and cons of the certificate-based signature, by comparing it with digital signatures in other popular public key systems. We then defined several new types of adversaries and gave a new security model of certificate-based signatures. Our model is more elaborated compared to other existing security models of certificate-based signatures. We proposed a generic construction of certificate based signatures from certificateless signatures. Our generic construction is secure (in the random oracle model) under the security model proposed in this paper, if the underlying certificateless signatures satisfying certain security notions.

References

1. Au, M.H., Chen, J., Liu, J., Mu, Y., Wong, D., Yang, G.: Malicious KGC Attacks in Certificateless Cryptography. In: ASIACCS 2007, pp. 302–311. ACM, New York (2007), http://eprint.iacr.org/2006/255
2. Au, M.H., Liu, J., 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)
3. Al-Riyami, S.S., Paterson, K.G.: CBE from CL-PKE: A Generic Construction and Efficient Schemes. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 398–415. Springer, Heidelberg (2005)

4. 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)
5. Boneh, D., Franklin, M.: Identity-based Encryption from the Weil Pairing. SIAM J. Comput. 32, 586–615 (2003); a Preliminary Version Appeared In: Kilian, J. (ed.): CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. 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)
7. 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)
8. 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)
9. Girault, M.: Self-certified public keys. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 490–497. Springer, Heidelberg (1991)
10. Huang, X., Susilo, W., Mu, Y., Zhang, F.: On the Security of Certificateless Signature Schemes from Asiacrypt 2003. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 13–25. Springer, Heidelberg (2005)
11. Huang, X., Mu, Y., Susilo, W., Wong, D.S., Wu, W.: Certificateless Signature Revisited. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 308–322. Springer, Heidelberg (2007)
12. Liu, J.K., Baek, J., Susilo, W., Zhou, J.: Certificate Based Signature Schemes without Pairings or Random Oracles. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222. Springer, Heidelberg (2008), http://eprint.iacr.org/2008/275
13. Kang, G.H., Park, J.H.: Is it possible to have CBE from CL-PKE? In: Cryptology ePrint Archive, http://eprint.iacr.org/2005/431
14. 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)
15. 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)
16. 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)
17. Zhang, Z., Wong, D.S., Xu, J., Feng, D.: Certificateless public-key signature: Security model and efficient construction. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 293–308. Springer, Heidelberg (2006)

A Syntax of Certificateless Signature

1. $\mathsf{CL\text{-}Setup}(1^k) \rightarrow (CL\text{-}msk,\ CL\text{-}mpk,\ CL\text{-}params)$.
   This algorithm takes as input a security parameter 1^k and returns the master secret key $CL\text{-}msk$ and master public key $CL\text{-}mpk$. It also outputs a parameter $CL\text{-}params$ which is shared in the system. Here $CL\text{-}params$ includes the identity information space $\mathcal{ID}^{\mathcal{CL}}$, the set of the valid secret values $\mathcal{S}^{\mathcal{CL}}$, the set of the valid public key $\mathcal{PK}^{\mathcal{CL}}$, and etc.
2. $\mathsf{CL\text{-}PPKExtract}(CL\text{-}msk,\ CL\text{-}mpk,\ CL\text{-}params, ID) \rightarrow CL\text{-}PPK_{ID}$.
   This algorithm takes as input the master secret key $CL\text{-}msk$, the master public key $CL\text{-}mpk$, system parameter $CL\text{-}params$ and the user's identity information $ID \in \mathcal{ID}^{\mathcal{CL}}$. The output is the partial private key $CL\text{-}PPK_{ID}$, which will be secretly sent to the user.
3. $\mathsf{CL\text{-}SSValue}(CL\text{-}mpk,\ CL\text{-}params) \rightarrow CL\text{-}SV_{ID}$.
   This algorithm takes as input the master public key $CL\text{-}mpk$ and system parameter $CL\text{-}params$. The user ID runs this algorithm to generate its secret value $CL\text{-}SV_{ID} \in \mathcal{S}^{\mathcal{CL}}$.
4. $\mathsf{CL\text{-}SPKey}(CL\text{-}mpk,\ CL\text{-}params,\ CL\text{-}SV_{ID}) \rightarrow CL\text{-}PK_{ID}$.
   This algorithm takes as input the master public key $CL\text{-}mpk$, system parameter $CL\text{-}params$, the user ID's secret value $CL\text{-}SV_{ID} \in \mathcal{S}^{\mathcal{CL}}$. It outputs the public key $CL\text{-}PK_{ID} \in \mathcal{PK}^{\mathcal{CL}}$.
5. $\mathsf{CL\text{-}Sign}(m,\ CL\text{-}params,\ CL\text{-}mpk,\ ID,\ CL\text{-}SV_{ID},\ CL\text{-}PK_{ID},\ CL\text{-}PPK_{ID}) \rightarrow CL\text{-}\sigma$.
   This algorithm takes as input the message m to be signed, system parameter $CL\text{-}params$, the master public key $CL\text{-}mpk$, an identity ID, this identity's secret value $CL\text{-}SV_{ID} \in \mathcal{S}^{\mathcal{CL}}$, public key $CL\text{-}PK_{ID}$ and partial private key $CL\text{-}PPK_{ID}$. It outputs a certificateless signature $CL\text{-}\sigma$.
6. $\mathsf{CL\text{-}Verify}(CL\text{-}mpk,\ CL\text{-}params, ID,\ CL\text{-}PK_{ID},\ (m,\ CL\text{-}\sigma)) \rightarrow \{true, false\}$.
   This algorithm takes as input the master public key $CL\text{-}mpk$, system parameter $CL\text{-}params$, an identity ID, this identity's public key $CL\text{-}PK_{ID}$ and a message/signature pair $(m,\ CL\text{-}\sigma)$. It outputs $true$ if the signature is correct, or $false$ otherwise.

Remark. In general, KGC (Key Generation Center) performs the algorithms CL-Setup and CL-PPKExtract, the user runs the algorithms CL-SSValue, CL-SPKey and CL-Sign. Anyone can runs the algorithm CL-Verify to check the validity of the signature. To generate the secret value $CL\text{-}SV_{ID}$ and the public key $CL\text{-}PK_{ID}$, we assume that one does not need to use the partial private key $CL\text{-}PPK$. Most of CLS schemes satisfy this requirement (e.g., [11,17]).

The correctness requires that signatures generated by the algorithm CL-Sign can pass through the check in CL-Verify. That is, $\mathsf{CL\text{-}Verify}(CL\text{-}mpk, CL\text{-}params, ID, CL\text{-}PK_{ID}, (m, \mathsf{CL\text{-}Sign}(m, CL\text{-}params, CL\text{-}mpk, ID, CL\text{-}SV_{ID}, CL\text{-}PK_{ID}, CL\text{-}PPK_{ID})) \rightarrow true$. Please refer to [11] for security definitions of CLS.

Cryptanalysis of Mu et al.'s and Li et al.'s Schemes and a Provably Secure ID-Based Broadcast Signcryption (IBBSC) Scheme

S. Sharmila Deva Selvi, S. Sree Vivek*, Ragavendran Gopalakrishnan, Naga Naresh Karuturi**, and C. Pandu Rangan*

Indian Institute of Technology Madras
Theoretical Computer Science Laboratory
Department of Computer Science and Engineering
Chennai, India
{sharmila,svivek,ragav,nnaresh,prangan}@cse.iitm.ac.in

Abstract. In applications like wireless content distribution, a central authority needs to deliver encrypted data to a large number of recipients in such a way that only a privileged subset of users can decrypt it. In addition, to avert junk content or spam, subscribers must have source authentication with respect to their broadcasters. The limited memory and computational power of mobile devices, coupled with escalating costs of wireless bandwidth make efficiency a major concern. *Broadcast signcryption*, which enables the broadcaster to simultaneously encrypt and sign the content meant for a specific set of users in a single logical step, provides the most efficient solution to this dual problem of confidentiality and authentication. It is arguably most efficiently implemented in the ID-based setting because of its well known advantages. Only three IBBSC schemes exist in literature, one of which has already been shown to be flawed and its security leaks fixed. In this paper, we show that the remaining two — Mu et al.'s scheme and Li et al.'s scheme are also flawed. Specifically, we show that while Mu et al.'s scheme is insecure with respect to unforgeability, Li et al.'s scheme can be totally broken (with respect to both unforgeability and confidentiality). Following this, we propose a new IBBSC scheme and formally prove its security under the strongest existing security models for broadcast signcryption (IND-CCA2 and EUF-CMA).

Keywords: Signcryption, Cryptanalysis, ID-based Cryptosystem, Broadcast Encryption, Provable Security, Random Oracle, Bilinear Pairing.

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

** Work supported by Project No. CSE/05-06/075/MICO/CPAN on Foundation Research in Cryptography sponsored by Microsoft Research India.

1 Introduction

With the advent of mobile and portable devices such as cell phones and PDAs used in wireless networks, accessing multimedia content through these devices in the wireless network is increasingly popular. On the other hand, a wireless network is much easier to eavesdrop than a wired network. Therefore, the need to securely deliver multimedia content to the user over a wireless network is becoming more important and critical. Furthermore, wireless communication is a good way to broadcast messages to many users in one go. The most efficient way to broadcast information securely is through a *broadcast encryption scheme* in which a broadcaster can send secure information to dynamic selective recipients such that no other recipients outside the set could recover the secret information. A broadcasting news channel may face this problem, for example, when a large number of people subscribe to a daily exclusive news feature. Normally, a broadcast encryption scheme is used to distribute a session key to many users and the intended users would be able to recover the session key, which is used to decrypt encrypted multimedia content sent by a broadcaster.

In many applications, it is also desirable that the users have source authentication with respect to their broadcaster, in order to eliminate spam content. Continuing the example of the news channel, if all the users who subscribe to the news feed receive meaningless noise or any unwanted content, then the broadcaster is going to lose them. This results in the need for *authenticated broadcast encryption*, otherwise known as *broadcast signcryption*. The efficiency of a broadcast signcryption scheme is mainly measured by three parameters — length of transmission messages, storage cost, and computational overhead at a user device. All these parameters are extremely important to mobile devices as they have limited memory and computational power as compared to a personal computer, and wireless bandwidth is extremely costly. Identity-based (ID-based) schemes are the most suited for meeting these restrictions, because of the unique advantage that they provide — the public key of a user can be computed from any publicly available parameter of that user that is unique to him, thereby eliminating the complex public key infrastructure that would otherwise have to be employed.

Related Work. The area of broadcast signcryption or authenticated broadcast encryption is relatively new compared to its parent primitives, namely signcryption and broadcast encryption. Some of the broadcast signcryption schemes that have been proposed in the recent past are [1,3,7]. In the ID-based setting, to the best of our knowledge, only three such schemes exist till date. In 2004, Bohio et al. [2] and Mu et al. [5] proposed two IBBSC schemes. The third scheme was proposed by Li et al. [4] in 2008. In Bohio et al.'s scheme, a secret value must be established apriori among the users before the protocol is initiated. Hence, though the scheme achieves constant size ciphertexts, the set of receivers is essentially static. This scheme cannot therefore be strictly viewed as a broadcast encryption scheme. Nevertheless, flaws have been discovered and fixed by Sharmila et al. [6]. Coming to Mu et al.'s scheme, the authors have proven their scheme secure against two types of attacks called *insider attack* and *outsider attack*. In the former, they show that a legal user of the system cannot find the secret key of the broadcaster and in the latter, they show that an external adversary cannot decrypt and recover the encrypted message. As far as authentication is concerned, though they claim their scheme to be unforgeable, they do not prove it formally. Li et al. claim their scheme to be secure with respect to authentication as well as confidentiality, but prove neither formally.

Our Contribution. We standardize the formal framework for IBBSC and the security models for confidentiality and authentication for IBBSC (which we call IND-IBBSC-CCA2 and EUF-IBBSC-CMA respectively). We take up two IBBSC schemes that have been proposed — one by Mu et al.[1] [5] and another by Li et al. [4]. Though Mu et al. claim their scheme to be unforgeable, they do not prove it formally. We demonstrate a universal forgeability attack on their scheme — any legal user, on receiving and decrypting a valid ciphertext from a broadcaster, can generate a valid ciphertext on any message on behalf of that broadcaster for the same set of legal receivers to which the broadcaster signcrypted the earlier message, without knowing any secrets. Li et al. claim that their scheme provides message authentication and confidentiality, though they prove neither property formally. We demonstrate that any eavesdropper or non-privileged user can decrypt the signcrypted ciphertext of a broadcaster. Further, we show how their scheme can be totally broken by demonstrating how a legitimate user can recover the secret key of his broadcaster on seeing a valid signcryption. Following this, we also propose a new IBBSC scheme as a fix to Li et al.'s faulty scheme, which we formally prove the secure under the strongest existing security models for broadcast signcryption (IND-CCA2 and EUF-CMA).

Organization. The rest of this paper is structured as follows. In Section 2, we review the underlying cryptographic concepts that are involved like bilinear pairings and the related computational problem. In Section 3, we present the general framework for ID-based Broadcast Signcryption (IBBSC) and its formal security models for authentication as well as confidentiality. Next, in Section 4, we review the ID-based Authenticated broadcast encryption scheme of Mu et al. We present our insider universal forgeability attack on this scheme in Section 5. In Section 6, we review Li et al.'s IBBSC scheme. Our attacks on this scheme come next in Section 7. Following this, in Section 8, we lay out the details of our new IBBSC scheme. We present the formal proof of confidentiality of our improved scheme in Section 9. The proof of unforgeability is presented in the full version of this paper. The proofs are presented in the strongest existing security models for IBBSC. In Section 11, we discuss the efficiency of our scheme. Finally, in Section 12, we conclude the discussion and pose some interesting open problems.

[1] Though they call their scheme as an authenticated broadcast encryption scheme, it achieves the same security goals as broadcast signcryption and hence, their scheme is also a broadcast signcryption scheme.

2 Preliminaries

2.1 Bilinear Pairing

Let $\mathbb{G}_1$ be an additive cyclic group generated by P, with prime order q, and $\mathbb{G}_2$ be a multiplicative cyclic group of the same order q. A bilinear pairing is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties.

- ***Bilinearity.*** For all $P, Q, R \in \mathbb{G}_1$,
  - $\hat{e}(P+Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$
  - $\hat{e}(P, Q+R) = \hat{e}(P, Q)\hat{e}(P, R)$
  - $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$
- ***Non-Degeneracy.*** There exist $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq I_{\mathbb{G}_2}$, where $I_{\mathbb{G}_2}$ is the identity element of $\mathbb{G}_2$.
- ***Computability.*** There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

2.2 Computational Diffie-Hellman Problem (CDHP)

Given $(P, aP, bP) \in \mathbb{G}_1^3$ for unknown $a, b \in \mathbb{Z}_q^*$, the CDH problem in $\mathbb{G}_1$ is to compute abP.

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

$$Adv_{\mathcal{A}}^{CDH} = Pr\left[\mathcal{A}(P, aP, bP) = abP \mid a, b \in \mathbb{Z}_q^*\right]$$

The *CDH Assumption* is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{CDH}$ is negligibly small.

3 ID-Based Broadcast Signcryption (IBBSC)

In this section, we present the general framework for IBBSC along with the formal security model that addresses the security properties of confidentiality and authentication.

3.1 Framework of ID-Based Broadcast Signcryption (IBBSC)

A generic ID-based broadcast signcryption scheme for sending a single message from a broadcaster to t users consists of the following probabilistic polynomial time algorithms.

1. ***Setup*(k).** Given a security parameter k, the Private Key Generator (PKG) generates the public parameters *params* and master secret key *msk* of the system.

2. ***Keygen***(ID_A)***.*** Given an identity ID_A, the PKG, using the public parameters *params* and the master secret key msk, computes the corresponding private key S_A and transmits it to A in a secure way.
3. ***Signcrypt***$(m, ID_A, \mathcal{L} = \{ID_1, ID_2, \dots, ID_t\}, S_A)$***.*** To send a message m to t users with identities $(ID_1, ID_2, \dots, ID_t)$, the broadcaster A with identity ID_A and private key S_A runs this algorithm to obtain the signcrypted ciphertext σ.
4. ***Designcrypt***$(\sigma, ID_A, ID_i, S_i)$***.*** When user i with identity ID_i and private key S_i receives the signcrypted ciphertext σ from his broadcaster A with identity ID_A, he runs this algorithm to obtain either the plain text m or $\perp$ according as whether σ was a valid signcryption from identity ID_A to identity ID_i or not.

For consistency, if $\sigma = Signcrypt\,(m, ID_A, \{ID_1, ID_2, \dots, ID_t\}, S_A)$, then we require that for all $1 \leq i \leq t$, $m = Designcrypt\,(\sigma, ID_A, ID_i, S_i)$.

3.2 Security Model for ID-Based Broadcast Signcryption

The two security properties that are desired out of any IBBSC scheme are *message confidentiality* and *unforgeability.* We formally extend the existing strongest security notions for encryption and digital signatures (IND-CCA2 and IND-CMA respectively) to IBBSC below.

Indistinguishability under Adaptive Chosen Ciphertext Attack for IBBSC (IND-IBBSC-CCA2). An ID-based broadcast signcryption scheme is semantically secure against adaptive chosen ciphertext attack (IND-IBBSC-CCA2) if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage Adv_{IBBSC}^{CCA2} in the following game.

1. The adversary $\mathcal{A}$ submits to the challenger the set of identities of the receivers $\mathcal{L} = \{ID_1, ID_2, \dots, ID_t\}$ on which he wishes to be challenged. The challenger $\mathcal{C}$ then runs *Setup*(k) and sends the system public parameters *params* to the adversary $\mathcal{A}$.
2. In the first phase, $\mathcal{A}$ makes polynomially bounded number of queries to the following oracles.
   (a) **Keygen Oracle** — $\mathcal{A}$ produces an identity ID and queries for the secret key of ID. The *Keygen Oracle* returns S_{ID} to $\mathcal{A}$.
   (b) **Signcrypt Oracle** — $\mathcal{A}$ produces a message m, broadcaster identity ID_A and a list of receiver identities $ID_1, ID_2, \dots, ID_t$. $\mathcal{C}$ returns $\sigma = Signcrypt\,(m, ID_A, \{ID_1, ID_2, \dots, ID_t\}, S_A)$, the signcrypted ciphertext, to $\mathcal{A}$, where the secret key S_A is computed from $Keygen(ID_A)$.
   (c) **Designcrypt Oracle** — $\mathcal{A}$ produces a broadcaster identity ID_A, receiver identity ID_i and a signcryption σ. The challenger $\mathcal{C}$ returns to $\mathcal{A}$, the result of $Designcrypt\,(\sigma, ID_A, ID_i, S_i)$, where the secret key S_i is computed from $Keygen(ID_i)$. The result returned is $\perp$ if σ is an invalid signcrypted ciphertext from ID_A to ID_i.

3. $\mathcal{A}$ produces two messages m_0 and m_1 of equal length from the message space $\mathcal{M}$ and an arbitrary broadcaster identity ID_A. The adversary must not have queried any of the t receivers' secret keys. The challenger $\mathcal{C}$ flips a coin, sampling a bit $b \leftarrow \{0,1\}$ and obtains the challenge signcrypted ciphertext by running $Signcrypt\left(m_b, ID_A, \{ID_1, ID_2, \ldots, ID_t\}, S_A\right)$, which is returned to $\mathcal{A}$.
4. $\mathcal{A}$ is allowed to make polynomially bounded number of new queries as in Step 2 with the restrictions that it should not query the *Designcryption Oracle* for the designcryption of σ^* or the *Keygen Oracle* for the secret keys of $ID_1, ID_2, \ldots, ID_t$.
5. Finally, $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$. The advantage of the adversary in winning this game is given by $Adv_{IBBSC}^{CCA2} = |Pr[b' = b] - \frac{1}{2}|$

We mention that this model of security takes into account collusion resistance too, because we provide the adversary with the secret keys of every user of the system except the ones he attacks.

Existential Unforgeability under Adaptive Chosen Message Attack for IBBSC (EUF-IBBSC-CMA). An ID-based broadcast signcryption scheme is existentially unforgeable under adaptive chosen message attack (EUF-IBBSC-CMA) if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage Adv_{IBBSC}^{CMA} in the following game.

1. The adversary $\mathcal{A}$ submits to the challenger, the identity of the target broadcaster ID_A whose signcryption he aims to forge (for an arbitrary message). The challenger $\mathcal{C}$ then runs $Setup(k)$ and sends the system public parameters *params* to the adversary $\mathcal{A}$.
2. In the first phase, $\mathcal{A}$ makes polynomially bounded number of queries to the following oracles.
   (a) **Keygen Oracle** — $\mathcal{A}$ produces an identity ID and queries for the secret key of ID. The *Keygen Oracle* returns S_{ID} to $\mathcal{A}$.
   (b) **Signcrypt Oracle** — $\mathcal{A}$ produces a message m, broadcaster identity ID_A and a list of receiver identities $ID_1, ID_2, \ldots, ID_t$. $\mathcal{C}$ returns $\sigma = Signcrypt\left(m, ID_A, \{ID_1, ID_2, \ldots, ID_t\}, S_A\right)$, the signcrypted ciphertext, to $\mathcal{A}$, where the secret key S_A is computed from $Keygen(ID_A)$.
   (c) **Designcrypt Oracle** — $\mathcal{A}$ produces a broadcaster identity ID_A, receiver identity ID_i and a signcryption σ. The challenger $\mathcal{C}$ returns to $\mathcal{A}$, the result of $Designcrypt\left(\sigma, ID_A, ID_i, S_i\right)$, where the secret key S_i is computed from $Keygen(ID_i)$. The result returned is $\perp$ if σ is an invalid signcrypted ciphertext from ID_A to ID_i.
3. $\mathcal{A}$ produces a signcrypted ciphertext σ from the broadcaster ID_A to the list of his receivers $\mathcal{L}$ and wins the game if the private key of broadcaster ID_A was not queried and $\perp$ is not returned by $Designcrypt(\sigma, ID_A, ID_i, S_i)$ for any $ID_i \in \mathcal{L}$ and σ is not the output of a previous query to the *Signcrypt Oracle*. The advantage of the adversary in winning this game is given by $Adv_{IBBSC}^{CMA} = Pr[Designcrypt(\sigma, ID_A, ID_i, S_i) \neq \perp \mid ID_i \in \mathcal{L}]$

We mention that this model of security takes into account collusion resistance too, because we allow the adversary to query for the secret keys of any entity.

4 Overview of IBBSC Scheme of Mu et al.

Mu et al.'s IBBSC scheme [5] consists of the three algorithms *Keygen* (which includes *Setup* as well), *Encrypt* and *Decrypt*, which we describe below.

1. ***KeyGen***(k, n_b, n)**.** Here, k is a security parameter, n_b is the number of broadcasters and n is the number of users in the system.
   (a) **Broadcaster Setup**
      i. Select master private keys $s_i \in \mathbb{Z}_q$ for all broadcasters B_i $(i = 1, \dots, n_b)$.
      ii. Select three strong public one-way hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1, H_2 : \{0,1\}^* \rightarrow \{0,1\}^k, H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q$.
      iii. Extract the public keys $Q_{B_i} \leftarrow H_1(ID_{B_i})$, where ID_{B_i} $(i = 1, \dots, n_b)$ are the unique identifiers of broadcasters.
      iv. Compute the private keys of the broadcasters $S_{B_i} \leftarrow s_i Q_{B_i}$ and $\bar{S}_{B_i} \leftarrow s_i P$.
   (b) **User Setup**
      i. Select $x_i \in \mathbb{Z}_q, i = 1, \cdots, n$ and assign each of $\{x_i\}$ to a user. Assign $\{x_i\}_{1 \le i \le n}$ to all broadcasters.
2. ***Signcrypt***$(\mathcal{L}, ID_{B_i}, m)$**.** Here, $\mathcal{L} = \{1, 2, \dots, t\}$ where $t \le n$ is the number of privileged users to whom broadcaster B_i wishes to send a message $m \in \mathbb{Z}_q$. Without loss of generality, we have assumed it is the first t members that are privileged.
   (a) Compute the following.
      i. $\prod_{j=1}^{t} (x - x_j)$, generating a polynomial function $f(x) = \sum_{\ell=0}^{t} c_\ell x^\ell$
      ii. $P_0 \leftarrow r(c_0 P + \bar{S}_{B_i}), P_1 \leftarrow c_1 rP, \dots, P_t \leftarrow c_t rP$
      iii. $k \leftarrow \hat{e}(P, r^2 S_{B_i})$
      iv. $y \leftarrow m \oplus H_3(k)$
      v. $X \leftarrow r(r - H_2(y)) S_{B_i}$
   (b) Broadcast $(y, X, ID_{B_i}, P_0, \dots, P_t)$
3. ***Designcrypt***$(y, X, ID_{B_i}, ID_j, x_j, P_0, \dots, P_t)$**.** Here, $(y, X, ID_{B_i}, P_0, \dots, P_t)$ is the ciphertext received by a member with identity ID_j whose secret value is x_j.
   (a) Compute $D \leftarrow \sum_{\ell=0}^{t} x_j^\ell P_\ell$.
   (b) Compute $k \leftarrow \hat{e}(P, X) \cdot \hat{e}(D\ ,\ H_2(y) Q_{B_i})$.
   (c) Compute $m \leftarrow H_3(k) \oplus y$.

5 Attack on IBBSC Scheme of Mu et al.

Mu et al. claimed that their scheme provides both confidentiality and unforgeability. They prove the former rigorously, but do not give the formal proof for unforgeability. We show in this section that their scheme is universally forgeable which is a major attack. Once a legitimate user gets a ciphertext from the broadcaster (intended for a set of users) and decrypts it (being a member of the intended set), he can generate a valid ciphertext for any message m^* as if it were generated by the broadcaster for the same set of users. We describe how this attack proceeds in this section.

Let *Alice* be a broadcaster with identity ID_{B_i} of the system and *Eve* be any legitimate user. *Eve* has just received a ciphertext, say $(y, X, ID_{B_i}, P_0, \cdots, P_t)$ and decrypts it (we assume that *Eve* is present in the set $\mathcal{L} = \{1, 2, \ldots, t\}$). If *Eve* wants to generate the ciphertext of any message m^* as if it were generated by *Alice* for the same list $\mathcal{L}$, with identities $ID_1, ID_2, \ldots, ID_t$, *Eve* just has to do the following.

1. As a result of decrypting the ciphertext, *Eve* gets the value of $D = r\bar{S}_{B_i}$.
2. Choose $r^* \in_R \mathbb{Z}_q^*$ and compute the following.
   (a) $P^* = P_0 - D + r^*P = rC_0p + r^*P$
   (b) $k^* = \hat{e}(r^*P, P)$
   (c) $y^* = m^* \oplus H_3(k)$
   (d) $X^* = r^*P - r^*H_2(y)Q_{B_i}$
3. $(y^*, X^*, ID_{B_i}, P_0^*, P_1, P_2, \ldots, P_t)$ is now a valid ciphertext of *Alice* for the message m^* generated by *Eve* for the list of users $\mathcal{L}$ with identities $\{ID_j\}_{1 \le j \le t}$

We now prove that the ciphertext generated by *Eve* is indeed a valid ciphertext from *Alice* to the receivers in $\mathcal{L}$ on the message m^*.

Decrypt$(y^*, X^*, ID_{B_i}, ID_j, x_j, P_0^*, \ldots, P_t)$**.** A receiver with identity ID_j uses his secret value x_j to decrypt the ciphertext $(y^*, X^*, ID_{B_i}, P_0^*, P_1, P_2, \ldots, P_t)$ obtained from *Eve* as follows. He computes the following.

1. $D^* \leftarrow \sum_{\ell=0}^{t} x_j^{\ell} P_\ell = r^*P$
2. $k^* \leftarrow \hat{e}(P, X^*) \cdot \hat{e}(D^*\ ,\ H_2(y^*)Q_{B_i}) = \hat{e}(r^*P, P)$
3. $m^* \leftarrow H_3(k) \oplus y^*$

From this it is clear that *Eve* can succeed in generating a ciphertext for an arbitrary message m^* with *Alice* as sender and identities ID_j, $1 \le j \le t$ as receivers without knowing the secret key of *Alice* and only knowing a previous valid ciphertext from *Alice* to this set of users and its decryption.

6 Overview of IBBSC Scheme of Li et al.

The IBBSC scheme of Li et al. [4] consists of the following algorithms.

1. ***Setup*** The steps in the setup phase are given below.
   (a) The security parameter of the scheme is k. The trusted authority chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, where $|q| = k$, a generator P of $\mathbb{G}_1$, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and hash functions $H_0 : \{0,1\}^{k_1} \rightarrow \mathbb{G}_1$, $H_1 : \{0,1\}^{k_0+k_2} \rightarrow \mathbb{Z}_q^*$ and $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^{k_0+k_2}$, where k_0, k_1 and k_2 are the number of bits required to represent a $\mathbb{G}_1$ element, an identity and a message respectively. The master private keys are $s_i \in_R \mathbb{Z}_q^*$ and the master public keys are $P_{pub}^i = s_i P$, one for each broadcaster $i = 1, 2, \ldots, n_b$. The public parameters of this scheme are $\langle \mathbb{G}_1, \mathbb{G}_2, n_b, n, \hat{e}, P, P_{pub}^1, P_{pub}^2, \ldots, ..., P_{pub}^m, H_0, H_1, H_2 \rangle$. Here, n_b and n represent the number of broadcasters and the number of users respectively.
   (b) The public keys of the broadcasters B_i with identities ID_{B_i} are $Q_{B_i} = H_0(ID_{B_i})$ and their private keys are $S_{B_i} = s_i Q_{B_i}$.
   (c) Select $x_j \in \mathbb{Z}_q^*$ for $1 \leq j \leq n$ and assign each x_j to a user. Publish $\{x_j\}_{1 \leq j \leq n}$ to all broadcasters.
2. ***Signcrypt*** To sign a message m, broadcaster B_i does the following.
   (a) Select a subset of $\{x_j\}$ and denote it by $\left\{x_j'\right\}_{1 \leq j \leq t}$ and these elements are associated with the t users who are entitled to receive the message.
   (b) Compute $\Pi_{j=1}^t \left(x - x_j'\right)$ and generate a polynomial function $f(x) = \sum_{\ell=0}^{t} c_\ell x^\ell$. It is to be noted that $f(x_j') = 0$ for $1 \leq j \leq t$.
   (c) Choose $r \in_R \mathbb{Z}_q^*$ and compute $X = rQ_{B_i}$, $P_0 = r(c_0 P + S_{B_i})$, $P_1 = c_1 r P$, $\ldots$, $P_t = c_t r P$.
   (d) Compute $h_1 = H_1(X \| m)$ and $Z = (r + h_1) S_{B_i}$.
   (e) Compute $\omega = \hat{e}(rS_{B_i}, P)$.
   (f) Compute $y = H_2(\omega) \oplus (Z \| m)$.
   (g) Broadcast the signcrypted ciphertext $\langle X, y, ID_{B_i}, P_0, \ldots, P_t \rangle$.
3. ***Designcrypt*** The user j decrypts $\langle X, y, ID_{B_i}, P_0, \ldots, P_t \rangle$ by performing the following computations.
   (a) Compute $D = \sum_{\ell=0}^{t} \left(x_j'\right)^\ell P_\ell$.
   (b) Compute $\omega = \hat{e}(P, D)$.
   (c) Compute $Z \| m = y \oplus H_2(\omega)$.
   (d) Compute $h_1 = H_1(X \| m)$.
   (e) Return the message m if $\hat{e}(Z, P) = \hat{e}(P_{pub}^i\ ,\ X + h_1 Q_{B_i})$, else return $\perp$.

7 Attacks on IBBSC Scheme of Li et al.

The ID-based broadcast signcryption scheme proposed by Li et al. in [4] claims to resolve the problems of both authentication and confidentiality by combining broadcast encryption and signcryption. The authors fail to prove either claim formally. We show that this scheme actually provides neither confidentiality nor authentication. In fact, in both properties, the worst possible attacks (complete recovery of encrypted message and exposure of secret key) can be successfully mounted on this scheme.

7.1 Attack on Confidentiality

Anybody who has the public parameters of the system can decrypt the message that is sent by the broadcaster, not just the intended receiver. This is because, in the *Signcrypt* algorithm, the value ω is being calculated as $\omega = \hat{e}\left(rS_{B_i}, P\right)$. Li et al. claim that only a legitimate user can recover $D = rS_{B_i}$ by evaluating the summation and then use it to compute ω as $\omega = \hat{e}\left(D, P\right)$. But any adversary who eavesdrops on the broadcasted signcryption knows the value of $X = rQ_{B_i}$. Also, $P^i_{pub} = s_iP$ is a public parameter. So, an adversary need not find D from the summation to compute ω. He can simply compute ω as $\omega = \hat{e}\left(X, P^i_{pub}\right) = \hat{e}\left(rQ_{B_i}, s_iP\right) = \hat{e}\left(rS_{B_i}, P\right)$, because $S_{B_i} = s_iQ_{B_i}$. And once the adversary gets ω, it is trivial to retrieve the message as $Z\|m = y \oplus H_2(\omega)$.

7.2 Attack on Authentication

In this scheme the secret key of the broadcaster is exposed during designcryption process. Once the secret key of the broadcaster is obtained, any user can forge the signcryptions of the broadcaster. This is the worst possible attack that can be mounted on any scheme. If a message m is broadcasted to t users under that broadcaster, it can be shown that all the t users can compute the secret key S_{B_i} of the broadcaster. This *total break* can be achieved by doing the following.

Any legitimate user *Alice* can get the values of D and Z during the designcryption process of a valid signcrypted ciphertext by following the normal protocol as follows.

1. $D = \sum\limits_{\ell=0}^{t} (x'_{Alice})^{\ell} P_{\ell} = rS_{B_i}$.
2. Compute $\omega = \hat{e}(P, D)$.
3. $Z\|m = y \oplus H_2(\omega)$ from which he can obtain Z.

We know that $Z = (r + h_1)S_{B_i}$, therefore $Z - D = h_1S_{B_i}$. Using the component X of the ciphertext and the message m, the user computes $h_1 = H_1(X\|m)$. Now $S_{B_i} = h_1^{-1} \cdot (Z - D)$.

8 Improved ID-Based Broadcast Signcryption Scheme

In this section, we propose an improved IBBSC scheme as a fix for the faulty scheme of Li et al. We follow the framework of a general ID-based broadcast signcryption scheme that we presented in Section 3.1. The algorithms that comprise our scheme are described below.

Setup(k) — Let k be the security parameter of the system. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order q (where $|q| = k$) and let P be the generator of $\mathbb{G}_1$ and $\hat{e}$ be a bilinear map defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Let n_0, n_1, n_2 and n_3 denote the number of bits required to represent an identity, an element of $\mathbb{G}_1$, an element of $\mathbb{G}_2$ and a message respectively. Consider four hash functions $H_0 : \{0, 1\}^{n_1} \rightarrow \mathbb{Z}_q^*$,

$H_1 : \{0,1\}^{n_0} \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^{n_0+2n_1+n_3} \rightarrow \mathbb{Z}_q^*$, $H_3 : \{0,1\}^{n_2} \rightarrow \{0,1\}^{n_1+n_3}$. The master private keys are $s_i \in_R \mathbb{Z}_q^*$ and the master public keys are $P_{pub}^i = s_i P$, one for each broadcaster $i = 1, 2, \ldots, n_b$. $s \in_R \mathbb{Z}_q^*$ and $P_{pub} = sP$ function as a global master private and public keys. The public parameters of this scheme are $\langle \mathbb{G}_1, \mathbb{G}_2, n_b, n, \hat{e}, P, P_{pub}, P_{pub}^1, P_{pub}^2, \ldots, P_{pub}^m, H_0, H_1, H_2, H_3 \rangle$. Here, n_b and n represent the number of broadcasters and the number of users respectively.

Keygen(ID) — Depending on whether ID is the identity of a broadcaster or a user, this algorithm performs the following functions.

– The public keys of the broadcasters B_i with identities ID_{B_i} are $Q_{B_i} = H_1(ID_{B_i})$ and their private keys are $S_{B_i} = s_i Q_{B_i}$.
– The public keys of the users j with identities ID_j are $Q_j = H_1(ID_j)$ and their private keys are $S_j = sQ_j$.

In addition, when a user j joins or subscribes to a broadcaster B_i, he sends his precomputed (he would compute it the first time he subscribes to a broadcaster) secret value $x_j = H_0(S_j)$ to that broadcaster.

Signcrypt$(m, ID_{B_i}, \{ID_1, ID_2, \ldots, ID_t\}, S_{B_i})$ — To signcrypt a message m to t of his users with identities $ID_1, ID_2, \ldots ID_t$, a broadcaster B_i does the following.

1. Compute the polynomial $f(x) = \prod_{j=1}^{t} (x - x_j) = \sum_{j=0}^{t} c_j x^j$, where x_j was derived by the user j from his secret key — $x_j = H_0(S_j)$.
2. Choose $r \in_R \mathbb{Z}_q^*$ and compute the following.
   (a) $X = rQ_{B_i}$
   (b) $P_0 = rc_0 P + X, P_1 = rc_1 P, \ldots, P_t = rc_t P$
   (c) $h_2 = H_2(ID_{B_i} \| X \| P_0 \| m)$
   (d) $Z = (r + h_2) S_{B_i}$
   (e) $\omega = \hat{e}(X, s_i P)$
   (f) $y = H_3(\omega) \oplus (Z \| m)$
3. Broadcast the signcrypted ciphertext $\sigma = (y, ID_{B_i}, P_0, P_1, \ldots, P_t)$.

Designcrypt(σ, ID_j, S_j) — A user j, where $1 \leq j \leq t$, on receiving the signcrypted ciphertext $\sigma = (y, ID_{B_i}, P_0, P_1, \ldots, P_t)$, to designcrypt, he does the following.

1. He computes the following.
   (a) $X' = \sum \ell = 0^t P_\ell x_j^\ell$, where $x_j = H_0(S_j)$.
   (b) $\omega' = \hat{e}(X', s_i P)$
   (c) $Z' \| m' = y \oplus H_3(\omega')$
2. Return the message m' if $\hat{e}(Z', P) = \hat{e}(X' + H_2' Q_{B_i}, s_i P)$, where $h_2' = H_2(ID_{B_i} \| X' \| P_0 \| m')$.

It is easy to see that our scheme is correct. In the next section, we prove the confidentiality of our improved scheme. We prove the unforgeability of our scheme in the full version.

9 Proof of Confidentiality of Our IBBSC Scheme

Theorem. *Our improved ID-based broadcast signcryption scheme is secure against any IND-IBBSC-CCA2 adversary $\mathcal{A}$ under the random oracle model if CDHP is hard in $\mathbb{G}_1$.*

Proof. The challenger $\mathcal{C}$ receives an instance (P, aP, bP) of the CDH problem. His goal is to determine the value of abP. Suppose there exists an IND-IBBSC-CCA2 adversary $\mathcal{A}$ for our improved scheme. We show that $\mathcal{C}$ can use $\mathcal{A}$ to solve the CDH problem. $\mathcal{C}$ will set the random oracles $\mathcal{O}_{H_0}$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{KeyExtract}$, $\mathcal{O}_{Signcrypt}$ and $\mathcal{O}_{Designcrypt}$. The answers to the oracles $\mathcal{O}_{H_0}$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, and $\mathcal{O}_{H_3}$ are randomly selected; therefore, to maintain consistency, $\mathcal{C}$ will maintain four lists $L_0 = \langle R, \omega \rangle$, $L_1 = \langle ID_i, x_i, \hat{x}_i, S_i, Q_i \rangle$, $L_2 = \langle ID, X, P_0, m, h_2 \rangle$, and $L_3 = \langle \omega, h_3 \rangle$. The reasons for and meanings of the elements of these lists will become clear during the discussion of the corresponding oracles. We assume that $\mathcal{A}$ will ask for $H_1(ID)$ before ID is used in any key extraction, signcryption and designcryption queries. First, the adversary $\mathcal{A}$ outputs a list $\mathcal{L} = \{ID_1, ID_2, \ldots, ID_{t^*}\}$ of the members whom he proposes to attack. Let there be n_b broadcasters B_1, B_2, …, B_{n_b}. Then, the challenger $\mathcal{C}$ gives $\mathcal{A}$ the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, n_b, n, \hat{e}, P, P_{pub}, P^1_{pub}, P^2_{pub}, \ldots, P^{n_b}_{pub}, H_0, H_1, H_2, H_3 \rangle$, where $P_{pub} = aP$ and $s_i \in_R \mathbb{Z}_q^*$ and $P^i_{pub} = s_i P$. The descriptions of the oracles follow.

Oracle $\mathcal{O}_{\mathbf{H_0}}(\mathbf{R})$. This oracle scans the list L_1 and does the following for every entry of ID_i.

1. Check if ID_i's entry has an S_i that is equal R. If yes, then do the following.
   (a) Check if $ID_i \in \mathcal{L}$ and $\hat{e}(R, P) = \hat{e}(aP, bP)^{\hat{x}_i}$
   (b) If so, it means that the adversary is querying for the hash of the secret key of one of the users in $\mathcal{L}$. That is, $R = ab\hat{x}_i P$. The challenger then aborts this simulation, and returns the answer to the CDH problem as $R.\,\hat{x}_i^{-1}$.
   (c) If not, then return x_i.
2. If not, then browse through the list L_0 for any entry (R, h_0). If there is such an entry, then return h_0. If there is no such entry, then select a new[2] $h_0 \in_R \mathbb{Z}_q^*$, add the tuple (R, h_0) to L_0 and return h_0.

Oracle $\mathcal{O}_{\mathbf{H_1}}(\mathbf{ID})$. $\mathcal{C}$ checks if there exists a tuple $(ID, x_{ID}, \hat{x}_{ID}, S_{ID}, Q_{ID})$ in L_1. If such a tuple exists, $\mathcal{C}$ answers with Q_i. Otherwise, $\mathcal{C}$ does the following.

1. If ID is a broadcaster's identity, say ID_{B_i}, then choose a new $\hat{x}_{ID} \in_R \mathbb{Z}_q^*$ and $\hat{x}_{ID} \in_R \mathbb{Z}_q^*$, and set $Q_{ID} = \hat{x}_{ID} P$, $S_{ID} = \hat{x}_{ID} s_i P$.
2. If $ID \notin \mathcal{L}$ and is a user's identity, choose a new $x_{ID} \in_R \mathbb{Z}_q^*$ and $\hat{x}_{ID} \in_R \mathbb{Z}_q^*$, and set $Q_{ID} = \hat{x}_{ID} P$, $S_{ID} = \hat{x}_{ID} aP$.

[2] By new, we mean that the random value chosen must not have been already chosen during an earlier execution.

3. If $ID \in \mathcal{L}$, choose a new $x_{ID} \in_R \mathbb{Z}_q^*$ and $\hat{x}_{ID} \in_R \mathbb{Z}_q^*$, and set $Q_{ID} = \hat{x}_{ID}bP$, $S_{ID} = \perp$.
4. Add $(ID, x_{ID}, \hat{x}_{ID}, S_{ID}, Q_{ID})$ to the list L_1 and return Q_{ID}.

Oracle $\mathcal{O}_{\mathbf{H_2}}$ $(\mathbf{ID} \| \mathbf{X} \| \mathbf{P_0} \| \mathbf{m})$. $\mathcal{C}$ checks if there exists a tuple (ID, X, P_0, m, h_2) in L_2. If such a tuple exists, $\mathcal{C}$ returns h_2. Otherwise, $\mathcal{C}$ chooses a new $h_2 \in_R \mathbb{Z}_q^*$, adds the tuple (ID, X, P_0, m, h_2) to L_2 and returns h_2.

Oracle $\mathcal{O}_{\mathbf{H_3}}$ (ω). $\mathcal{C}$ checks if there exists a tuple (ω, h_3) in L_3. If such a tuple exists, $\mathcal{C}$ returns h_3. Otherwise, $\mathcal{C}$ chooses a new $h_3 \in_R \{0,1\}^{n_1+n_3}$, adds the tuple (ω, h_3) to L_3 and returns h_3.

Oracle $\mathcal{O}_{\mathbf{KeyExtract}}(\mathbf{ID})$. If L_1 does not contain an entry for ID, return $\perp$. Otherwise, $\mathcal{C}$ recovers the tuple $(ID, x_{ID}, \hat{x}_{ID}, S_{ID}, Q_{ID})$ from L_1 and returns S_{ID}.

Oracle $\mathcal{O}_{\mathbf{Signcrypt}}$ $(\mathbf{m}, \mathbf{ID_{B_i}}, \{\mathbf{ID_1}, \mathbf{ID_2}, \ldots, \mathbf{ID_t}\})$. On receiving this query, $\mathcal{C}$ checks if there is an entry for ID_{B_i} and all the ID_js in L_1. If not, then $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ retrieves $(ID_{B_i}, x_{B_i}, \hat{x}_{B_i}, S_{B_i}, Q_{B_i})$ from L_1, and executes ***Signcrypt***$(m, ID_{B_i}, \{ID_1, ID_2, \ldots, ID_t\}, S_{B_i})$ as usual and returns what the signcryption algorithm returns. For the signcryption algorithm, the challenger retrieves the entries of these t users from list L_1 and uses the x_j values in them.

Oracle $\mathcal{O}_{\mathbf{Designcrypt}}$ $(\sigma_{\mathbf{B_i}}, \mathbf{ID_j})$. The challenger $\mathcal{C}$, on receiving the signcryption $\sigma_{B_i} = (y, ID_{B_i}, y, P_0, P_1, \ldots, P_t)$, first checks if there are entries for ID_{B_i} and ID_j in L_1. If not, then $\mathcal{C}$ returns $\perp$. Otherwise, $\mathcal{C}$ retrieves the entry $(ID_j, x_j, \hat{x}_j, S_j, Q_j)$ from L_1 and executes ***Designcrypt***$(\sigma_{B_i}, ID_j, S_j)$ in the normal way and returns what the designcryption algorithm returns.

After the first query stage, $\mathcal{A}$ outputs two plaintext messages m_0 and m_1 of equal length and provides a broadcaster's identity ID_B. Now, $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$, retrieves $(ID_B, x_B, \hat{x}_B, S_B, Q_B)$ from L_1, and executes ***Signcrypt***$(m_b, ID_B, \mathcal{L}, S_B)$ as usual and returns what the signcryption algorithm returns as the challenge signcrypted ciphertext.

$\mathcal{A}$ can perform queries as above. However, it cannot query the designcryption oracle with the challenge signcryption. At the end of the simulation, $\mathcal{A}$ outputs a bit b' for which he believes that the challenge signcryption is the signcryption of $m_{b'}$ from ID_B to its subscribers. The only way the adversary can win the game, other than by guessing, is by decrypting the signcrypted ciphertext using one of the t users' secret keys. This is because, to gain any information from the ciphertext y, he needs to know about ω, for which he must know the x_j of one of the users. And since x_j is a result of a hash function, he must know the input to the hash function, namely the secret key of the user j. So, if the adversary has a non-negligible advantage of winning this game, then with a non-negligible probability, he must have queried the oracle $\mathcal{O}_{H_0}$ with the secret key of one of the t users. And, once this query is made, the challenger can get the value of abP, as described in the description of $\mathcal{O}_{H_0}$ above.

10 Proof of Unforgeability of Our IBBSC Scheme

Theorem. *Our ID-based broadcast signcryption scheme is secure against any EUF-IBBSC-CMA adversary $\mathcal{A}$ under the random oracle model if CDHP is hard in $\mathbb{G}_1$.*

Proof. The proof is presented in the full version of this paper.

11 Efficiency of Our IBBSC Scheme

In this section, we discuss the efficiency of the improved IBBSC scheme. The major parameters involved are the computation costs for *signcryption* and *designcryption* operations, the communication cost and the storage at the user's end. For computational cost, we consider the number of pairing computations performed, as they are the costliest operations involved. Our improved scheme performs one pairing computation during *Signcrypt* and three per user during *Designcrypt*, which is the same as that of Li et al.'s scheme, and one more than that of Mu et al.'s scheme. For the communication cost, we still have to broadcast $O(t)$ group elements if the number of subscribers is t. Coming to storage cost, we consider the storage at both the broadcaster and user. The broadcaster has to store information about every subscriber and so the storage cost for him is $O(t)$. Users do not have to store anything other than their secret keys and precomputed secrets. An added advantage of our scheme over Li et al.'s scheme is that users have the option of registering to their preferred subscribers, whereas in Li et al.'s scheme, the x_j values of users are published to all broadcasters in the *Setup* phase itself. Also, no matter how many broadcasters a user j subscribes to, he need only maintain one precomputed secret value x_j.

12 Conclusion

In this paper, we have considered the problem of secure and authenticated content distribution over large networks, especially wireless networks, which, on one hand, are increasingly becoming popular choices for the modern civilization, what with the advent of mobile and portable devices such as cell phones and PDAs, and on the other hand, are much easier to eavesdrop than wired networks. ID-based broadcast signcryption schemes provide the solution to this problem and in the context of mobile devices being the computers at the end users, the efficiency of such schemes becomes very important — there is limited memory and computational power that is available. First, we have demonstrated an existential forgery attack on Mu et al.'s scheme and a total break of Li et al.'s scheme. Following this, we have proposed an improved IBBSC scheme to fix the security leak of Li et al's scheme and also proven its IND-CCA2 and EUF-CMA security formally in the random oracle model. These are the strongest existing security notions for message confidentiality and authentication respectively. Our scheme performs no worse than Li et al.'s scheme. In fact, we enhance the flexibility of their scheme by allowing users to join broadcasters of their choice.

Future Work. Our improved scheme suffers from the fact that the size of the signcryption that is to be broadcasted is linear in the number of privileged users to whom the broadcaster intends to send the content. In the context of mobile networks, it would be a phenomenal improvement if this can be made constant size or even logarithmic in the number of privileged users. When looking at the number of pairing computations that are involved in the scheme, it is worthwhile to see if the number of pairing computations can be further reduced during designcryption, though it seems unlikely to be able to do so without compromising the security of the system. Another drawback of this scheme is that the coefficients of a t-degree polynomial have to be evaluated everytime a signcryption operation is done. It would be nice to have an IBBSC scheme where this calculation is avoided.

References

1. Aparna, R., Amberker, B.B.: Authenticated secure group communication using broadcast encryption key computation. In: ITNG 2008: Fifth International Conference on Information Technology - New Generations, pp. 348–353 (April 2008)
2. Bohio, M.J., Miri, A.: An authenticated broadcasting scheme for wireless ad hoc network. In: 2nd Annual Conference on Communication Networks and Services Research (CNSR), pp. 69–74 (2004)
3. Kanazawa, F., Ohkawa, N., Doi, H., Okamoto, T., Okamoto, E.: Broadcast encryption with sender authentication and its duality. In: International Conference on Convergence Information Technology 2007, pp. 793–798 (November 2007)
4. Li, F., Xin, X., Hu, Y.: Indentity-based broadcast signcryption. Computer Standards and Interfaces 30(1-2), 89–94 (2008)
5. Mu, Y., Susilo, W., Lin, Y.-X., Ruan, C.: Identity-based authenticated broadcast encryption and distributed authenticated encryption. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, pp. 169–181. Springer, Heidelberg (2004)
6. Sharmila Deva Selvi, S., Sree Vivek, S., Naresh Karuturi, N., Gopalakrishnan, R., Pandu Rangan, C.: Cryptanalysis of bohio et al.'s id-based broadcast signcryption scheme for wireless ad-hoc networks. In: Proceedings of Sixth Annual Conference on Privacy, Security and Trust, PST 2008 (2008)
7. How Tan, C., Ming Teo, J.C., Amundsen, J.-A.: Authenticated broadcast encryption scheme. In: AINAW 2007: 21st International Conference Advanced Information Networking and Applications Workshops, vol. 1, pp. 512–518 (May 2007)

Sanitizable and Deletable Signature*

Tetsuya Izu[1], Noboru Kunihiro[2],
Kazuo Ohta[3], Makoto Sano[3], and Masahiko Takenaka[1]

[1] FUJITSU LABORATORIES Ltd.C
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
{izu,takenaka}@labs.fujitsu.com
[2] The University of Tokyo,
5-1-5, Kashiwanoha, Kashiwa, 277-8561, Japan
kunihiro@k.u-tokyo.ac.jp
[3] The University of Electro-Communications,
1-5-1, Chofugaoka, Chofu, 182-8585, Japan
ota@ice.uec.ac.jp

Abstract. Recently, the sanitizable signature attracts much attention since it allows to modify (sanitize) the document for hiding partial information without keeping the integrity of the disclosed subdocuments. Sanitizable signatures are quite useful in governmental or military offices, where there is a dilemma between disclosure laws for public documents and privacy or diplomatic secrets. Since a verifier can detect whether the document was sanitized or not, especially which subdocuments was sanitized, the scheme does not establish the perfect hiding. In order to solve the problem, the deletable signature was introduced in 2006. However, because these schemes are not compatible to each other, we have to select the scheme to meet the requirement. In this paper, we propose the *sanitizable and deletable signature* as a combination of the sanitizable signature and the deletable signature. We also establish two concrete sanitizable and deletable signatures based on the deletable signature by Miyazaki, Hanaoka and Imai.

Keywords: Digital signature, sanitizable signature, deletable signature, aggregate signature.

1 Introduction

Recently, governmental entities are forged to disclose documents because of the disclosure laws. When the document has privacy or diplomatic secrets, in old day, physical maskings were used for hiding such secrets. However, its analogy for digital documents are not established yet. In addition, in these days, digital documents are stored with digital signatures in order to assure the integrity of documents. Since current signature schemes can not distinguish such appropriate alternations on the original document from inappropriate alternations (forgeries), a direct analogy of physical masking does not work well.

* This research was done while the first author was partially, and the second author was fully in the University of Electro-Communications.

In order to solve the problem, the *sanitizable signature* was introduced in 2001 [11], in which, after generating a signer's signature on an original document, specific entities (called *sanitizers*) can modify the document for hiding partial information and generate sanitized documents. A verifier confirms the integrity of disclosed parts of the sanitized document from the signature and the sanitized document. In addition, the secrecy of sanitized parts is assured, namely, no information of sanitized parts will be leaked after the sanitizations. Sanitized signatures are so attractive that many constructions have been proposed [1, 4, 5, 6, 7, 10, 11, 12]. In the sanitizable signature schemes, a verifier can detect whether the document is sanitized or not, moreover, which subdocuments are sanitized. In other words, the hiding of the sanitizable signature is not perfect. The *deletable signature*, introduced in 2006 [9], is a digital signature scheme in which a subdocument (and corresponding data) can be deleted without keeping the integrity of the remaining subdocuments. Since these schemes are not compatible to each other, we have to select the scheme to meet various requirements. In addition, we cannot sanitize and delete on the same document even with these schemes.

Contribution of this Paper

In this paper, we introduce the *sanitizable and deletable signature* as a combination of the sanitizable signature and the deletable signature, in which, after generating a signer's signature on an original document (an ordered set of subdocuments), specific entities, called *revisers*, are allowed to sanitize or delete subdocuments for hiding partial information. A verifier checks the integrity of the disclosed subdocuments. In addition, the secrecy of sanitized or deleted subdocuments is established. We also establish two sanitizable and deletable signatures SDS1 and SDS2 based on the deletable signature by Miyazaki, Hanaoka and Imai [9]. In the proposed schemes, each subdocument has a subdocument status such as SADP (Sanitization Allowed and Deletion Prohibited) or SDA (Sanitized and Deletion Allowed). Among possible seven subdocument status, SDS1 supports four (SPDP, SADA, SDA and D) while SDS2 supports six (SPDP, SADP, SDP, SADA, SDA and D).

2 Preliminaries

This section introduces some signature schemes including the general aggregate signature by Boneh, Gentry, Lynn and Shacham [2] and the deletable signature by Miyazaki, Hanaoka and Imai [9].

In this paper, $\mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T$ are multiplicative cyclic groups with order p (prime) and $g_1,\ g_2$ are generators of $\mathbb{G}_1,\ \mathbb{G}_2$ (namely, $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$). We assume that the Computational Diffie-Hellman (CDH) problem in these groups are hard. Let e be a bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ such that $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in \mathbb{G}_1,\ v \in \mathbb{G}_2,\ a, b \in \mathbb{Z}$ (bilinearity) and $e(g_1, g_2) \neq 1$ (non-degeneracy). We also use two cryptographic hash functions $H_0 : \{0,1\}^* \to \{0,1\}^\ell$ and $H : \{0,1\}^* \to \mathbb{G}_2$. H_0 is a standard hash function and we assume that a certain value of ℓ

Algorithm 1. A description of BGLS's aggregate signature

```
KeyGen (of the $i$-th signer)
1. Generate $\mathsf{sk}_i \overset{R}{\leftarrow} \mathbb{Z}/p\mathbb{Z}$ randomly and set $\mathsf{pk}_i \leftarrow g_2^{\mathsf{sk}_i}$.
Output: A secret and public key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{G}_2$

Sign (by the $i$-th signer)
Input: A document $M_i \in \{0,1\}^*$ and a secret key $\mathsf{sk}_i \in \mathbb{Z}/p\mathbb{Z}$
1. Set $\sigma_i \leftarrow H(M_i)^{\mathsf{sk}_i}$.
Output: An individual signature $\sigma_i \in \mathbb{G}_1$

Agg
Input: Individual signatures $\sigma_{j_1}, \ldots, \sigma_{j_k} \in \mathbb{G}_1$, an aggregate signature $\sigma \in \mathbb{G}_1$
1. Set $\sigma' \leftarrow \sigma \times \sigma_{j_1} \times \cdots \times \sigma_{j_k}$.
Output: An aggregate signature $\sigma' \in \mathbb{G}_1$

AggVerify
Input: Documents $M_1, \ldots, M_n \in \{0,1\}^*$, an aggregate signature $\sigma \in \mathbb{G}_1$ and signers' public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n \in \mathbb{G}_2$
1. Check whether $e(\sigma, g_2) = \prod_{i=1}^{n} e(H(M_i), \mathsf{pk}_i)$ holds. If not, output invalid and terminate, otherwise output valid and terminate.
```

is provided implicitly in the following. For a construction of H in the random oracle model, see [3].

2.1 Aggregate Signature

An *aggregate signature* is a digital signature in which the special party, called the *aggregator*, is allowed to compress n signatures generated by n signers on n documents into a signature, called the *aggregate signature*, with the (almost) same size. A concept of the aggregate signature is introduced by Boneh, Gentry, Lynn, and Shacham [2]. They also provided a concrete scheme from bilinear maps in the same paper as a natural extension of the short signature by Boneh, Lynn and Shacham [3]. Since a generation of signers' signatures and an aggregation of signatures are proceeded in separate algorithms, their scheme is called the *general aggregate signature*. On the other hand, Lysyanskaya, Micali, Reyzin, and Shacham provided another aggregate signature scheme from trap-door permutations [8]. Since a generation of signers' signatures and an aggregation are proceeded in the same algorithm, their scheme is called the *sequential aggregate signature*. One of the distinguishing property between the general and the sequential schemes is that, when an aggregate signature is valid, sequential verifiers can obtain aggregate signatures output by signers while general verifiers can not.

Algorithm 1 shows a concrete description of BGLS's general aggregate signature, which is the main tool for our sanitizable and deletable signature. Here, documents $M_1, \ldots, M_n$ should be distinct in order to avoid the potential attack

[2]. Since the aggregate signature σ is multiplicative, one can remove an individual signature σ_i from the aggregate signature σ (in which σ_i is compressed): compute σ/σ_i.

Define some notions for describing the security. A co-CDH problem is a problem to compute $h^a \in \mathbb{G}_2$ from given $g_1,\ g_1^a \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, while a co-DDH problem is a decision problem to determine whether $a = b$ or not from given $g_1,\ g_1^a \in \mathbb{G}_1$ and $h,\ h^b \in \mathbb{G}_2$. These problems are generalizations of standard CDH, DDH problems. A group pair $(\mathbb{G}_1, \mathbb{G}_2)$ is called a co-GDH pair if co-CDH problem is hard, but co-DDH problem is easy. A co-GDH assumption is an assumption that $(\mathbb{G}_1, \mathbb{G}_2)$ is a co-GDH pair. In fact, in BGLS's scheme, since a bilinear map e is defined over $\mathbb{G}_1 \times \mathbb{G}_2$, co-DDH problem is easily solved.

Let us consider the following game where an adversary $\mathcal{A}$ attempts to forge an aggregate signature in BGLS's scheme: in the setup, $\mathcal{A}$ receives a randomly generated public key $\mathsf{pk}_1 \in \mathbb{G}_1$. Then, $\mathcal{A}$ requests signatures with pk_1 on adaptively-chosen messages to the signing oracle. Finally, $\mathcal{A}$ outputs $n-1$ additional public keys $\mathsf{pk}_2, \ldots, \mathsf{pk}_n$, messages $M_1, \ldots, M_n$ and an aggregate signature σ. If σ is valid over $M_1, \ldots, M_n, \mathsf{pk}_1, \ldots, \mathsf{pk}_n$, and M_1 is not trivial (namely, $\mathcal{A}$ did not request a signature on M_1 with pk_1), we consider that the adversary wins the game (the general aggregate chosen-key model [2]). It is proved that BGLS's aggregate signature is secure in the general aggregate chosen-key model under co-GDH assumption, namely $\mathcal{A}$'s advantage over coin tosses is negligible. For further security discussions, see [2].

2.2 Sanitizable Signature

A *sanitizable signature* is digital signature in which, after generating a signer's signature on an original document (an ordered set of subdocuments), specific entities, called *sanitizers*, are allowed to modify the document by replacing the contents of subdocuments into their hash values for hiding partial information. A verifier checks the integrity of disclosed subdocuments of the sanitized document from the signature and the sanitized document. In addition, the secrecy of closed subdocuments is established (no information of sanitized subdocuments will be leaked after the sanitizations). The sanitizable signature is so attractive in many applications that many constructions have been proposed up to the moment [1, 4, 5, 6, 7, 9, 10, 11, 12].

These sanitizable signature schemes have a common structure: for a document $M = (M_1, \ldots, M_n)$ be a document, a signer's signature is generated on a concatenation $h = H_0(M_1)||\cdots||H_0(M_n)$. When a sanitizer sanitizes a subdocument M_i, he replaces M_i by $H_0(M_i)$. Since a verifier can obtain all $H_0(M_i)$, the verification will be proceeded. However, because of this structure, it was hard to delete the subdocument M_i: the sanitizer cannot delete the corresponding information from the signer's signature.

2.3 Deletable Signature

The *deletable signature* is a digital signature in which, after generating a signer's signature on an original document, specific entities, called *deleters*, are allowed

Algorithm 2. A description of MHI's deletable signature

```
KeyGen
1. Generate $\mathsf{sk} \overset{R}{\leftarrow} \mathbb{Z}/p\mathbb{Z}$ randomly and set $\mathsf{pk} \leftarrow g_2^{\mathsf{sk}}$.
Output: A secret and public key pair $(\mathsf{sk}, \mathsf{pk}) \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{G}_2$

Sign
Input: A document $\{M_1, \ldots, M_n\}$ and a secret key $\mathsf{sk} \in \mathbb{Z}/p\mathbb{Z}$
1. Generate a random value $ID$ (as a document ID) and set $\bar{M}_0 \leftarrow ID,\ \bar{M}_i \leftarrow ID||M_i$ for each subdocument $\bar{M}_i$ $(i = 1, \ldots, n)$.
2. Generate an indivisual signature $\sigma_i \leftarrow H(\bar{M}_i)^{\mathsf{sk}}$ for each padded subdocument $\bar{M}_i$ $(i = 0, \ldots, n)$.
3. Generate an aggregate signature $\sigma \leftarrow \prod_{i=0}^{n} \sigma_i$.
4. Set an initial value of index sets $P,\ A \in \{1, \ldots, n\}$ such that $P \cup A = \{1, \ldots, n\}$ and $P \cap A = \phi$. The index 0 should be included in $P$.
Output: A document $\{\bar{M}_0, \ldots, \bar{M}_n\}$, indivisual signatures $\sigma_1, \ldots, \sigma_n$, an aggregate signature $\sigma \in \mathbb{G}_1$ and index sets $P,\ A$

Delete
Input: A document $\{\bar{M}_0, \ldots, \bar{M}_k\}$, indivisual signatures $\{\sigma_i\}_{i \in A}$, an aggregate signature $\sigma$ and index sets $P,\ A$
1. Determine new index sets $P' \supseteq P,\ A' \subseteq A,\ D' \subseteq A$ such that $P' \cup A' \cup D' = \{1, \ldots, k\}$ and $P' \cap A' = A' \cap D' = D' \cap P' = \phi$.
2. Do the following procedure for each index $i$:
    – Deletion: if $i \in D'$, update $\sigma \leftarrow \sigma/\sigma_i$ and delete $\bar{M}_i,\ \sigma_i$.
    – Prohibition: if $i \in P' \backslash P$, delete $\sigma_i$.
    – Otherwise: do nothing.
3. Renumber the indexes.
Output: A document $\{\bar{M}_0, \ldots, \bar{M}_{k'}\}$, indivisual signatures $\{\sigma_i\}_{i \in A'}$, an aggregate signature $\sigma' \in \mathbb{G}_1$ and index sets $P',\ A'$.

Verify
Input: A document $\{\bar{M}_0, \ldots, \bar{M}_k\}$, indivisual signatures $\{\sigma_i\}_{i \in A}$, an aggregate signature $\sigma$, index sets $P,\ A$ and a signer's public key $\mathsf{pk}$
1. Check whether the document identifiers in each subdocument is same. If not, output invalid and terminate.
2. Check whether $e(\sigma, g_2) = \prod_{i=0}^{k} e(H(\bar{M}_i), \mathsf{pk})$ hold. If not, output invalid and terminate.
3. Check whether $e(\sigma_i, g_2) = e(H(\bar{M}_i), \mathsf{pk})$ hold for subdocuments $\bar{M}_i$ $(i \in A)$. If not, output invalid and terminate.
4. Output valid and terminate.
```

to delete subdocuments for hiding partial information. A verifier checks the integrity of the remaining document. In addition, the secrecy of deleted subdocuments is established. However, different from the sanitizable signature, a verifier cannot detect whether subdocuments were deleted or not.

The only deletable signature scheme, based on BGLS's aggregate signature, was proposed by Miyazaki, Hanaoka and Imai [9]. Algorithm 2 shows a concrete description of MHI's deletable signature, where index sets P, A, D are subsets of the set of all indexes $\{1, \ldots, k\}$ (k is the number of current subdocuments). Here, the document ID is used not to copy subdocument and corresponding individual signature in other documents generated by the same signer. In addition to the deletable property, MHI's deletable signature can control subdocument status in the sense that deletion is 'Prohibited (P)' or 'Allowed (A)'. If a subdocument is in the status 'P', it cannot be deleted, while a subdocument in 'A' can. If we regard the deleted subdocument as a status, MHI's deletable signature controls three status P, A and D for each subdocument.

Another distinguishing feature of MHI's deletable signature is the target document: a document is assumed to be a set of subdocuments (rather than an ordered set). Thus, two documents 'Taro and Jiro' and 'Jiro and Taro' are considered the same document in the scheme, and the security is discussed and proved for such documents. There may exist an application which uses such documents, however, it is a fact that there exits various applications in which these two documents are treated differently. In fact, the target documents of previously proposed sanitizable signatures are ordered subdocuments [4, 10].

3 Proposed Schemes

In this section, we propose a concept of the sanitizable and deletable signature. We also establish two sanitizable and deletable signatures SDS1 and SDS2.

3.1 Concept

As described in section 2, the sanitizable signature and the deletable signature are used for hiding partial information with keeping the integrity of disclosed parts. Since these schemes are not compatible to each other, we have to select the scheme to meet various requirements. The *sanitizable and deletable signature* is a combination of the sanitizable signature and the deletable signature. In the scheme, after generating a signer's signature on an original document (an ordered set of subdocuments), specific entities, called *reviser*, are allowed to sanitize or delete subdocuments for hiding partial information. A verifier checks the integrity of the disclosed subdocuments. In addition, the secrecy of sanitized or deleted subdocuments is established.

The sanitizable and deletable signature is consists of four algorithms Key Generator (KeyGen), Signer (Sign), Reviser (Revise) and Verifier (Verify). KeyGen generates a signer's secret and public key pair. Sign publishes a signature on an original document, while Revise sanitizes or deletes subdocuments. Multiple revisers are assumed. Finally, Verify checks the integrity of disclosed subdocuments of the revised document.

3.2 Approach

In order to construct the sanitizable and deletable signature, we choose the deletable signature as a based algorithm, since the common structure of the sanitizable signatures is not suitable for subdocument deletions.

There are two problems for constructing the sanitizable and deletable signature based on MHI's deletable signature. The first problem is that, in MHI's signature, the order of subdocuments are not considered. For this problem, we introduce subdocument ID for each subdocument (in addition to the document ID): for a given subdocument M_i, a signer generates the document ID and the subdocument ID_i and sets $\bar{M}_i \leftarrow ID||ID_i||M_i$. For a document $M = (M_1, \ldots, M_n)$, subdocuments IDs are generated increasingly (or decreasingly). Since the order changes are detected by the subdocument ID, MHI's scheme can handle the ordered subdocuments.

The other possible problem is that, a subdocument $\bar{M}_i = ID||ID_i||M_i$ is sanitized, it may be natural to replace it by its hash value $H_0(\bar{M}_i)$. However, since the document ID and subdocument ID are lost, the copy from other documents or the order change are possible (the upper figure in Figure 1). If we replace by $ID||ID_i||H_0(\bar{M}_i)$, since ID and ID_i can be forged by an adversary, the integrity of the document ID and the subdocument ID are not established. In the proposed schemes, a signer publishes an individual signature on $ID||ID_i||H_0(\bar{M}_i)$ rather than $ID||ID_i||\bar{M}_i$ (namely, $\sigma_i \leftarrow H(ID||ID_i||H_0(\bar{M}_i))^{\mathsf{sk}}$). When a reviser sanitizes the subdocument $\bar{M}_i$, he replaces it by $ID||ID_i||H_0(\bar{M}_i)$. Since the integrity of ID and ID_i are verified by the individual σ_i, above mentioned attacks can be avoided (the lower figure in Figure 1).

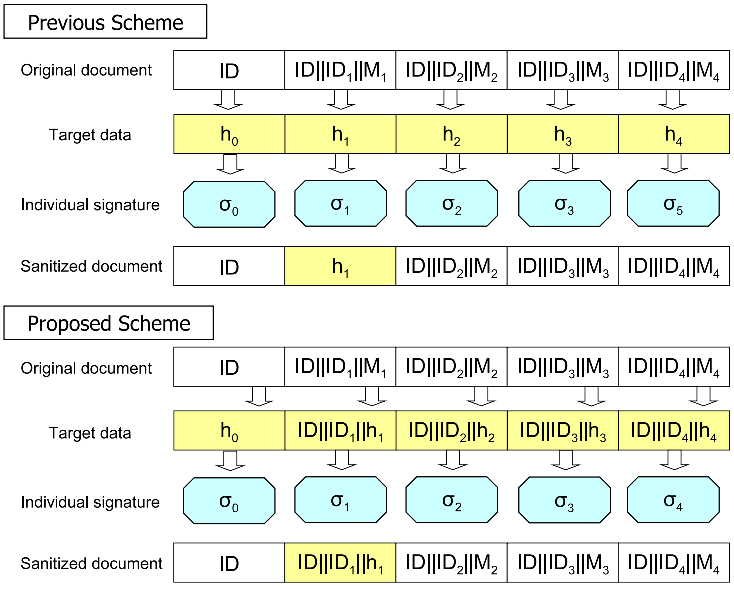
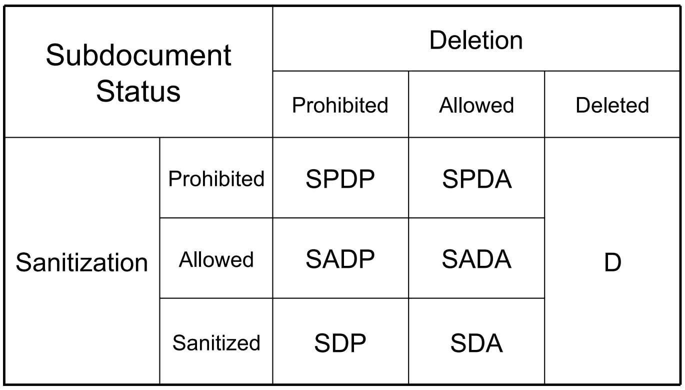


Fig. 1. The document ID and the subdocument ID



Subdocument Status		Deletion		
		Prohibited	Allowed	Deleted
Sanitization	Prohibited	SPDP	SPDA	D
	Allowed	SADP	SADA	
	Sanitized	SDP	SDA	

Fig. 2. Subdocument status

3.3 Subdocument Status

In addition to the sanitizable and deletable properties, our schemes control subdocument status similar to MHI's scheme. We have three status Prohibited, Allowed and Sanitized (Deleted) for sanitizations and deletion, and thus there are nine possible combinations. However, when the subdocument is deleted, status on sanitization does not make a sense. Thus, in the sanitizable and deletable signature, the following seven subdocument status can be defined (Figure 2): SPDP (Sanitization Prohibited and Deletion Prohibited), SADP (Sanitization Allowed and Deletion Prohibited), SDP (Sanitized and Deletion Prohibited), SPDA (Sanitization Prohibited and Deletion Allowed), SADA (Sanitization Allowed and Deletion Allowed), SDA (Sanitized and Deletion Allowed) and D (Deleted).

SADA is the initial status for all subdocuments. The first scheme (SDS1) supports four status SPDP, SADA, SDA and D (Figure 4), while the second scheme (SDS2) supports six status SPDP, SADP, SDP, SADA, SDA and D (Figure 6).

3.4 Proposed Scheme 1 (SDS1)

This subsection describes the first sanitizable and deletable signature (SDS1) based on MHI's deletable signature introduced in section 2.

As described in section 3.2, MHI's deletable signature can be applied to a document with ordered subdocuments, and, in addition, sanitizations can be realized. Thus we establish the first sanitizable and deletable signature SDS1 (Figure 3). In the beginning, each subdocument is padded by a randomly generated document ID and subdocument ID. Here, subdocument IDs are increasing sequence. Then, a signer publishes individual signatures σ_i and an aggregate signature. In SDS1, revisers can sanitize or delete subdocuments by using these data. A detailed description of SDS1 is in Algorithm 3.

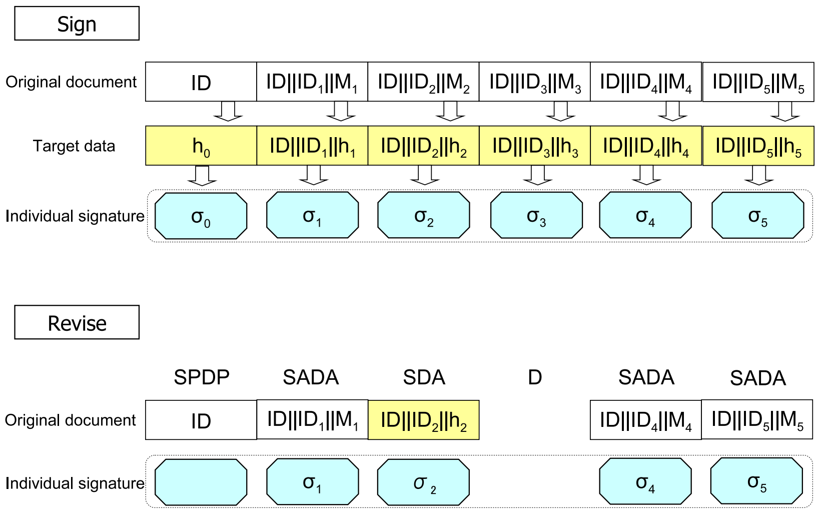
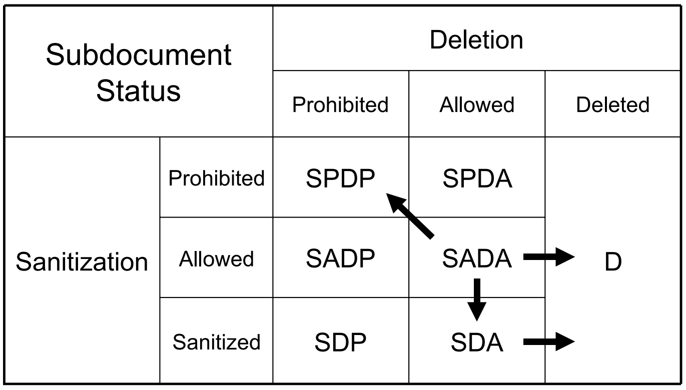


Fig. 3. Outline of SDS1



Subdocument Status		Deletion: Prohibited	Deletion: Allowed	Deletion: Deleted
Sanitization	Prohibited	SPDP	SPDA	D
	Allowed	SADP	SADA	
	Sanitized	SDP	SDA	

Fig. 4. Subdocument status of SDS1

SDS1 supports four subdocument status SPDP, SADA (initial status), SDA and D (Figure 3). To delete a subdocument $\bar{M}_i$, $\bar{M}_i$ and related data are deleted. Especially, the aggregate signature is updated by $\sigma \leftarrow \sigma/\sigma_i$. To sanitize $\bar{M}_i$, the subdocument is updated by $\bar{M}_i \leftarrow ID||ID_i||H_0(ID||ID_i||\bar{M}_i)$. Since the individual signature σ_i are not deleted, a reviser can delete this subdocument afterwards. A status transition of SDS1 is summarized in Figure 4. In SDS1, only four transitions are possible.

Let us consider the security of SDS1. Since the proposed sanitizable and deletable signature is a combination of the sanitizable signature and the deletable signature, the integrity of disclosed subdocuments, the secrecy of sanitized

Algorithm 3. A description of the 1st sanitizable and deletable signature (SDS1)

KeyGen

```
1. Generate $\mathsf{sk} \overset{R}{\leftarrow} \mathbb{Z}/p\mathbb{Z}$ randomly and set $\mathsf{pk} \leftarrow g_2^{\mathsf{sk}}$.
Output: A secret and public key pair $(\mathsf{sk}, \mathsf{pk}) \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{G}_2$
```

Sign

```
Input: A document $(M_1, \ldots, M_n)$ and a secret key $\mathsf{sk} \in \mathbb{Z}/p\mathbb{Z}$
1. Generate random values $ID$ (as a document ID) and $ID_1, \ldots, ID_n$ (as subdocument IDs in increasing) and set $\bar{M}_0 \leftarrow ID$, $\bar{M}_i \leftarrow ID||ID_i||M_i$ for each subdocument $\bar{M}_i$ $(i = 1, \ldots, n)$.
2. Generate an individual signature $\sigma_i \leftarrow H(ID||ID_i||H_0(\bar{M}_i))^{\mathsf{sk}}$ for each padded subdocument $\bar{M}_i$ $(i = 0, \ldots, n)$.
3. Generate an aggregate signature $\sigma \leftarrow \prod_{i=0}^{n} \sigma_i$.
4. Set $SADA \leftarrow \{1, \ldots, n\}$, $SPDP \leftarrow \{0\}$, $SDA \leftarrow \phi$ as index sets.
Output: A document $(\bar{M}_0, \ldots, \bar{M}_n)$, individual signatures $\sigma_1, \ldots, \sigma_n$, an aggregate signature $\sigma \in \mathbb{G}_1$ and index sets $SADA$, $SPDP$, $SDA$
```

Revise

```
Input: A document $(\bar{M}_0, \ldots, \bar{M}_k)$, individual signatures $\{\sigma_i\}_{i \in SADA \cup SDA}$, an aggregate signature $\sigma$ and index sets $SADA$, $SPDP$, $SDA$
1. Determine disjoint new index sets $SPDP' \supseteq SPDP$, $SADA' \subseteq SADA$, $SDA'$, $D' \subseteq SADA \cup SDA$ such that their union is $\{1, \ldots, k\}$.
2. Do the following procedure for each index $i$:
    – Deletion: if $i \in D'$, update $\sigma \leftarrow \sigma/\sigma_i$ and delete $\bar{M}_i$, $\sigma_i$.
    – Prohibition: if $i \in SPDP' \setminus SPDP$, delete $\sigma_i$.
    – Sanitized: if $i \in SDA' \setminus SDA$, update $\bar{M}_i \leftarrow ID||ID_i||H_0(\bar{M}_i)$.
    – Otherwise: do nothing.
3. Renumber the indexes.
Output: A document $(\bar{M}_0, \ldots, \bar{M}_{k'})$, individual signatures $\{\sigma_i\}_{i \in SADA' \cup SDA'}$, an aggregate signature $\sigma' \in \mathbb{G}_1$ and index sets $SADA'$, $SPDP'$, $SDA'$.
```

Verify

```
Input: A document $(\bar{M}_0, \ldots, \bar{M}_k)$, individual signatures $\{\sigma_i\}_{i \in SADA \cup SDA}$, an aggregate signature $\sigma$, index sets $SADA, SPDP, SDA$ and a signer's public key $\mathsf{pk}$
1. Check whether the document ID in each subdocument is same. If not, output invalid and terminate.
2. Check whether the subdocument ID in each subdocument is increasing. If not, output invalid and terminate.
3. Compute $h_i$ for each subdocument such that $h_i \leftarrow ID||ID_i||H_0(\bar{M}_i)$ for $i \in SPDP \cup SADA$ and $h_i \leftarrow \bar{M}_i$ for $i \in SDA$. 4. Check whether $e(\sigma_i, g_2) = e(h_i, \mathsf{pk})$ hold for subdocuments $\bar{M}_i$ $(i \in SADA \cup SDA)$. If not, output invalid and terminate.
5. Check whether $e(\sigma, g_2) = \prod_{i=0}^{k} e(h_i, \mathsf{pk})$ hold. If not, output invalid and terminate.
6. Output valid and terminate.
```

subdocuments and the secrecy of deleted subdocuments should be established. In SDS1, the integrity of disclosed subdocuments, namely, subdocuments $\bar{M}_i$ such that $i \in SADA \cup SPDP$, are checked by the aggregate signature σ. In addition, the justification of the sanitization is also checked with the aggregate signature. On the other hand, since the sanitized subdocument is updated by its hash value, it is infeasible to recover the subdocument. For deleted subdocuments, it entirely hard to recover the subdocument since all related information is deleted. Thus, SDS1 established the integrity of the disclosed subdocuments and the secrecy of closed (sanitized/deleted) subdocuments.

3.5 Proposed Scheme 2 (SDS2)

This subsection describes another sanitizable and deletable signature (SDS2) based on MHI's deletable signature and SDS1. Different from SDS1, SDS2 uses two aggregate signatures σ and τ (and their corresponding individual signatures). An outline of SDS2 is in Figure 5 and a detailed description of SDS2 is in Algorithm 4.

One of the feature of SDS1 is that, a sanitized subdocument can be deleted. By using two aggregate signatures, we newly support two additional subdocument status SADP and SPD. Thus, a subdocument can be the status 'SDP' (Sanitized and Deletion Prohibited). Consequently, SDS2 supports six status SPDP, SADP, SDP, SADA, SDA and D (Figure 6).

The security discussion of SDS2 can be checked similarly to that of SDS2. Thus, SDS2 established the integrity of the disclosed subdocuments and the secrecy of closed (sanitized/deleted) subdocuments.

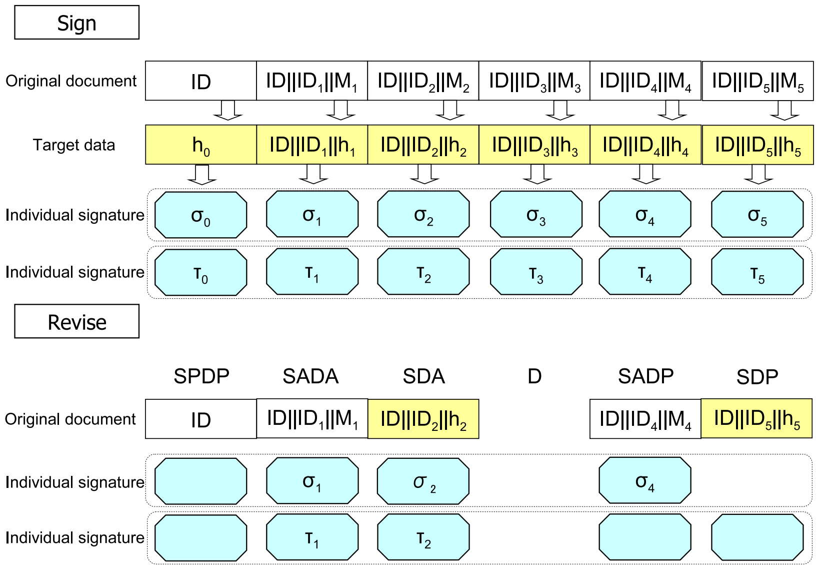


Fig. 5. Outline of SDS2

Algorithm 4. A description of the 2nd sanitizable and deletable signature (SDS2)

```
KeyGen
1. Generate $\mathsf{sk} \overset{R}{\leftarrow} \mathbb{Z}/p\mathbb{Z}$ randomly and set $\mathsf{pk} \leftarrow g_2^{\mathsf{sk}}$.
Output: A secret and public key pair $(\mathsf{sk}, \mathsf{pk}) \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{G}_2$
```

```
Sign
Input: A document $(M_1, \ldots, M_n)$ and a secret key $\mathsf{sk} \in \mathbb{Z}/p\mathbb{Z}$
1. Generate random values $ID$ and $ID_1, \ldots, ID_n$ (in increasing) and set $\bar{M}_0 \leftarrow ID$, $\bar{M}_i \leftarrow ID||ID_i||M_i$ for each subdocument $\bar{M}_i$ $(i = 1, \ldots, n)$.
2. Generate individual signatures $\sigma_i \leftarrow H(ID||ID_i||H_0(\bar{M}_i)||0)^{\mathsf{sk}}$ and $\tau_i \leftarrow H(ID||ID_i||H_0(\bar{M}_i)||1)^{\mathsf{sk}}$ for each padded subdocument $\bar{M}_i$ $(i = 0, \ldots, n)$.
3. Generate aggregate signatures $\sigma \leftarrow \prod_{i=0}^{n} \sigma_i$ and $\tau \leftarrow \prod_{i=0}^{n} \tau_i$.
4. Set $SADA \leftarrow \{1, \ldots, n\}$, $SPDP \leftarrow \{0\}$, $SDA \leftarrow \phi$, $SADP \leftarrow \phi$, $SDP \leftarrow \phi$ as index sets.
Output: A document $(\bar{M}_0, \ldots, \bar{M}_n)$, individual signatures $\sigma_1, \ldots, \sigma_n$, $\tau_1, \ldots, \tau_n$, aggregate signatures $\sigma$, $\tau \in \mathbb{G}_1$ and index sets $SADA$, $SPDP$, $SDA$, $SDAP$, $SPD$
```

```
Revise
Input: A document $(\bar{M}_0, \ldots, \bar{M}_k)$, individual signatures $\{\sigma_i\}_{i \in SADA \cup SDA \cup SADP}$, $\{\tau_i\}_{i \in SADA \cup SDA}$, aggregate signatures $\sigma$, $\tau$ and index sets $SADA$, $SPDP$, $SDA$, $SDAP$, $SPD$
1. Determine disjoint new index sets $SPDP' \supseteq SPDP$, $SADA' \subseteq SADA$, $SDA'$, $D' \subseteq SADA \cup SDA$, $SADP'$, $SDP' \supseteq SDP$. such that their union is $\{1, \ldots, k\}$.
2. Do the following procedure for each index $i$:
    – Deletion: if $i \in D'$, update $\sigma \leftarrow \sigma/\sigma_i$ and delete $\bar{M}_i$, $\sigma_i$.
    – Prohibition: if $i \in SPDP'$ and $i \in SADA$, delete $\sigma_i$ and $\tau_i$.
    – Prohibition: if $i \in SPDP'$ and $i \in SADP$, delete $\sigma_i$.
    – Prohibition: if $i \in SADP'$ and $i \in SADA$, delete $\tau_i$.
    – Sanitization: if $i \in SDA'$ and $i \in SADA$, update $\bar{M}_i \leftarrow ID||ID_i||H_0(\bar{M}_i)$.
    – Sanitization: if $i \in SDP'$ and $i \in SADP$, update $\bar{M}_i \leftarrow ID||ID_i||H_0(\bar{M}_i)$ and delete $\sigma_i$.
    – Sanitization: if $i \in SDP'$ and $i \in SADA$, update $\bar{M}_i \leftarrow ID||ID_i||H_0(\bar{M}_i)$ and $\sigma \leftarrow \sigma/\sigma_i$, and delete $\sigma_i$, $\tau_i$.
    – Sanitization: if $i \in SDP'$ and $i \in SDA$, update $\sigma \leftarrow \sigma/\sigma_i$, and delete $\sigma_i$, $\tau_i$.
    – Otherwise: do nothing.
3. Renumber the indexes.
Output: A document $(\bar{M}_0, \ldots, \bar{M}_{k'})$, individual signatures $\{\sigma_i\}_{i \in SADA' \cup SDA' \cup SADP'}$, $\{\tau_i\}_{i \in SADA' \cup SDA'}$, aggregate signatures $\sigma$, $\tau$ and index sets $SADA'$, $SPDP'$, $SDA'$, $SDAP'$, $SPD'$
```

Verify

Input: A document $(\bar{M}_0, \ldots, \bar{M}_k)$, individual signatures $\{\sigma_i\}_{i \in SADA \cup SDA \cup SADP}$, $\{\tau_i\}_{i \in SADA \cup SDA}$, aggregate signatures σ, τ, index sets $SADA$, $SPDP$, SDA, $SDAP$, SPD and a signer's public key pk

1. Check whether the document ID in each subdocument is same. If not, output $\mathsf{invalid}$ and terminate.
2. Check whether the subdocument ID in each subdocument is increasing. If not, output $\mathsf{invalid}$ and terminate.
3. Compute h_i, h'_i for each subdocument such that $h_i \leftarrow ID||ID_i||H_0(\bar{M}_i)||0$ for $i \in SPDP \cup SADA \cup SADP$, $h_i \leftarrow \bar{M}_i||0$ for $i \in SDA$, $h'_i \leftarrow ID||ID_i||H_0(\bar{M}_i)||1$ for $i \in SPDP \cup SADA \cup SADP$ and $h'_i \leftarrow \bar{M}_i||1$ for $i \in SDA \cup SDP$. 4. Check whether $e(\sigma_i, g_2) = e(h_i, \mathsf{pk})$ hold for subdocuments $\bar{M}_i$ $(i \in SADA \cup SDA \cup \cup SADP)$. Also check whether $e(\tau_i, g_2) = e(h'_i, \mathsf{pk})$ hold for subdocuments $\bar{M}_i$ $(i \in SADA \cup SDA)$. If not, output $\mathsf{invalid}$ and terminate.
5. Check whether $e(\sigma, g_2) = \prod_{i \in \{0,\ldots,k\} \setminus SDP} e(h_i, \mathsf{pk})$ hold. Also check whether $e(\tau, g_2) = \prod_{i \in \{0,\ldots,k\}} e(h'_i, \mathsf{pk})$ hold. If not, output $\mathsf{invalid}$ and terminate.
6. Output valid and terminate.

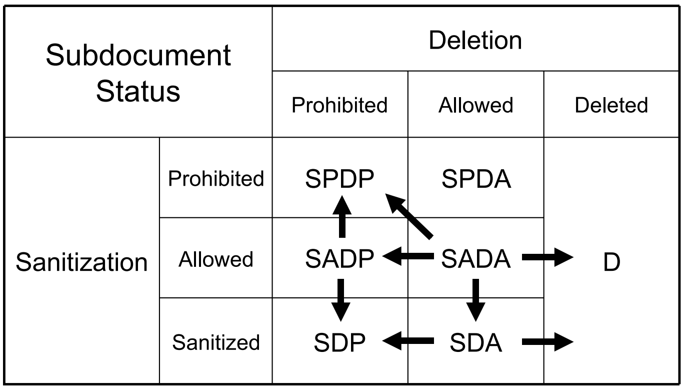


Fig. 6. Subdocument status of SDS2

4 Comparison

In this section, we compare the proposed sanitizable and deletable signatures SDS1 and SDS2 with the sanitizable signature by Miyazaki et al. [10][1], the sanitizable signature by Suzuki et al. from bilinear maps [12], and the deletable signature by Miyazaki, Hanaoka and Imai [9] from viewpoints of functions and efficiency (the number of signatures). A comparison is summarized in Table 1.

Previous privacy-preserving signatures were mono-function, while the proposed sanitizable and deletable signature supports multiple functions. Thus the proposed scheme can meet various requirements for hiding partial information.

[1] Functions and efficiency of other sanitizable signatures are almost same to this scheme.

Table 1. A comparison of privacy-preserving signatures

	Function					Efficiency	
	Sanitization	Deletion	SDA	SADP/SDP	SPDA	# of signatures sig.	# of agg. sig.
[10]	Yes	No	—	—	—	1	0
[12]	Yes	No	—	—	—	$n+1$	1
[9]	No	Yes	—	—	—	n	1
SDS1	**Yes**	**Yes**	**Yes**	No	No	n	1
SDS2	**Yes**	**Yes**	**Yes**	**Yes**	No	$2n$	2

However, as a tradeoff, the efficiency is not better than previous schemes. Especially, SDS2 requires much signatures than other schemes.

5 Concluding Remarks

This paper proposes a concept of the sanitizable and deletable signature as a combination of the sanitizable signature and the deletable signature. In addition, we proposed two concrete schemes based on MHI's deletable signatures. In spite of our constructions, a subdocument status SPDA is not supported. One of the reason may be the contradictional property of this status: Sanitization is Prohibited, but Deletion is Allowed. Constructing a scheme in which all subdocument status (including SPDA) will be a next task. Another task is to reduce the number of signatures in the proposed schemes.

References

1. Ateniese, G., Chou, D.H., Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.D., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Boneh, D., Gentry, G., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
4. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: PIATS: A partially sanitizable signature scheme. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 72–83. Springer, Heidelberg (2005)
5. Izu, T., Kunihiro, N., Ohta, K., Takenaka, M., Yoshioka, T.: Sanitizable Signature Schemes with Aggregation. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 51–64. Springer, Heidelberg (2007)
6. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic Signature Scheme. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)

7. Klonowski, M., Lauks, A.: Extended Sanitizable Signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
8. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential Aggregate Signatures from Trapdoor Permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
9. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally Signed Document Sanitizing Scheme Based on Bilinear Maps. In: 1st ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS 2006), pp. 343–354. ACM Press, New York (2006)
10. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control. The Institute of Electronics. Information and Communication Engineers (IEICE) Trans. on Fundamentals E88-A(1), 239–246 (2005)
11. Steinfeld, R., Bull, L., Zheng, Y.: Content Extraction Signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
12. Suzuki, M., Isshiki, T., Tanaka, K.: Sanitizable Signature with Secret Information: Technical report, 2006 Symposium on Cryptography and Information Security (SCIS 2006), p. 273 (2006)

An Efficient Scheme of Common Secure Indices for Conjunctive Keyword-Based Retrieval on Encrypted Data

Peishun Wang[1], Huaxiong Wang[1,2], and Josef Pieprzyk[1]

[1] Center for Advanced Computing – Algorithms and Cryptography, Department of Computing, Macquarie University, NSW 2109, Australia
{pwang,hwang,josef}@ics.mq.edu.au
[2] Division of Mathematical Sciences, Nanyang Technological University, Singapore

Abstract. We consider the following problem: members in a dynamic group retrieve their encrypted data from an untrusted server based on keywords and without any loss of data confidentiality and member's privacy. In this paper, we investigate common secure indices for conjunctive keyword-based retrieval over encrypted data, and construct an efficient scheme from Wang *et al.* dynamic accumulator, Nyberg combinatorial accumulator and Kiayias *et al.* public-key encryption system. The proposed scheme is trapdoorless and keyword-field free. The security is proved under the random oracle, decisional composite residuosity and extended strong RSA assumptions.

Keywords: Common secure index, conjunctive keyword search, dynamic accumulator.

1 Introduction

Instead of using their own servers, users often outsource their storage and backup needs to external data warehouses. This arrangement gives users significant financial saving but at the same time, introduces a risk to the privacy of their data. If the server cannot be trusted with the document contents, sensitive documents should be stored in encrypted form. However, a question naturally occurs in this scenario, that is, how the users retrieve documents based on their content. An ideal solution would be to let the server search the encrypted documents and return only relevant ones without leaking any information about the keywords or document contents. There are many research works that address this problem, some consider the single-user scenario [1,2,3,4,6,7,14], and some make searches for groups [5,8,13,15,17,18]. In this paper, we study searchable secure indices for a dynamic group system, and propose an efficient scheme of common secure indices for conjunctive keyword-based retrieval over encrypted data (CSI-CKR).

Related Works. Song, Wagner, and Perrig [14] first introduced the notion of searchable encryption for a single-user. They proposed a scheme in the symmetric key setting, which encrypts each word (or each pattern) of a document separately.

Goh [6] presented a notion of secure index, which uses an index of keywords that is created by a Bloom filter. Boneh *et al* [2] originally presented public key schemes for keyword search, which is a mechanism for tagging messages that are generated under the public key. The encrypted messages can be searched using the trapdoor for any particular keyword. Golle, Staddon and Waters [7] first constructed schemes for conjunctive keyword search over encrypted data. The search techniques for single-users were further studied in [1,3,4].

Hwang *et al.* [8] designed a public key based conjunctive keyword search scheme for a group of users from bilinear pairing. Wang *et al.* [17] recently designed a scheme of threshold privacy preserving keyword searches. Unfortunately, these two works are for static groups and the schemes do not support dynamic groups.

To the best of our knowledge, there are four works on secure indices for a dynamic group system. Park *et al.* [13] proposed search schemes for groups based on Goh's scheme [6]. In their schemes, the secret keys, such as authentication keys, encryption keys and index generation keys, are produced by computing one-way hash key chain, and a set of new secret keys is generated after a member leaves. If a leaving member reveals her group key to the server, the server administrator can obtain all data encrypted previously, since he can compute all previous secret keys by hashing the current group key. Additionally, the size of a query would become larger as the number of leaving members increases. Wang *et al*'s work [15] addressed the shortcomings of Park *et al*'s schemes, formally introduced the notion of common secure indices for conjunctive keyword-based retrieval over encrypted data (CSI-CKR) and proposed a new scheme. Curtmola *et al.* [5] presented a searchable symmetric encryption for dynamic groups, but their setting is different, and their scheme cannot support conjunctive keyword searches. Note that, the above conjunctive keyword search schemes define keyword fields in the index. This means, to make conjunctive keyword searches, the values of positions of conjunctive keywords in a secure index have to be given as compulsory information. Very recently Wang *et al.* [18] presented a new scheme of conjunctive keyword searches on encrypted data without keyword fields and extended it to the setting of dynamic groups. However, the size of a query in their scheme is linear in the number of keywords contained in a secure index.

Our Contributions. The contributions of this paper are three-fold:

(1) We construct a new CSI-CKR, and prove the security under the random oracle, decisional composite residuosity and extended strong RSA assumptions.

(2) We analyse efficiency of the proposed scheme by comparing with Wang *et al*'s scheme [15], and show that the proposed CSI-CKR is more efficient.

(3) Our scheme has three attractive properties. (a) It is trapdoorless, this means, we do not use any public or secret keys to generate a trapdoor for a list of keywords. (b) It is keyword field free, *i.e.*, our scheme eliminates the need for keyword fields in secure indices. (c) It provides the instantiation of algorithms for encryption and decryption.

Organization. Section 2 provides notation, models and cryptographic preliminaries. In Section 3 we construct a CSI-CKR. Section 4 analyses the efficiency. Finally Section 5 includes conclusions.

2 Preliminaries

Throughout this paper, we use the following notations. Let $a \xleftarrow{R} A$ denote that an element a is chosen uniformly at random from the set A. PPT denotes *probabilistic polynomial time*, $|A|$ stands for the cardinality of a set A, and $|k|$ denotes the number of bits needed to represent an integer k. For a positive number $k > 1$, $[k]$ denotes the set of integers $\{1, \ldots, \lfloor k \rfloor\}$. The symmetric set difference of two sets A and B is denoted by $A \triangle B = (A \setminus B) \cup (B \setminus A)$. We write $x||y$ to denote the concatenation of the bit-strings x, y. We assume that the upper bound of the number of keywords in a document is m.

2.1 Models

Wang *et al.* introduced a definition of CSI-CKR in [15]. We now briefly review the model, definition and security of CSI-CKR.

The Model. CSI-CKR has three parties: a trusted group manager (GM), members in the dynamic group and a server. First, GM setups the system and distributes an authentication code to every member. A member encrypts her data, generates the corresponding secure indices, and stores them on the server. When a member wants to retrieve the documents containing some keywords, she makes the searchable information for the keywords, and sends it along with her authentication code to the server. Then, for the legitimate member, the server tests all secure indices to find the matched data, and returns them to the member. Finally, the member interacts with GM to get the plaintext data.

The Definition. CSI-CKR consists of five components: *SystemSetup, AuthCodGen, DataGen, DataQuery, DataDcrypt.* In details, the *SystemSetup* instantiates the scheme, the *AuthCodGen* generates members' PIN numbers, their secure codes and a secure test code, the *DataGen* builds searchable encrypted data, the *DataQuery* retrieves the matched data, and the *DataDcrypt* decrypts the encrypted data. Due to the limitation of space, we omit the formal definition. For the details, refer to [15].

Security Requirement. CSI-CKR provides data privacy and member privacy. Specifically, data privacy ensures that the server is not able to extract any information about the data. That means, the encrypted data, common secure indices, queries and searches do not leak anything about their contents. Also, data privacy guarantees that any leaving member is not able to search and retrieve data after her revocation. Member privacy prevents any body (excluding the group manager) to impersonate a legitimate member to query the data. In addition, although a member interacts with the group manager, member privacy guarantees that the group manager knows nothing about the data the member retrieves.

To prove the security of the proposed construction, we need a security game called Indistinguishability of Ciphertext from Ciphertext (ICC) [7,1], which captures the notion of security known as semantic security against chosen keyword-attacks (IND-CKA) for secure indices [6].

Definition 1. ICC *is a security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. As stated in [15], the common secure index CSI_R of a document R only concerns the keyword list L in R, so we use CSI_L instead of CSI_R in the following game.*

Setup. *$\mathcal{A}$ adaptively selects a polynomial number of keyword lists, L, and requests the common secure indices, CSI_L, from $\mathcal{C}$.*

Queries. *$\mathcal{A}$ may query $\mathcal{C}$ on a keyword list L' and receive the image $Ix_{L'}$ of L'. With $Ix_{L'}$, $\mathcal{A}$ can invoke* **SrhInd(**$Ix_{L'}$**,** CSI_L**)** *on a common secure index CSI_L to determine if all keywords in the list L' are contained in L or not.*

Challenge. *After making a polynomial number of queries, $\mathcal{A}$ decides on a challenge by picking two keyword lists L_0 and L_1 such that $\mathcal{A}$ must not have asked for the image of any word in $L_0 \triangle L_1$, and sends them to $\mathcal{C}$. Then $\mathcal{C}$ chooses $b \xleftarrow{R} \{0,1\}$, assigns a unique identifier DI_b, and invokes* **IndGen(**R, PK_g**)** *to obtain the common secure index CSI_{L_b}, then returns CSI_{L_b} to $\mathcal{A}$. After the challenge is issued, $\mathcal{A}$ is allowed again to query $\mathcal{C}$ with the restriction that $\mathcal{A}$ may not ask for the image of any word in $L_0 \triangle L_1$.*

Response. *Eventually $\mathcal{A}$ outputs a bit $b_{\mathcal{A}}$, and is successful if $b_{\mathcal{A}} = b$. The advantage of $\mathcal{A}$ in winning this game is defined as $Adv_{\mathcal{A}} = |Pr[b = b_{\mathcal{A}}] - 1/2|$, and the adversary is said to have an ϵ-advantage if $Adv_{\mathcal{A}} > \epsilon$.*

2.2 Assumptions

We briefly review two complexity assumptions called Decisional Composite Residuosity (DCR) [12] and extended strong RSA (es-RSA) [16]

Definition 2 (DCR Assumption). *There is no PPT algorithm that has advantage at least ϵ in distinguishing between: (i) tuples of the form $(n, u^n \bmod n^2)$ where n is a composite RSA modulus and $u \xleftarrow{R} Z^*_{n^2}$, and (ii) tuples of the form (n, v) where $v \xleftarrow{R} Z^*_{n^2}$.*

Definition 3 (es-RSA Assumption). *There exists no PPT algorithm that, given a safe number n whose factors are secret and a number $x \in Z^*_{n^2}$, outputs a pair of integers (s, y) such that $x = y^s \bmod n^2$, $n^2 > s > 2$ and $y \in Z^*_{n^2}$.*

3 Construction

In the view of technique, CSI-CKR consists of three processes: membership authentication in a dynamic group, conjunctive keyword search, and data encryption and decryption. To construct a CSI-CKS scheme, we need to find a membership authentication protocol, a conjunctive keyword search scheme and a specific data cryptosystem for such a setting. We introduce a dynamic accumulator for membership authentication in Section 3.1, a conjunctive keyword search scheme in Section 3.2 and a data cryptosystem in Section 3.3, then integrate the three techniques to a new CSI-CKR in Section 3.4.

3.1 Wang *et al.* Dynamic Accumulator

A dynamic accumulator is an algorithm, which merges a large set of elements into a constant-size value such that for an element accumulated, there is a witness confirming that the element was included into the value, with a property that accumulated elements can be dynamically added and deleted into/from the original set. Wang *et al.* [16] presented a dynamic accumulator for batch updates at *ICICS* 2007. We will use their dynamic accumulator, which we call WWP-DA, to authenticate memberships in the proposed scheme.

First we briefly review their construction. The Wang *et al.* Dynamic Accumulator is defined as a tuple $\mathcal{DA}$=(**KeyGen, ACCVal, WitGen, Verify, AddEle, DelEle, UpdWit**) of polynomial time algorithms, where **KeyGen** instantiates the scheme, **ACCVal** computes an accumulated value, **WitGen** creates the witness for every element, **Verify** checks if a given element is accumulated in the value or is not, **AddEle** adds some new elements to the accumulated value, **DelEle** deletes some elements from the accumulated value, and **UpdWit** updates witnesses for the elements that have been accumulated in the old accumulated value and also are accumulated in the new one. The detail of construction is as follows.

KeyGen(k, M)**:** Given a security parameter k and the upper bound M on the number of accumulated elements, generate a suitable safe modulus n that is k-bit long and create an empty set V. Let $\mathcal{C} = Z^*_{n^2} \backslash \{1\}$ and $T' = \{3, \cdots, n^2\}$. Select adaptively a number $\sigma \in Z_{n^2}$ and compute $\beta = \sigma\lambda \bmod \phi(n^2)$ such that $\beta \in T'$, where λ is Carmichael's function $\lambda(n) = lcm(p-1, q-1)$, and $\phi(n^2)$ is Euler's Totient function $\phi(n^2) = n\phi(n)$. Choose $\gamma \overset{R}{\longleftarrow} Z_{\phi(n^2)}$ such that $\gamma \notin \{\beta, \sigma\}$. Set the public key $P_u = (n, \beta)$ and the private key $P_r = (\sigma, \lambda, \gamma)$, then output the parameter $\mathcal{P} = (P_u, P_r)$.

AccVal$(L, \mathcal{P})$**:** Given a set of m distinct elements $L = \{c_1, \ldots, c_m\}$ ($L \subset \mathcal{C}$, $1 < m \leq M$) and the parameter $\mathcal{P}$, choose $c_{m+1} \overset{R}{\longleftarrow} \mathcal{C}$, and compute

$$x_i = F(c_i^{\gamma\sigma^{-1}} \bmod n^2) \bmod n \ (i = 1, \ldots, m+1),$$

$$v = \sigma \sum_{i=1}^{m+1} x_i \bmod n,$$

$$y_i = c_i^{\gamma\beta^{-1}} \bmod n^2 \ (i = 1, \ldots, m+1), \text{ and}$$

$$a_c = \prod_{i=1}^{m+1} y_i \bmod n^2.$$

Then output the accumulated value v and the auxiliary information a_c and $A_l = \{y_1, \ldots, y_m\}$.

WitGen($a_c, A_l, \mathcal{P}$): Given the auxiliary information a_c and A_l, and the parameter $\mathcal{P}$, choose randomly a set of m numbers $T = \{t_1, \ldots, t_m\} \subset T' \setminus \{\beta, \gamma\}$ $(i = 1, \ldots, m)$, and compute

$$w_i = a_c y_i^{\frac{-t_i}{\gamma}} \bmod n^2 \ (i = 1, \ldots, m).$$

Then output the witness $W_i = (w_i, t_i)$ for c_i $(i = 1, \ldots, m)$.

Verify(c, W, v, P_u): Given an element c, its witness $W = (w, t)$, the accumulated value v and the public key P_u, test whether $\{c, w\} \subset \mathcal{C}$, $t \in T'$ and $F(w^\beta c^t \bmod n^2) \equiv v \pmod n$. If so, output Yes; otherwise, output No.

AddEle($L^\oplus, a_c, v, \mathcal{P}$): Given a set of elements $L^\oplus = \{c_1^\oplus, \ldots, c_k^\oplus\}$ $(L^\oplus \subset \mathcal{C} \setminus L, 1 \le k \le M - m)$ to be inserted, the auxiliary information a_c, the accumulated value v and the parameter $\mathcal{P}$, choose $c_{k+1}^\oplus \xleftarrow{R} \mathcal{C}$ and a set of k numbers $T^\oplus = \{t_1^\oplus, \ldots, t_k^\oplus\} \xleftarrow{R} T' \setminus \{T \cup \{\beta, \gamma\}\}$, and compute

$$\begin{aligned}
x_i^\oplus &= F((c_i^\oplus)^{\gamma\sigma^{-1}} \bmod n^2) \bmod n \ (i = 1, \ldots, k+1), \\
v' &= v + \sigma \sum_{i=1}^{k+1} x_i^\oplus \bmod n, \\
y_i^\oplus &= (c_i^\oplus)^{\gamma\beta^{-1}} \bmod n^2, \ (i = 1, \ldots, k+1), \\
a_u &= \prod_{i=1}^{k+1} y_i^\oplus \bmod n^2, \text{ and} \\
w_i^\oplus &= a_c a_u (y_i^\oplus)^{\frac{-t_i^\oplus}{\gamma}} \bmod n^2 \ (i = 1, \ldots, k).
\end{aligned}$$

Set $a_c = a_c a_u \bmod n^2$, $T = T \cup T^\oplus$ and $V = V \cup \{a_u\}$.
Then output the new accumulated value v' corresponding to the set $L \cup L^\oplus$, the witnesses $W_i^\oplus = (w_i^\oplus, t_i^\oplus)$ for the new added elements $c_i^\oplus$ $(i = 1, \ldots, k)$ and the auxiliary information a_u and a_c.

DelEle($L^\ominus, a_c, v, \mathcal{P}$): Given a set of elements $L^\ominus = \{c_1^\ominus, \ldots, c_k^\ominus\}$ $(L^\ominus \subset L, 1 \le k < m)$ to be deleted, the auxiliary information a_c, the accumulated value v and the parameter $\mathcal{P}$, choose $c_{k+1}^\ominus \xleftarrow{R} \mathcal{C}$, and compute

$$\begin{aligned}
x_i^\ominus &= F((c_i^\ominus)^{\gamma\sigma^{-1}} \bmod n^2) \bmod n \ (i = 1, \ldots, k+1), \\
v' &= v - \sigma \sum_{i=1}^{k} x_i^\ominus + \sigma x_{k+1}^\ominus \bmod n, \\
y_i^\ominus &= (c_i^\ominus)^{\gamma\beta^{-1}} \bmod n^2 \ (i = 1, \ldots, k+1), \text{ and} \\
a_u &= y_{k+1}^\ominus \prod_{j=1}^{k} (y_j^\ominus)^{-1} \bmod n^2.
\end{aligned}$$

Set $a_c = a_c a_u \bmod n^2$ and $V = V \cup \{a_u\}$.
Then output the new accumulated value v' corresponding to the set $L \setminus L^\ominus$ and the auxiliary information a_u and a_c.

UpdWit(W_i, a_u, P_u): Given the witness W_i, the auxiliary information a_u and the public key P_u, compute $w_i' = w_i a_u \bmod n^2$, then output the new witness $W_i' = (w_i', t_i)$ for the element c_i.

Batch Update. Each element in the set V created by the algorithm **KeyGen** and updated by the algorithms **AddEle** and **DelEle** is related to a time when the element was added to V, and all element are arranged chronologically. When an element wants to use the accumulator after he missed N times update of witness, he can contact the accumulator and tell her the time of his last update, then the accumulator checks the set V, collects all data items $\{v_{i_1}, \ldots, v_{i_N}\} \subset V$ that the element did not use, computes the update information $a_u = v_{i_1} \ldots v_{i_N} \bmod n^2$, and returns a_u to the element. On receiving a_u, the element computes $w_i' = w_i a_u \bmod n^2$ to obtain the new witness $W' = (w_i', t_i)$.

3.2 Conjunctive Keyword Searches

First we reintroduce the well known definition of conjunctive keyword searches (CKS) on encrypted data [3,7] as follows.

Definition 4. *A* **CKS** *consists of the following four algorithms:*

HashGen(s, m): *takes as input a parameter s and the upper bound m, and outputs a system parameter P.*
BuildIndex(L, P): *takes as input a keyword list L and the system parameter P, and outputs its index I_L.*
MakeImage(L', P): *takes as input a keyword list $L' = \{w_i\}_{i=1,\ldots,k} (k \geq 1)$ and the system parameter P, outputs the image $x_{L'}$ of the list L'.*
Test($x_{L'}$, I_L): *takes as input the image $x_{L'}$ and the index I_L, outputs* **Yes** *if the keyword list $L' \subseteq L$ or* **No** *otherwise.*

Intuitively, CKS aims to capture the notion that an adversary should learn no other information about a document from its index and the image of a keyword list that was explicitly given to him except the fact whether the document contains all keywords in the list or not.

Now we use the Nyberg combinatorial accumulator [11] to construct a scheme of CKS as follows.

HashGen(s, m): Given a parameter s and the upper bound m, compute

$$l = e(\ln 2) s m' \lg m',$$

where $m' = m + 1$, and then choose a one-way hash function

$$H : \{0,1\}^* \to \{0,1\}^l,$$

where $l = rd$ for some integers r, d.
Finally output the system parameter $P = \{H, r, d\}$.

BuildIndex(R, P): For a keyword list $L = \{w_1, \ldots, w_k\}$ ($m \geq k \geq 1$) in a data R, the member selects a distinct document identifier $DI = w_0$, and does the following.

Step 1. If $k < m$, pads the keyword list $\{w_1, \ldots, w_k\}$ into $\{w_1, \ldots, w_m\}$ with $w_j = (j \times w_0)||w_0$ ($j = k+1, \ldots, m$). For each w_i ($i = 0, \ldots, m$), computes: $\bar{w}_i = H(w_i)$. Since H produces a hash digest of length $l = rd$, $\bar{w}_i$ can be viewed as an r-digit number in base 2^d, *i.e.* $\bar{w}_i = (\bar{w}_i^1, \ldots, \bar{w}_i^r)$, where $|\bar{w}_i^j| = d, j = 1, \ldots, r$.

Step 2. Maps $\bar{w}_i$ to Ix_i ($i = 0, \ldots, m$) : $Ix_i = (Ix_i^1, \ldots, Ix_i^r) = f(\bar{w}_i)$ by applying the following assignment for $j = 1, \ldots, r$:

$$Ix_i^j = \begin{cases} 0 \text{ if } \bar{w}_i^j = 0 \\ 1 \text{ otherwise.} \end{cases}$$

Step 3. Finally, computes the bitwise product (denoted with $\odot$) of Ix_0, Ix_1, ..., Ix_m:

$$Iy = (Iy_1, \ldots, Iy_r) = Ix_0 \odot Ix_1 \ldots \odot Ix_m,$$

where $Iy_j = Ix_0^j \odot Ix_1^j \ldots \odot Ix_m^j$ ($j = 1, \ldots, r$);

Step 4. Outputs $CSI_R = Iy$ as the common secure index of R.

MakImg(L', P): Given a keyword list $L' = \{w_1, \ldots, w_{k'}\}$, the member outputs the image $Ix_{L'}$ of the keyword list L'. The algorithm proceeds as follows:

Step 1. For $i = 1, \ldots, k'$, computes: $\bar{w}_i = (\bar{w}_i^1, \ldots, \bar{w}_i^r) = H(w_i)$, where $|\bar{w}_i^j| = d, j = 1, \ldots, r$.

Step 2. Maps $\bar{w}_i \longmapsto Ix_i$ ($i = 1, \ldots, k'$): $Ix_i = (Ix_i^1, \ldots, Ix_i^r) = f(\bar{w}_i)$ by replacing each $\bar{w}_i^j$ with 0 if and only if $\bar{w}_i^j = 0$, or with 1 otherwise.

Step 3. Computes $Ix_{L'} = (Ix_1^1 \odot Ix_2^1 \ldots \odot Ix_{k'}^1, \ldots, Ix_1^r \odot Ix_2^r \ldots \odot Ix_{k'}^r)$ as the image of L'.

Test($Ix_{L'}, CSI_R$): For every CSI_R, the server checks whether $Iy_i = 0$ in the CSI_R if $Ix^i = 0$ in the $Ix_{L'}$ for all $i = 1, \ldots, r$. If so, output **Yes**; Otherwise, **No**.

Notice that the above CKS scheme is trapdoorless, as we do not generate any trapdoor for a list of keywords. In addition, it is keyword field free, since we eliminate the need for the number of keyword fields and the values of positions of conjunctive keywords in the keyword fields. However, keyword fields are the compulsory information in all previous CKS schemes.

Theorem 1. *The proposed CKS is semantically secure against chosen keyword attacks under ICC.*

Proof. Suppose the proposed scheme is not semantically secure against chosen keyword-attacks, *i.e.*, there exists an adversary $\mathcal{A}$ who has an ϵ-advantage to win the **ICC** game, then we build an algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ as a subroutine to determine with an $\frac{\epsilon}{(2^d-1)^r}$-advantage if H is a pseudo-random function or a random function. Let $\mathcal{A}'$ query an oracle $\mathcal{O}_H$ for the unknown function $H : \{0,1\}^* \rightarrow \{0,1\}^l$. When running the algorithms **IndGen(R, PK_g)** and **MakImg(L', PK_g)**, $\mathcal{A}'$ substitutes evaluations of H with queries to the oracle $\mathcal{O}_H$. $\mathcal{A}'$ uses $\mathcal{A}$ in **ICC** game as follows:

Setup. $\mathcal{A}'$ chooses a polynomial number of keywords from a keyword domain uniformly at random and sends them to $\mathcal{A}$. Then, $\mathcal{A}$ adaptively returns a polynomial number of keyword lists. For each keyword list L, $\mathcal{A}'$ assigns a unique identifier DI, and runs the algorithm **IndGen**(R, PK_g) to build the common secure index CSI_L. After all the indices are created, $\mathcal{A}'$ gives them with respective keyword lists to $\mathcal{A}$.

Queries. $\mathcal{A}$ may query $\mathcal{A}'$ for the image of a keyword list L'. $\mathcal{A}'$ runs **MakImg** (L', PK_g) to make the image $Ix_{L'}$ of L' and replies. On receiving $Ix_{L'}$, $\mathcal{A}$ can invoke **SrhInd**$(Ix_{L'}, CSI_L)$ on every common secure index CSI_L to determine if all keywords in the list L' are contained in L or not.

Challenge. After making a polynomial number of queries, $\mathcal{A}$ decides on a challenge by picking two keyword lists L_0 and L_1 such that $\mathcal{A}$ must not have asked for the image of any word in $L_0 \triangle L_1$, and sending them to $\mathcal{A}'$. Then $\mathcal{A}'$ chooses $b \xleftarrow{R} \{0, 1\}$, assigns a unique identifier DI_b to L_b, and invokes **IndGen**(R, PK_g) to obtain the common secure index CSI_{L_b} for L_b, then returns it to $\mathcal{A}$. After the challenge of determining b for $\mathcal{A}$ is issued, $\mathcal{A}$ is allowed again to query $\mathcal{A}'$ with the restriction that $\mathcal{A}$ may not ask for the image of any word in $L_0 \triangle L_1$.

Response. Finally $\mathcal{A}$ outputs a bit $b_{\mathcal{A}}$, and is successful if $b_{\mathcal{A}} = b$. If $b_{\mathcal{A}} = b$, then $\mathcal{A}'$ outputs 1, indicating that it guesses that H is a pseudo-random function. Otherwise, $\mathcal{A}'$ outputs 0.

When H is a pseudo-random function. Because $\mathcal{A}$ has an ϵ-advantage to win the **ICC** game, and $\mathcal{A}'$ simulates the challenger $\mathcal{C}$ perfectly in the **ICC** game. Considering the map: d bits is mapped to 1 bit and only $\overbrace{0 \cdots 0}^{d}$ is mapped to 0, so we have

$$|Pr[\mathcal{A}'_{H_k} = 1 | k \xleftarrow{R} \{0, 1\}^l] - \frac{1}{2}| > \frac{\epsilon}{(2^d - 1)^r}. \tag{1}$$

When H is a random function. The hash value of a keyword is treated as a random value, it follows that each image can be viewed as a random value. For any two different keywords, their images as two random values are independent to each other. Considering the restriction imposed on $\mathcal{A}$'s choice of queries in **ICC** game: $\mathcal{A}$ never and ever asks for the image of any word in $L_0 \triangle L_1$, therefore, it is infeasible for $\mathcal{A}$ to learn anything about the images of keywords in $L_0 \triangle L_1$ from common secure indices and images of any other keywords (not in $L_0 \triangle L_1$). Additionally, as d bits are mapped to 1 bit in making image, it decreases further the possibility of correlating the image of one keyword with ones of others. As a result, we only need to consider the challenge keyword lists. Depending on the number of keywords in $L_0 \triangle L_1$, we identify two cases:

case 1: $|L_0 \triangle L_1| = 0$.

Without loss of generality, assume that the keyword list L_0 and L_1 have the same set of m keywords. Let Ix' be the image of the keyword list L_0 (the same to L_1). The two identifiers DI_0 and DI_1 that $\mathcal{A}'$ assigns to L_0 and L_1, respectively, are distinct and unrepeated, and $\mathcal{A}$ learns nothing about

DI_0 and DI_1. So, from $\mathcal{A}$'s view, the images Ix_{DI_0} of DI_0 and Ix_{DI_1} of DI_1 as two random values are indistinguishable. As two bit products of a value Ix' with other two indistinguishable numbers Ix_{DI_0} and Ix_{DI_1}, the common secure indices CSI_{L_0} and CSI_{L_1} are indistinguishable to $\mathcal{A}$.

case 2: $|L_0 \triangle L_1| > 0$.

Because each keyword list with less than m keywords is padded with unrepeated numbers to the length m in the algorithm **IndGen**(R, PK_g), without loss of generality, we assume that the keyword list L_0 and L_1 have $m-1$ common keywords and one distinct keyword (i.e. $w_{L_0} \in L_0$, $w_{L_1} \in L_1$, and w_{L_0} is different from w_{L_1}, so, $L_0 \triangle L_1 = \{w_{L_0}, w_{L_1}\}$). As in **case 1**, $\mathcal{A}'$ assigns two unique identifiers DI_0 and DI_1 to L_0 and L_1, respectively, and $\mathcal{A}$ learns nothing about DI_0 and DI_1, thus, the images Ix_{DI_0} of DI_0 and Ix_{DI_1} of DI_1 as two random values are indistinguishable to $\mathcal{A}$. Since $\mathcal{A}$ is not allowed to query $\mathcal{A}'$ for the images of w_{L_0} and w_{L_1}, that is, the image $Ix_{w_{L_0}}$ of w_{L_0} and the image $Ix_{w_{L_0}}$ of w_{L_1} as two random values are unknown to $\mathcal{A}$, so the images $Ix_{w_{L_0}}$ and the image $Ix_{w_{L_0}}$ are indistinguishable to $\mathcal{A}$. Although $\mathcal{A}$ may know the image Ix' of the keyword list $L_0 \setminus \{w_{L_0}\}$ (identical to $L_1 \setminus \{w_{L_1}\}$), the common secure indices CSI_{L_0} as a bit product of Ix' with two unknown random values Ix_{DI_0} and $Ix_{w_{L_0}}$ and CSI_{L_1} as a bit product of Ix' with two unknown random values Ix_{DI_1} and $Ix_{w_{L_1}}$ are indistinguishable to $\mathcal{A}$.

Based on the above analysis, $\mathcal{A}$ at best guesses b correctly with probability 1/2. Thus we have

$$Pr[\mathcal{A}'_f = 1 | f \xleftarrow{R} \{H : \{0,1\}^* \rightarrow \{0,1\}^l\}] = 1/2. \tag{2}$$

From (1) and (2), we get

$$|Pr[\mathcal{A}'_{H_k} = 1 | k \xleftarrow{R} \{0,1\}^l]$$
$$- Pr[\mathcal{A}'_f = 1 | f \xleftarrow{R} \{H : \{0,1\}^* \rightarrow \{0,1\}^l\}]| > \frac{\epsilon}{(2^d-1)^r}.$$

Therefore, we have proven the desired conclusion.

3.3 Three-Party Cryptosystem

To fulfil the security requirement for the data encryption and decryption in CSI-CKR, we introduce a specific three-party cryptosystem (TPC), in which, there are three parties: the server, group members, and the group manager. The group manager generates a pair of public and private keys for the system. The members encrypt their data with the public key and store them in the server. When a member retrieves encrypted data, she has to decrypt the data by interacting with the group manager. The security requires that the interaction should preserve the confidentiality of private key held by the group manager and the data the member wants to decrypt. It means that a member cannot get any information about the private key the group manager holds and the group manager gets no information about the data the member has.

Recently Kiayias *et al.* proposed a public-key encryption scheme [9]. We use their cryptosystem to construct a TPC as follows. We also adopt their notations. For the details of the Kiayias *et al.* cryptosystem, refer to [9].

The TPC consists of the following five algorithms:

SetUp(τ): Given a security parameter $\tau \in Z^+$, output the parameters of the KTY cryptosystem (n, g_1, y, h, z), where n is an RSA modulus, $g_1 = g^{2n} \bmod n^2$ for some $g \overset{R}{\leftarrow} Z^*_{n^2}$, $z \overset{R}{\leftarrow} [\frac{n^2}{4}] \setminus \{1, 2\}$, $y = g_1^z$ and $h = n + 1$. The public key is $Puk = \{n, g_1, y, h\}$, and the private key is $Prk = \{z\}$.

Encrpt(m, Puk): Given a data d and the public key Puk, choose $r_a \overset{R}{\leftarrow} [\frac{\sqrt{n}}{2}]$, compute $u = g_1^{r_a}$ and $v = y^{r_a} h^d$, and output the encrypted data $\{u, v\}$.

MaskDa(u): Given the first part u of an encrypted data, choose $r_b \overset{R}{\leftarrow} [\frac{\sqrt{n}}{2}] \setminus \{1, 2\}$, compute $u' = u^{r_b}$, and then output the masked data u' and keep r_b as a one-time secret key.

PartDe(u', Prk, Puk): Given the masked data u' and the keys Prk, Puk, check if $u' \in Z^*_{n^2}$. If so, output the decrypted masked data $\bar{u} = (u')^{-z} \bmod n^2$; otherwise, terminate the protocol.

FullDe($\bar{u}, v, r_b, Puk$): Given the decrypted masked data $\bar{u}$, the second part of the encrypted data v, the one-time secret key r_b and the public key Puk, compute

$$d' = v^{r_b}\bar{u} - 1 \bmod n^2, \text{ and}$$
$$d = (d' \cdot r_b^{-1} \bmod n)/n,$$

and output the data d.

An execution of the TPC goes as follows. First, the group manager runs the algorithm **SetUp(τ)** to generate the public and private keys for the system. Group members execute the algorithm **Encrpt(m, Puk)** to encrypt their data with the public key, and store them in the server. After a member retrieves an encrypted data, the member chooses a one-time secret key and executes the algorithm **MaskDa(u)** to mask the encrypted data, then sends it to the group manager. On receiving the masked data, the group manager runs the algorithm **PartDe(u', Prk, Puk)** to decrypt the masked data with the private key, and returns it to the member. Finally, the member runs the algorithm **FullDe($\bar{u}, v, r_b, Puk$)** with the one-time secret key to obtain the original plaintext data.

We now turn to show the security of the TPC.

Theorem 2. *The proposed TPC is secure under the DCR and es-RSA assumptions.*

Proof. First, let's consider the member's privacy. In algorithm **MaskDa(u)**, the encrypted data u is masked by a random power r_b that the member chooses secretly and the manager does not know. If the es-RSA assumption holds in $Z^*_{n^2}$, it is computationally impossible for the manager to compute u from u'. So, the manager knows nothing about the data the member retrieved. In addition, r_b is a one-time secret key, therefore, even though the member asks the manager to decrypt the same data many times, the manager never knows that the data is identical.

Next, we consider the security of private key z. Although the member knows u' and $\bar{u} = (u')^{-z} \bmod n^2$ for some z, under the es-RSA assumption, it is computationally impossible for the member to get the private key z.

Finally, the Kiayias *et al.* cryptosystem is secure under the DCR assumption, and the proposed TPC inherits the security from the Kiayias *et al.*'s scheme immediately.

3.4 New CSI-CKR

Now we apply the above WWP-DA, CKS and TPC to construct a new CSI-CKR scheme as follows.

SystemSetup: It includes the algorithm **KeyGen** in the WWP-DA, the algorithm **HashGen** in the CKS and the algorithm **SetUp** in the TPC.

Given a security parameter and some parameters, GM runs the three algorithms **KeyGen, HashGen, SetUp** to output the system public key and secret key, which include all the public keys and private keys in the WWP-DA, CKS and TPC, respectively.

AuthCodGen: It includes the five algorithms **AccVal, WitGen, AddEle, DelEle** and **UpdWit** in the WWP-DA.

For the members of the original group, GM runs the algorithms **AccVal** and **WitGen** to generate the PIN number t_i and the secure code $\{w_i, c_i\}$ for every member, and sends the secure test code v to the server. GM runs the algorithms **AddEle, DelEle** and **UpdWit** to process the deletion or addition of group members.

DataGen: It includes the algorithm **BuildIndex** in the CKS and the algorithm **Encrpt** in the TPC.

A group member runs the algorithm **BuildIndex** to create the secure index and the algorithm **Encrpt** to encrypt the data, and then uploads them to the server.

DataQuery: It includes the algorithms **MakImg** and **Test** in the CKS and the algorithm **Verify** in the WWP-DA.

A group member runs the algorithm **MakImg** to generate an image for a keyword list, and sends it along with the member's PIN number and secure code to the server. On receiving the query, the server runs the algorithm **Verify** to check if the query is legitimate. For a legal query, the server runs the algorithm **Test** to search on all secure indices, and returns all matched data to the member.

DataDcrypt: It includes the algorithms **MaskDa, PartDe** and **FullDe** in the TPC cryptosystem.

The group member who receives a data from the server runs the algorithm **MaskDa** to mask the data and sends the masked data to GM. GM runs the algorithm **PartDe** to partly decrypt the data and returns it to the member. Finally, the member runs the algorithm **FullDe** to get the plaintext.

In above scheme, we apply the WWP-DA to manage memberships in a straightforward way, so the scheme holds the properties of unforgeability and confidentiality of group memberships. Plus Theorem 1 and 2, we know immediately that the proposed CSI-CKR scheme provides the security for data privacy and member privacy. So, we have the following theorem.

Theorem 3. *The proposed CSI-CKR scheme is secure.*

4 Efficiency

We now discuss the computation, communication and space complexity of our construction, and compare it to Wang *et al*'s scheme [15].

For the membership authentication, the RSA accumulator used in [15] is more efficient than the WWP-DA. However, the WWP-DA has a special property of batch updates, which makes it very efficient to activate a revoked membership. This property is becoming much important in the e-world.

To evaluate the efficiency of conjunctive keyword searches in details, we use the data given in [10] to evaluate the security of schemes under that the security has to be guaranteed until the year 2020. That means, RSA modulus n is at least 1881 bits. Furthermore, we assume that the upper bound of the number of keywords in a list for each document is 16, and the number of keywords to search is 4. We use the false positives rate 2^{-10} in [6]. Then the comparison is as following:

Table 1. Detailed Evaluation for Wang *et al*'s Scheme and Ours

Scheme	Wang *et al*'s	Ours
CCBI	16$\times$ 1881-bit-Hash + 18 scalar multiplications in G_1	17$\times$1208-bit-Hash + 17 bitwise multiplications of of 302-bit strings
SI	31977	302
CCMIT	4$\times$ 1881-bit-Hash + 5 exponentiations in Z_{n^2} + 8 multiplications in Z_{n^2}	4$\times$1208-bit-Hash + 4 bitwise multiplications of 302-bit strings
CCSI	1 multiplication and 1 devision, 1 exponentiation in Z_{n^2}, 1 subtraction 5 additions in Z_n	bits checked < 302 (the average < 76 bits)
SQ	3766 bits,	302 bits

* CCBI: computation complexity for building an index
* SI: storage for an index
* CCMIT: computation complexity for making an image or trapdoor
* CCSI: computation complexity for searching an index
* SQ: the size of a query

From the above table, we know that our scheme is more efficient than Wang *et al*'s in keyword search on the operations of building indices, storing indices, making images or trapdoors, searching indices and the size of a query. In particular, the proposed search scheme is trapdoorless and uses the binary check to test if a common secure index contains the keywords, which speeds up retrieval more efficiently.

In addition, most of conjunctive keyword search schemes require the sequence of keywords for building indices and the values of positions of the conjunctive keywords in the keyword fields for searching on indices, but our scheme eliminates this requirement, so it is more convenient for practical application.

In Wang *et al.*'s scheme, the technique of blind signature is used to mask the encrypted data, but they did not give any detailed construction of encryption and decryption, we have no way to compare the details between these two schemes. However, in our scheme, the instantiation of algorithms for encryption and decryption are provided.

5 Conclusions

With Wang *et al.* dynamic accumulator, Nyberg combinatorial accumulator and Kiayias *et al.* public-key encryption system, we constructed an efficient CSI-CKR scheme and proved its security in the random oracle model. The proposed CSI-CKR enjoys several special features, namely trapdoorless keyword search, keyword field free indices and detailed data cryptosystem.

Acknowledgments

The work was in part supported by Australian Research Council Discovery grants DP0663452, DP0558773 and DP0665035. Huaxiong Wang's research was in part supported by Singapore Ministry of Education grant T206B2204.

References

1. Ballard, L., Kamara, S., Monrose, F.: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005)
2. Boneh, D., 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)
3. 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)
4. Crescenzo, G.D., Saraswat, V.: Public Key Encryption with Searchable Keywords Based on Jacobi Symbols. In: Srinathan, K., Pandu Rangan, C., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 282–296. Springer, Heidelberg (2007)

5. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: ACM CCS 2006, pp. 79–88. ACM Press, New York (2007)
6. Goh, E.-J.: Secure indexes. In: Cryptology ePrint Archive, Report, 2003/216 (February 25, 2004), http://eprint.iacr.org/2003/216/
7. Golle, P., Staddon, J., Waters, B.: Secure Conjunctive Search over Encrypted Data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
8. Hwang, Y.H., Lee, P.J.: Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007)
9. Kiayias, A., Tsiounis, Y., Yung, M.: Group Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007)
10. Lenstra, A.K., Verheul, E.R.: Selecting Cryptographic Key Sizes. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 446–465. Springer, Heidelberg (2000)
11. Nyberg, K.: Fast accumulated hashing. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 83–87. Springer, Heidelberg (1996)
12. Paillier, P.: Public-Key Cryptosystems based on Composite Degree Residue Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
13. Park, H.A., Byun, J.W., Lee, D.H.: Secure Index Search for Groups. In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2005. LNCS, vol. 3592, pp. 128–140. Springer, Heidelberg (2005)
14. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 44–55 (May 2000)
15. Wang, P., Wang, H., Pieprzyk, J.: Common Secure Index for Conjunctive Keyword-Based Retrieval over Encrypted Data. In: Jonker, W., Petković, M. (eds.) SDM 2007. LNCS, vol. 4721, pp. 108–123. Springer, Heidelberg (2007)
16. Wang, P., Wang, H., Pieprzyk, J.: A New Dynamic Accumulator for Batch Updates. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 98–112. Springer, Heidelberg (2007)
17. Wang, P., Wang, H., Pieprzyk, J.: Threshold Privacy Preserving Keyword Searches. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 646–658. Springer, Heidelberg (2008)
18. Wang, P., Wang, H., Pieprzyk, J.: Keyword Field-free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups. In: Franklin, M.K., Hui, L.C.K., Wang, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 178–195. Springer, Heidelberg (2008)

Extension of Secret Handshake Protocols with Multiple Groups in Monotone Condition[*]

Yutaka Kawai[1], Shotaro Tanno[1], Takahiro Kondo[2], Kazuki Yoneyama[1,**], Noboru Kunihiro[3], and Kazuo Ohta[1]

[1] The University of Electro-Communications 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan
{kawai,tanno,yoneyama,ota}@ice.uec.ac.jp
[2] Mizuho Information Research Institute, Inc. Advanced Information Technologies Division 5-16-6, Hakusan, Bunkyo-ku, Tokyo 112-0001 Japan
takahiro.kondo@mizuho-ir.co.jp
[3] The University of Tokyo 5-1-5 Kashiwanoha, Kashiwa-shi, Chiba 277-8561, Japan
kunihiro@k.u-tokyo.ac.jp

Abstract. Secret Handshake protocol allows members of the same group to authenticate each other secretly, that is, two members who belong to the same group can learn counterpart is in the same group, while non-member of the group cannot determine whether the counterpart is a member of the group or not. Yamashita and Tanaka proposed Secret Handshake Scheme with Multiple Groups (SHSMG). They extended a single group setting to a multiple groups setting where two members output "*accept*" iff both member's affiliations of the multiple groups are identical. In this paper, we first show the flaw of their SHSMG, and we construct a new secure SHSMG. Second, we introduce a new concept of Secret Handshake scheme, "monotone condition Secret Handshake with Multiple Groups (mc-SHSMG)", in order to extend the condition of "*accept*". In our new setting of handshake protocol, members can authenticate each other in monotone condition (not only both member's affiliations are identical but also the affiliations are not identical). The communication costs and computational costs of our proposed mc-SHSMG are fewer than the trivial construction of mc-SHSMG.

Keywords: Secret Handshake with Multiple Groups, Privacy preserving authentication, Anonymity, Monotone Condition.

A Secret Handshake protocol was introduced by Balfanz et al. [2], which allows *two* members of the same group to authenticate each other *secretly* and share a key for the further communication. "Secretly" means, if the two members belong to the same group, each member learns that counterpart belongs to the same group. Otherwise he learns nothing about counterpart's affiliation. Also, an

[*] This research was done while the third and fifth author were fully in the University of Electro-Communications.

[**] Supported by JSPS Research Fellowships for Young Scientists.

eavesdropper or a man in the middle learns nothing from the protocol (including whether two members belong to the same group, or even they are members of any single group).

For example, Secret Handshake can be used in next scenario. Assume that CIA agent, Alice, wants to authenticate to Bob only if he is a CIA agent. Moreover, if Bob is not a CIA agent, Bob is not be able to determine whether Alice is a CIA agent or not.

In this subsection, we review prior researches of Secret Handshake Schemes.

- **Two party Secret Handshake scheme with single group** (SHS)
  Balfanz et al. [2] constructed two party Secret Handshake Scheme with one group (SHS) by using key agreement protocol. Their scheme is secure under the Bilinear Diffie Hellman (BDH) assumption. Then Castelluccia et al. [3] constructed more efficient SHS using CA-oblivious encryption (see **Fig. 5**). Their scheme is secure under the Computational Diffie Hellman (CDH) assumption, which is weaker and more standard assumption than (BDH). Also, Ateniese et al. [1] constructed SHS by using Identity based encryption (IBE). They considered the flexibility of Secret Handshake protocol which members can specify the group of the member with whom they would like to authenticate. They use asymmetric DDH hard groups to satisfy the security property of SHS.
- **Multi party Secret Handshake scheme with single group** (MPSHS)
  Tudik and Xu [5,6] extended the Secret Handshake protocol to a *multi party* setting (MPSHS). They combined a Group signature scheme, a Centralized group key distribution scheme and a Distributed group key agreement scheme in order to construct a MPSHS framework. In their model, any number of members can authenticate to others iff all handshake players belong to the same group. Jarecki et al. [4] also constructed a MPSHS by combining the scheme of [3] and a group key agreement protocol.
- **Two party Secret Handshake scheme with multiple groups** (SHSMG)
  Yamashita and Tanaka [7] extended the Secret Handshake protocol from a single group setting to a *multiple groups* setting (SHSMG) by using the scheme of [3]. In their model, two members can authenticate each other only iff each one's affiliations of multiple groups are *identical*. Unfortunately, as we will describe in Section 3, the SHSMG of [7] did not satisfy one of the security property of Secret Handshake scheme, Detector Resistance (defined in Section 2).

In this paper, we focus on the multiple group setting of Secret Handshake. First, we show that SHSMG of [7] is not secure in the meaning of Detector Resistance (see Section 2 for the definition of the DR) and propose a concrete SHSMG which satisfies Detector Resistance and all the other basic security properties of Secret Handshake protocol.

Moreover, we introduce a new concept of Secret Handshake scheme with multiple groups, in order to extend the condition of *accept*. In the multiple group setting of [7], the authentication condition might be quite restrictive because Secret Handshake protocol should be flexible in order to be applicable to various situations. For example, Secret Handshake protocol can be used in social networks such as online dating. Suppose Alice has a list of requirements that partner must match and she does not want to reveal. If we use previous SHSMG, we can find the partner only whose requirement is *equal* to Alice's requirements. This equality for an authentication condition is inconvenient. We want to relax the authentication condition to be general.

The contributions of this paper are summarized as follows.

- The flaw of the [7] and proposal of secure SHSMG (see Section 3).
  - We show that SHSMG of [7] does not satisfy Detector Resistance which is one of the security properties of SHSMG. We describe a detector attack for SHSMG against [7] specifically. Also, we repair this scheme and propose a new secure SHSMG based on [3].
- Introduction of a new SHSMG concept (see Section 4).
  - We introduce a new SHSMG concept, monotone condition Secret Handshake scheme with multiple groups (mc-SHSMG) which allows members to authenticate each other in a monotone condition. We notice that, in this paper, the *monotone condition* means the authentication condition which satisfies the "monotone property" (see Section 5).
    Moreover, we propose a concrete scheme of mc-SHSMG, and give a security proof of it. Our proposed mc-SHSMG is based on SHS of [3]. We indicate that trivial mc-SHSMG which is constructed from [3] is not efficient. Here, no efficiency means that computational and communication cost are large. Then, we propose an idea to decrease them by using notion of "monotone property" which we applied to our proposed scheme.

In this section, we define the model and security properties of SHSMG.

In Secret Handshake system, there exists two entities in the group G^i(G^i means group i) as follows.

member : Who belongs to groups and executes **Handshake** protocol. Member obtains certificate $cert_i$ from **GA**.

Group Authority(GA) Who creates the group. **GA** issues certificate to member by executing **AddMember** protocol, and registers it. Denote $\mathbf{GA}_i$ as **GA** of G^i .

SHSMG consists the following four algorithms.

Setup : is the trusted algorithm which takes input as the security parameter k, outputs the public parameters **params**.

CreateGroup : is the key generation algorithm executed by $\mathbf{GA}_i$ which takes input as **params**, outputs the group public key G_i and $\mathbf{GA}_i$'s secret key gsk_i.

AddMember : is the protocol executed between $\mathbf{GA}_i$ and member U which takes common input as **params**, G_i, member's identity ID_U and $\mathbf{GA}_i$'s private input as gsk_i. $\mathbf{GA}_i$ use gsk_i to issue the certificate $cert_U$ corresponding to ID_U. The member keeps the certificate secretly.

Handshake : is the authentication protocol executed between member U and V which take common inputs as ID_U, ID_V, **params** and U's private inputs $(G_{U_1}, cert_{U_1})$,...,$(G_{U_n}, cert_{U_n})$ and V's private inputs $(G_{V_1}, cert_{V_1})$,...,$(G_{V_m}, cert_{V_m})$. At the end of the protocol, if U and V belong to the *same* multiple groups, then output *accept*. Otherwise output *reject*. When U and V execute **Handshake** protocol, we write $U \overset{\textbf{Handshake}}{\longleftrightarrow} V$.

SHSMG must satisfy following security properties: Correctness, Impersonator Resistance and Detector Resistance.

Correctness : If honest members U and V who belong to the same (multiple) groups $(\{G^{U_1}, \ldots, G^{U_n}\} = \{G^{V_1}, \ldots, G^{V_m}\})$ execute **Handshake** protocol, then both member output *accept*.

Impersonator Resistance$(\mathcal{IR})$ **:** Intuitively, the impersonator resistance is violated if an honest member V, who is a member of (multiple) groups $G^1, \ldots, G^n$, and an adversary $\mathcal{A}$, who *does not* belong to at least one of $G^1, \ldots, G^n$, execute **Handshake** protocol, then both member output *accept*. We say SHSMG is $\mathcal{IR}$ if there is no such polynomialy bounded adversary $\mathcal{A}$ who can make V output *accept* with non-negligible probability.

Detector Resistance$(\mathcal{DR})$ **:** Intuitively, an adversary $\mathcal{A}$ violates the detector resistance if $\mathcal{A}$ can decide whether some honest member V is a member of $G^1, \ldots, G^n$, even $\mathcal{A}$ *does not* belong to at least one of $G^1, \ldots, G^n$. We say SHSMG is $\mathcal{DR}$ if there is no such polynomialy bounded adversary $\mathcal{A}$ who can decide whether V is a member of each group $G^1, \ldots, G^n$ with probability non-negligibly higher than $1/2$.

In this section, we will point out the weakness of the scheme of Yamashita and Tanaka [7], and show the idea of how to construct a scheme which satisfies Detector Resistance and all the other basic property of Secret Handshake protocol. We show the scheme of SHSMG of [7] in **Fig.1**:

In this subsection, we will show the Detector Attack Game Model for SHSMG. *Overview of Detector Attack Game*: The detector attack challenger (who interacts the detector attacker) executes following handshake protocols, game 1 and 2, with a detector attacker.

- game1: the challenger executes the handshake protocol as honest player.
- game2: the challenger executes the handshake protocol as simulator player.

The challenger decides which game to be executed previously by the value of $j \in \{0, 1\}$. So, If $j = 0$ the challenger executes the handshake protocol as honest player at first. If $j = 1$, the challenger executes the handshake protocol as simulator at first.

$\mathcal{A}_{\mathcal{DR}}$ distinguish whether a challenger is simulator or honest player. Finally, $\mathcal{A}_{\mathcal{DR}}$ outputs j' as j.

Behavior as honest
The behavior of challenger as honest player are as follows: The input of challenger is $ID_{ch}, (G_i, \omega_{ch_i}, t_{ch_i})$ $(i = \{1, .., n\})$ and system parameters **params**.

Setup: Input of the security parameter k, pick prime number p, q of size k, element $g \in \mathbb{Z}_p^*$ of order q. $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$, $H_2 : \langle g \rangle \rightarrow \langle g \rangle$ is a hash function. It outputs system parameter **params**$=(p, q, g, H_1, H_2)$.

CreateGroup: $\mathbf{GA}_i$ picks group secret key $x_i \overset{R}{\leftarrow} \mathbb{Z}_q$, and outputs group public key $G_i := g^{x_i}$. Each G_i is permuted in order of the dictionary.

AddMember:

1. member $\longrightarrow \mathbf{GA}_i$: G_i,ID
   $\mathbf{GA}_i$ picks $r \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $\omega := g^r$, $t := r + x_i H_1(\omega, ID) \bmod q$.
2. member $\longleftarrow \mathbf{GA}_i$: ω, t
   Member keeps (ω, t) as a certificate.

Handshake: With input of the $(ID_U, (G_{U_i}, \omega_{U_i}, t_{U_i}))$ for $i = 1, \ldots, n$, $(ID_V, (G_{V_i}, \omega_{V_i}, t_{V_i}))$ for $i = 1, \ldots, n'$, from U and V, **Handshake** protocol is executed as follows.

1. $V \longrightarrow U$: $ID_V, \omega_{V_1}, \omega_{V_2}, \ldots, \omega_{V_{n'}}$
   If $n \neq n'$ then U outputs *reject*.
   Otherwise U executes as follows.
   $PK_{V_i} := \omega_{V_i} G_{U_i}^{H_1(\omega_{V_i}, ID_V)}$ $(i = 1, \ldots, n)$
   $m_U \overset{R}{\leftarrow} \langle g \rangle$, $s_U \overset{R}{\leftarrow} \mathbb{Z}_q$
   U computes $C_U := (c_{U_1}, c_{U_2})$ s.t.
   $c_{U_1} := g^{s_U}$,
   $c_{U_2} := m_U H_2(PK_{V_1}^{s_U}) \cdots H_2(PK_{V_{n'}}^{s_U})$
2. $V \longleftarrow U$: $ID_U, \omega_{U_1}, \omega_{U_2}, \ldots, \omega_{U_n}, C_U$
   If $n \neq n'$ then V outputs *reject*.
   Otherwise V executes as follows.
   $PK_{U_i} := \omega_{U_i} G_{V_i}^{H_1(\omega_{U_i}, ID_U)}$ $(i = 1, \ldots, n)$
   $m_V \overset{R}{\leftarrow} \langle g \rangle$, $s_V \overset{R}{\leftarrow} \mathbb{Z}_q$
   V computes $C_V := (c_{V_1}, c_{V_2})$ s.t.
   $c_{V_1} := g^{s_V}$,
   $c_{V_2} := m_V H_2(PK_{U_1}^{s_V}) \cdots H_2(PK_{U_1}^{s_V})$
   $m'_U := c_{U_2} / (H_2(c_{U_1}^{t_{V_1}}) \cdots H_2(c_{U_n}^{t_{V_n}}))$
   $resp_V := H_2(m'_U)$
3. $V \longrightarrow U$: $c_{V_1}, c_{V_2}, resp_V$
   U checks whether $H_2(m_U) \overset{?}{=} resp_V$.
   If equality holds, U outputs *accept*. Then U computes $m'_V := c_{V_2} / (H_2(c_{V_1}^{t_{U_1}}) \cdots H_2(c_{V_1}^{t_{U_n}}))$ and set $resp_U := H_2(m'_V)$.
   Otherwise U output *reject*.
4. $V \longleftarrow U$: $resp_U$
   V checks whether $H_2(m_V) \overset{?}{=} resp_U$. If equality holds, V outputs *accept*. Otherwise V outputs *reject*.

Fig. 1. Scheme of Yamashita and Tanaka

1): Challenger sends ID_{ch} and ω_{ch_i} $(i = 1, ..., n)$, receives $ID_{\mathcal{A}}$, $\omega_{\mathcal{A}_i}$ $(i = 1, ..., n)$, and $(c_{\mathcal{A}_1}, c_{\mathcal{A}_2})$.

2): Challenger computes for $i = 1, \ldots, n$

$$PK_{\mathcal{A}_i} = \omega_{\mathcal{A}_i} G_i^{H_1(\omega_{\mathcal{A}_i}, ID_{\mathcal{A}})}, \quad c_{ch_1} = g^{s_{ch}}, \quad c_{ch_2} = m_{ch} \prod_{i=1}^{n} H_2(PK_{\mathcal{A}_i}^{s_{ch}}),$$

$$m'_{\mathcal{A}} = \frac{c_{\mathcal{A}_2}}{\prod_{i=1}^{n} H_2(c_{\mathcal{A}_1}^{t_{ch_i}})} \quad \text{and} \quad resp_{ch} = H_2(m'_{\mathcal{A}}).$$

Here, $m_{ch} \leftarrow \langle g \rangle$ and $s_{ch} \leftarrow \mathbb{Z}_q$ is chosen uniformly by challenger.

3): Challenger sends $c_{ch_1}, c_{ch_2}, resp_{ch}$, and receives $resp_{\mathcal{A}}$.

Behavior as simulator player
The challenger chooses all parameters at uniformly random.

1): Challenger send ID_{ch} and ω_{ch_i} $(i = 1, ..., n)$, receives $ID_{\mathcal{A}}, \omega_{\mathcal{A}_i}$ $(i = 1, ..., n)$, and $(c_{\mathcal{A}_1}, c_{\mathcal{A}_2})$.
2): Challenger chooses $c_{ch_1}, c_{ch_2}, resp_{ch} \overset{R}{\leftarrow} \langle g \rangle$ and sends them.
3): Challenger receives $resp_{\mathcal{A}}$.

We will construct the procedure of $\mathcal{A}_{\mathcal{DR}}$ as follows. The goal of detector attack is to distinguish whether a challenger is honest player or simulator, namely $\mathcal{A}_{DR}$ breaks Detector Resistance which is basic security requirement of SHSMG.

The input of $\mathcal{A}_{\mathcal{DR}}$ is G_i $(i = 1, ..., n)$ and **params**.

1): $\mathcal{A}_{DR}$ receives ID_{ch} and ω_{ch_i} $(i = 1, ..., n)$. $\mathcal{A}_{DR}$ chooses $m_{\mathcal{A}} \leftarrow \langle g \rangle$, $s_{\mathcal{A}} \leftarrow \mathbb{Z}_q$ and computes for $i = 1, \ldots, n$ as follows;

$$PK_{ch_i} = \omega_{ch_i} G_{ch_i}^{H_1(\omega_{ch_i}, ID_{ch})}, \qquad c_{\mathcal{A}_1} = g^{s_{\mathcal{A}}} \quad \text{and} \quad c_{\mathcal{A}_2} = m_{\mathcal{A}} \prod_{i=1}^{n} H_2(PK_{ch_i}^{s_{\mathcal{A}}}).$$

2): $\mathcal{A}_{DR}$ chooses $ID_{\mathcal{A}} \leftarrow_R \{0,1\}^*$ and $\omega_{\mathcal{A}_i} \leftarrow \langle g \rangle$ $(i = 1, ..., n)$. $\mathcal{A}_{DR}$ sends $ID_{\mathcal{A}}$, $\omega_{\mathcal{A}_i}$, and $(c_{\mathcal{A}_1}, c_{\mathcal{A}_2})$ to the challenger.
3): $\mathcal{A}_{DR}$ receives $resp_{ch}$, c_{ch_1} and c_{ch_2}. If $resp_{ch} = H_2(m_{\mathcal{A}})$, $Ans_j = 1$. Otherwise $Ans_j = 0$. Here, $(j = \{0, 1\})$ which is chosen by the challenger.
4): After two times of handshake protocol were over, $\mathcal{A}_{DR}$ behaves as follow.
- **Case1:** $Ans_0 = Ans_1$
  $\mathcal{A}_{DR}$ picks $j' \leftarrow_R \{0,1\}$ and outputs j'.
- **Case2:** $Ans_0 = 1$ and $Ans_1 = 0$
  $\mathcal{A}_{DR}$ outputs $j' = 0$.
- **Case3:** $Ans_0 = 0$ and $Ans_1 = 1$
  $\mathcal{A}_{DR}$ outputs $j' = 1$.

When challenger executes **Handshake** as the honest player, since $\mathcal{A}_{DR}$ has the public keys of the groups that honest player belongs to, $\mathcal{A}_{DR}$ can generate $(c_{\mathcal{A}_1}, c_{\mathcal{A}_2})$ and verify the $resp_{ch}$. So, when the challenger is the honest player, $\mathcal{A}_{DR}$ always can distinguish the honest player or the simulator. When challenger executes **Handshake** as the simulator, the probability that the simulator can generate valid $resp_{ch}(= H_2(m_{\mathcal{A}}))$ is $\frac{1}{2^q}$. So, only if above case, $\mathcal{A}_{DR}$ fails Detector Attack.

Then, the success probability of Detector Attack is $1 - \frac{1}{2^q}$. This probability is non-negligible probability.

The reason that this attack succeeded is due to construction of $resp$. In [7], $\mathcal{A}_{DR}$ can confirm whether $resp_{ch}$ is valid without secret information (member certificate and secret key). So, $\mathcal{A}_{DR}$ can distinguish whether challenger is simulator or honest player, using only $m_{\mathcal{A}}$ which is chosen by $\mathcal{A}_{DR}$.

The remediation method of this flaw is as follows. In our secure SHSMG, $resp_{ch}$ is constructed by using $m_{\mathcal{A}}$ and m_{ch} as $H(m_{\mathcal{A}}, m_{ch})$. In this way, in order to decide whether $resp_{ch}$ is valid or not, $\mathcal{A}$ must decrypt given ciphertext c_{ch_1}, c_{ch_2} to get m_{ch}. We show the modified **Handshake** scheme in **Fig. 2**. Owing to the construction of $resp$, we use $H_3 : \{0,1\}^{2k} \rightarrow \{0,1\}^k$ addition to H_1 and H_2.

Handshake: With input of the $(ID_U, (G_{U_i}, \omega_{U_i}, t_{U_i}))$ for $i = 1, \ldots, n$, $(ID_V, (G_{V_i}, \omega_{V_i}, t_{V_i}))$ for $i = 1, \ldots, n'$ from U and V, **Handshake** protocol is executed as follows.

1. $V \longrightarrow U : ID_V, \omega_{V_1}, \omega_{V_2}, \ldots, \omega_{V_{n'}}$
   If $n \neq n'$ then U outputs *reject*.
   Otherwise U executes as follows.
   $PK_{V_i} := \omega_{V_i} G_{U_i}^{H_1(\omega_{V_i}, ID_V)}$ $(i = 1, \ldots, n)$
   $PK_V := \Pi_{i=1}^{n'} PK_{V_i}$
   $m_U \xleftarrow{R} \langle g \rangle$, $s_U \xleftarrow{R} \mathbb{Z}_q$
   U computes $C_U := (c_{U_1}, c_{U_2})$ s.t.
   $c_{U_1} := g^{s_U}, c_{U_2} := m_U \oplus H_2(PK_V^{s_U})$
2. $V \longleftarrow U : ID_U, \omega_{U_1}, \omega_{U_2}, \ldots, \omega_{U_n}, C_U$
   If $n \neq n'$ then V outputs *reject*.
   Otherwise V executes as follows.
   $PK_{U_i} := \omega_{U_i} G_{V_i}^{H_1(\omega_{U_i}, ID_U)}$ $(i = 1, \ldots, n)$
   $PK_U := \Pi_{i=1}^{n} PK_{U_i}$
   $m_V \xleftarrow{R} \langle g \rangle$, $s_V \xleftarrow{R} \mathbb{Z}_q$
   V computes $C_V := (c_{V_1}, c_{V_2})$ s.t.
   $c_{V_1} := g^{s_V}, c_{V_2} := m_V \oplus H_2(PK_U^{s_V})$
   $m'_U := c_{U_2} \oplus H_2(c_{U_1}^{t_{V_1} + \cdots + t_{V_n}})$
   $resp_V := H_3(m'_U, m_V)$
3. $V \longrightarrow U : C_V, resp_V$
   $m'_V := c_{V_2} \oplus H_2(c_{V_1}^{t_{U_1} + \cdots + t_{U_n}})$
   U checks whether $H_3(m_U, m'_V) \stackrel{?}{=} resp_V$. If equality holds, U outputs *accept* and computes $resp_U := H_3(m'_V, m_U)$. Otherwise U outputs *reject* and set $resp_U := *$.
4. $V \longleftarrow U : resp_U$
   V checks whether $H_3(m_V, m'_U) \stackrel{?}{=} resp_U$. If equality holds, V outputs *accept*. Otherwise V outputs *reject*.

Fig. 2. Our Proposed **Handshake** of SHSMG

In previous Secret Handshake scheme with multiple groups (SHSMG) [7], a member outputs *accept* iff U and V's affiliations of multiple groups are identical. Such the restriction of authentication condition might be quite restrictive, because Secret Handshake should be flexible in order to be applicable to various situations which we mentioned above. In this paper, we propose a scheme which a member can authenticate in monotone condition. (without limiting the situation like all affiliation of U and V are identical). We call this scheme monotone condition Secret Handshake scheme with multiple groups (mc-SHSMG). In this section, we will show the definition of mc-SHSMG.

First we will define the condition of outputting *accept* when a member executes **Handshake** protocol, as follows. We call this condition, *Authentication Condition*. This condition is decided in some way before the **Handshake** protocol, and given to the member before the execution of the protocol.

– *Authentication Condition* (AC)
  Let N be the total number of the groups, and X be the set of all groups, that is, $X = \{G^1, \ldots, G^N\}$. Here, the power set of X is denoted as $P(X) = \{Y | Y \subseteq X\}$. At this time, AC is a subset of $P(X)$.

mc-SHSMG consists of four algorithms whose definitions of **Setup**, **Create-Group** and **AddMember** are the same as those in SHSMG (see Section 2). We will define the **Handshake** as follows.

Handshake : is the authentication protocol executed between member U and V which take common inputs as AC, ID_U, ID_V, **params** and U's private inputs $(G_{U_1}, cert_{U_1})$,...,$(G_{U_n}, cert_{U_n})$ and V's private inputs $(G_{V_1}, cert_{V_1})$,...,$(G_{V_m}, cert_{V_m})$. At the end of the protocol, if U and V belongs to the groups which satisfy the AC, then output *accept.*

Let us review the oracle and **Handshake** players utilized in the definition of the security property of mc-SHSMG.

CG : is the **CreateGroup** oracle. Given G^i, set group public key G_i and output G_i.
AddM : is the **AddMember** oracle. Given (G_i, ID), output certificate *cert* corresponding to ID.
honesthonest member V who executes **Handshake** protocol. Given (**param s**,$AC, ID_V, ID_{\mathcal{A}}, (G_1, cert_{V_1}), \ldots, (G_N, cert_{V_N})$), V follows the **Handshake** protocol as prescribed ($ID_{\mathcal{A}}$ is counterpart's ID).
SIMsimulator which simulates the **Handshake** protocol. Given (**params**, $AC, ID_V, ID_{\mathcal{A}}$), it simulates as it's the execution of ID_V.

mc-SHSMG must satisfy following security properties: Correctness, Impersonator Resistance and Detector Resistance.

Correctness : If honest member U and V who belong to groups which satisfies AC execute **Handshake** protocol, then both members output *accept.*
Impersonator Resistance($\mathcal{IR}$) **:** Intuitively, the Impersonator Resistance is violated if adversary $\mathcal{A}$, who is a member of groups which *do not* satisfies the AC, and an honest member V, who is a member of groups which satisfy the AC, execute **Handshake** protocol, then both members output *accept.* We say mc-SHSMG is $\mathcal{IR}$ if every polynomialy bounded adversary $\mathcal{A}$ has negligible probability of winning in the following game against any string ID_V.

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathcal{IR}}(k)$
  $\mathbf{params} \leftarrow \mathbf{Setup}(k)$
  $(G_i, gsk_i) \leftarrow \mathbf{CreateGroup}(\mathbf{params})\{i = 1, ..., N\}$
  $cert_i \leftarrow \mathbf{AddMember}(gsk_i, \mathbf{params}, ID_V, G_i)\{i = 1, ..., N\}$
  $(AC, ID_A, State) \leftarrow \mathcal{A}^{\mathbf{CG},\mathbf{AddM}}(\mathbf{params}, ID_V, G_1, ..., G_N)$
  $\mathbf{honest}(\mathbf{params}, AC, ID_V, ID_{\mathcal{A}}, (G_1, cert_{V_1}), ..., (G_N, cert_{V_N}))$
  $\overset{\mathbf{Handshake}}{\longleftrightarrow} \mathcal{A}^{\mathbf{CG},\mathbf{AddM}}(State)$
  If $V$ outputs accept, then return 1. Otherwise return 0.
```

We say $\mathcal{A}$ wins if V outputs *accept* in above game. However, $\mathcal{A}$ must not construct AC using only the groups that $\mathcal{A}$ gets certificates as $ID_{\mathcal{A}}$ using by **AddM**. Also, $\mathcal{A}$ cannot ask $ID_{\mathcal{A}}$ and the group be included in AC to **AddM** after $\mathcal{A}$ outputs AC. We denote the advantage of the adversary $\mathcal{A}$ in breaking the $\mathcal{IR}$ of mc-SHSMG by

$$\mathbf{Adv}_{\mathcal{A}}^{\mathcal{IR}}(k) = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathcal{IR}}(k) = 1].$$

Detector Resistance($\mathcal{DR}$) :

Intuitively, the detector resistance is violated if adversary $\mathcal{A}$, who is a member of group which *does not* satisfies AC, and an honest member V, who is a member of groups which satisfy the AC, execute **Handshake** protocol, then $\mathcal{A}$ can decide whether V is a member of group which satisfies AC or not. We say mc-SHSMG is $\mathcal{DR}$ if every polynomialy bounded adversary $\mathcal{A}$ does not have probability non-negligibly higher than 1/2 of winning in the following game against any string ID_V.

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathcal{DR}}(k)$
  $\mathbf{params} \leftarrow \mathbf{Setup}(k)$
  $(G_i, gsk_i) \leftarrow \mathbf{CreateGroup}(\mathbf{params})\{i = 1, ..., N\}$
  $cert_i \leftarrow \mathbf{AddMember}(gsk_i, \mathbf{params}, ID_V, G_i)\{i = 1, ..., N\}$
  $(AC, ID_{\mathcal{A}}, State) \leftarrow \mathcal{A}^{\mathbf{CG},\mathbf{AddM}}(\mathbf{params}, ID_V, G_1, ..., G_N)$
  $j \overset{R}{\leftarrow} \{0,1\}$
  $\mathbf{player}_j := \mathbf{honest}; \mathbf{player}_{1-j} := \mathbf{SIM}$
  $\mathbf{player}_0 \overset{\mathbf{Handshake}}{\longleftrightarrow} \mathcal{A}^{\mathbf{CG},\mathbf{AddM}}(State)$
  $\mathbf{player}_1 \overset{\mathbf{Handshake}}{\longleftrightarrow} \mathcal{A}^{\mathbf{CG},\mathbf{AddM}}(State)$
  $j' \leftarrow \mathcal{A}^{\mathbf{CG},\mathbf{AddM}}(State)$
  If $\mathcal{A}$ outputs $j'$ such that $j = j'$, then return 1. Otherwise return 0.
```

We say $\mathcal{A}$ wins if $\mathcal{A}$ outputs j=j' in above game. However, $\mathcal{A}$ must not construct AC using only the groups that $\mathcal{A}$ gets certificates as $ID_{\mathcal{A}}$ using by **AddM**. Also, $\mathcal{A}$ cannot ask $ID_{\mathcal{A}}$ and the group be included in AC to **AddM** after $\mathcal{A}$ outputs AC. We denote the advantage of the adversary $\mathcal{A}$ in breaking the $\mathcal{DR}$ of mc-SHSMG by

$$\mathbf{Adv}_{\mathcal{A}}^{\mathcal{DR}}(k) = |\ \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathcal{DR}}(k) = 1] \text{ - } 1/2|.$$

In this section we will show the idea for the extension and the concrete scheme of mc-SHSMG.

In this paper we utilize logical function and truth table in order to express the monotone condition. Also we use next notations in the rest of this paper.

- $U \in G^i$ denotes that "U belongs to G^i". Moreover $U \in \{G^i, G^{i'}\}$ denotes that "U belongs to both G^i and $G^{i'}$".
- $G^i = 1$ represents $U \in G^i$, and $G^i = 0$ represents $U \notin G^i$.
- Z represents AC. The $\{Z|Z = 1\}$ represents the conditions of outputting *accept* when **Handshake** protocol is executed.

For example, we will show all possible cases of the total number of groups element is 3 ($N = 3$) in **Table 1**. In this case, the number of $P(X)$ is $|P(X)|$=8.

We define the AC as $\{Z|Z = 1\}$ in the above table. For example, assume that member U and V executes **Handshake** protocol when AC is (7) and (8)(which means $AC = \{\{G^1, G^2\}, \{G^1, G^2, G^3\}\}$). In this case, the protocol outputs *accept* when U and V belong a set of group either $\{G^1, G^2\}$ or $\{G^1, G^2, G^3\}$.

Table 1. Truth table($N = 3$)

	group			AC
	G^1	G^2	G^3	Z
(1)	0	0	0	0/1
(2)	0	0	1	0/1
(3)	0	1	0	0/1
(4)	0	1	1	0/1
(5)	1	0	0	0/1
(6)	1	0	1	0/1
(7)	1	1	0	0/1
(8)	1	1	1	0/1

In this manner, to combine $\{Z|Z = 1\}$, it is possible to execute the **Handshake** protocol in monotone condition. Here, scheme by Yamashita and Tanaka [7] could be considered as $\#\{Z|Z = 1\} = 1$, because members output *accept* only both member's affiliation is identical, which means there is only one AC.

On the other hand, consider the case where $Z = 1$ is set to (1). In this case, it outputs *accept* when handshake player do not belong to any group. Thus, there is no need to execute **Handshake** protocol because it does not make sense.

In our setting, we consider the situation where AC satisfies the monotone property defined as follows.

– Monotone property:
  We say AC satisfies the monotone property when AC satisfies the next expression; $\forall x, y \in P(X); (y \in AC) \wedge (y \leq x) \Rightarrow x \in AC$.

The reason why we consider the situation where AC satisfies the monotone property is as follows.

Let $(P(X), \leq)$ be the ordered set, and x, y be the element of $P(X)$. We will show the Hasse diagram of $(P(X), \leq)$ in **Fig. 3**. Condition x, y such that $x, y \in AC$ and $y \leq x$ should hold the following property.

"member U who can make V output *accept* in the condition x, can also V make output *accept* in condition y ".

This happens because member U keeps not only the certificate of x but also that of y. Which is to say, if we set the AC like $y \in AC$ and $x \notin AC$, the member who belongs to z(s.t. $(y \leq z) \wedge (y \neq z)$) can output *accept* even the member does not satisfy the AC. Such a thing must not happen.

Therefore, we consider the situation where AC satisfies the monotone property in our setting.

In this subsection, we discuss the communication costs and computational costs of our proposed scheme.

As we mentioned in Section 5.2, we consider the situation where AC satisfies the monotone property. Because of this, the size of AC (communication costs) increase and also computation cost increase . Thus, we will show how to decrease the communication costs and computational costs, by applying the next technique.

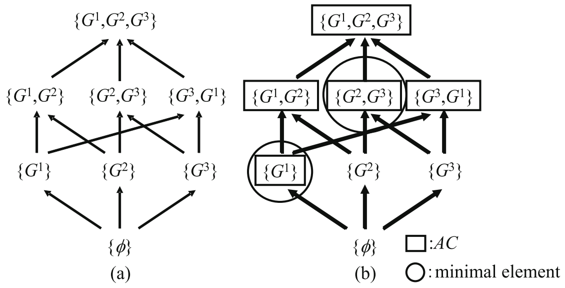


Fig. 3. (a)Hasse diagram in the case of $N = 3$, (b)AC and minimal element

Let's assume that AC to be AC=$\{\{G^1\}, \{G^1, G^2\}, \{G^2, G^3\}, \{G^3, G^1\}, \{G^1, G^2, G^3\}\}$. This example satisfies the monotone property. By a handshake player verifying all the 5 conditions, mc-SHSMG requirement can be achieved. We call this scheme as trivial scheme. Though, in the trivial scheme, a handshake player must verify all the conditions in AC, the communication costs and computation costs increase. But for the monotone property, all the conditions need not to be verified. It is sufficient to verify whether other member satisfies the minimal element of conditions or not, because member $U \in x$ can verify y correctly where $y \leq x$. In this example, we could see from **Fig. 3** that minimal element is $\{G^1\}$ and $\{G^2, G^3\}$. In verifying whether member satisfies $\{G^1\}, \{G^2, G^3\}$ or not, not only the member who belongs to $\{G^1\}, \{G^2, G^3\}$ but also the member who belongs to $\{G^1, G^2\}, \{G^3, G^1\}, \{G^1, G^2, G^3\}$ can prove that he satisfies the AC.

We use the same algorithm and protocol (**Setup**,**Create Group**, **AddMember**, **Recover**) as it is used in the scheme of Yamashita and Tanaka [7](there is a little different that we use $H_3 : \{0, 1\}^{3k} \rightarrow \{0, 1\}^k$ to construct a secure scheme which is mentioned in Section 3). The difference between our protocol and that of [7] is **Handshake** protocol and encryption scheme. The **Handshake** protocol and encryption scheme of proposed scheme is shown in **Figs**. 4 and 5. Member U does the following procedure before he executes **Handshake** protocol.

1. U obtains minimal elements of AC : $(AC_1, AC_2, \ldots, AC_l)$, using Hasse diagram. We denote a suffix set of groups which are contained in each minimal element AC_j as AC'_j and $AC' = (AC'_1, \ldots, AC'_l)$.
2. We denote the groups which are contained in $(AC_1, AC_2, \ldots, AC_l)$ as $G^{T_i} := G^{i'}$ for $i = 1, \ldots, n$, and the set of groups which U belongs to as $\mathcal{G}_U$. We notice that, $\{T_i | i = 1, \ldots, n\}$ is in the ascending order, that is, $T_1 < T_2 < \cdots < T_n$. Then U will set $(G_{U_i}, \omega_{U_i}, t_{U_i})$ for $i = 1, \ldots, n$ as follows:

$$(G_{U_i}, \omega_{U_i}, t_{U_i}) = \begin{cases} (G_{U_{i''}}, \omega_{U_{i''}}, t_{U_{i''}}) & (G^{T_i} \in \mathcal{G}_U) \\ s.t. G_{U_{i''}} = G_{T_i} & \\ (G_{T_i}, \omega_{T_i}, *) & (G^{T_i} \notin \mathcal{G}_U) \\ s.t. \omega_{T_i} = g^r (r \xleftarrow{R} \mathbb{Z}_q). & \end{cases}$$

Handshake:

– Let the number of minimal element of AC be l.

With input of the
$(AC, ID_U, (G_{U_i}, \omega_{U_i}, t_{U_i}))$ for $i = 1, \ldots, n$,
$(AC, ID_V, (G_{V_i}, \omega_{V_i}, t_{V_i}))$ for $i = 1, \ldots, n'$,
from U and V, **Handshake** protocol is executed as follows.

1. $V \longrightarrow U : ID_V, \omega_{V_1}, \omega_{V_2}, \ldots, \omega_{V_{n'}}$
   If $n \neq n'$ then U outputs *reject*.
   Otherwise U executes as follows.
   $PK_{V_i} :=$ Recover$(G_{U_i}, ID_V, \omega_{V_i})$ $(i = 1, \ldots, n')$
   $PK_{V_j} := \Pi_{i \in AC'_j} PK_{V_i}$ $(j = 1, \ldots, l)$
   $m_U \xleftarrow{R} \{0,1\}^k$
   $C_{U_j} :=$ Enc$_{PK_{V_j}}(m_U)$ $(j = 1, \ldots, l)$
2. $V \longleftarrow U : ID_U, \omega_{U_1}, \omega_{U_2}, \ldots, \omega_{U_n}, C_{U_1}, \ldots, C_{U_l}$
   If $n \neq n'$ then V outputs *reject*.
   Otherwise V executes as follows.
   $PK_{U_i} :=$ Recover$(G_{V_i}, ID_U, \omega_{U_i})$ $(i = 1, \ldots, n)$
   $PK_{U_j} := \Pi_{i \in AC'_j} PK_{U_i}$ $(j = 1, \ldots, l)$
   V picks ciphertext C_U corresponding to t_V, and executes as follows.
   $m'_U :=$ Dec$_{t_V}(C_U)$
   $m_V \xleftarrow{R} \{0,1\}^k$
   $C_{V_j} :=$ Enc$_{PK_{U_j}}(m_V)$ $(j = 1, \ldots, l)$
   $resp_V := H_3(m'_U, m_V)$
3. $V \longrightarrow U : C_{V_1}, \ldots, C_{V_l}, resp_V$
   U picks ciphertext C_V corresponding to t_U, and executes as follows.
   $m'_V :=$ Dec$_{t_U}(C_V)$
   U checks whether $H_3(m_U, m'_V) \overset{?}{=} resp_V$. If equality holds, U outputs *accept* and computes $resp_U := H_3(m'_V, m_U)$. Otherwise U outputs *reject* and set $resp_U := *$.
4. $V \longleftarrow U : resp_U$
   V checks whether $H_3(m_V, m'_U) \overset{?}{=} resp_U$. If equality holds, V outputs *accept*. Otherwise V outputs *reject*.

Fig. 4. Proposed **Handshake** of mc-SHSMG

3. Calculate t_U for arbitrary AC_j such that $AC_j \subseteq \mathcal{G}_U$.

$$t_U := \textstyle\sum_{i \in AC'_j} t_{U_i}$$

The member V does the same as U for steps 1-3.

In this section we will provide security proof of the proposed mc-SHSMG with respect to security requirement.

Let p, q be prime numbers. $\langle g \rangle$ denotes sub-group of $\mathbb{Z}_p^*$ generated by an element g where its order is q.

Definition 1 : Computational Diffie Hellman(CDH) problem
We say that $\mathcal{I}$ (τ, ϵ)-solves the CDH problem if, given (g, g^x, g^y), it is possible to compute g^{xy} whose running time is bounded by τ and its success probability is more than ϵ.

We say that CDH problem is (τ, ϵ)-hard if there is no algorithm $\mathcal{I}$ which can (τ, ϵ)-solve the CDH problem. We assume that there is no such algorithm $\mathcal{I}$ who can solve the CDH Problem with non-negligible probability. We call this CDH assumption.

Note that $G := g^x, \omega := g^r, t := r + xH_1(\omega, ID)$.
Recover(G, ID, ω) **:**
With input of the (G, ID, ω), computes $PK := \omega G^{H_1(\omega,ID)}$, and outputs encryption key PK.
Enc$_{PK}(m)$ **:**
PK is a encryption key. With input of the plaintext $m \in \{0,1\}^k$, picks $s \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $c_1 := g^s$, $c_2 := m \oplus H_2(PK^s)$. Output ciphertext C such that $C := (c_1, c_2)$.
Dec$_t(C)$ **:**
t is a decryption key. With input of the ciphertext C, computes $m' := c_2 \oplus H_2(c_1^t)$, and outputs plaintext m'.

Fig. 5. CA-Oblivious Encryption Scheme

Theorem 1 : The proposed scheme is complete.
Proof of Theorem 1 : Since it is easy to see that the scheme is complete, we omit the proof.

Theorem 2 : If CDH problem is hard, then the proposed scheme satisfies Impersonator Resistance.
Proof of Theorem 2 : Let $\mathcal{A}$ be an adversary who breaks Impersonator Resistance of the proposed scheme and can impersonate a member with advantage ϵ_A. Then we construct inverter $\mathcal{I}$ who uses $\mathcal{A}$ to solve an instance of CDH Problem with advantage ϵ_I. $\mathcal{I}$ is given (g, g^a, g^b) and try to compute g^{ab}.

We only give the brief sketch of the proof of Theorem 2. This proof is similar to the proof in [3].

In the simulation of Handshake protocol, $V(\mathcal{A})$ sends $resp_V \leftarrow \{0,1\}^k$ to $\mathcal{A}$, not $resp_V := H_3(m'_{\mathcal{A}}, m_V)$. This is because V does not know the value of m_V. But this is not a problem. The only way $\mathcal{A}$ can distinguish the real environment (conversation with an **honest**) and simulated environment is to check whether $resp_V = H_3(m_{\mathcal{A}}, m'_V)$ holds or not, where $m'_V = c_{V_{j2}} \oplus H_2(c_{V_{j1}}^{t_{\mathcal{A}}})$. And if $\mathcal{A}$ is possible to distinguish between real and simulated environment, then $\mathcal{I}$ can solve the CDH problem. This is the first case of the behavior of $\mathcal{A}$. The second case of the behavior of $\mathcal{A}$ is to make V output *accept*. In this case, CAI can also solve the CDH problem as same way as first case. Thus $\mathcal{I}$ breaks the CDH challenge.

Theorem 3 : If CDH problem is hard, then the propose scheme satisfies Detector Resistance.

This proof is similar to the proof in [3].

In this paper, we pointed out some weakness of the scheme of Yamashita and Tanaka [7] and proposed the remediation strategy of how to construct a scheme which satisfies all the basic security properties of the Secret Handshake protocol. Also, we proposed mc-SHSMG which two members can authenticate in monotone condition. Moreover, though communication costs and computational costs are large in trivial scheme, by using Hasse diagram, the proposed scheme makes member execute **Handshake** protocol in a less amount. It is obvious that, due to the flexibility, communication costs and computational costs are larger than the previous schemes.

In our scheme, if the handshake players both belong to groups which satisfy the AC, the situation that a handshake player can decide which condition that the counterpart satisfies, may happen. But, we unconcern this situation, because this may happen iff handshake players both belong to groups which satisfies the AC. What is important in the Secret Handshake protocol is, that a member who does not belong to groups which satisfy the AC can not do anything (like impersonate, or detect that counterpart belongs to groups which satisfies the AC or not). We leave to straighten out this situation as an open problem.

We have succeeded in relaxing the authentication to be general at the first step. Next we want to construct a SHSMG where member can authenticate in *arbitrary* condition. We might make this scheme by changing **AddMember** protocol, but it seems not to be efficient. We leave the construction of efficient arbitrary condition SHSMG as an open problem too.

References

1. Ateniese, G., Blanton, M., Kirsch, J.: Secret Handshake with Dynamic and Fuzzy Matching. In: Network and Distributed System Security Symposium (2007)
2. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.-C.: Secret Handshakes from Pairing-Based Key Agreements. In: IEEE Symposium on Security and Privacy, pp. 180–196 (2003)
3. 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)
4. Jarecki, S., Kim, J., Tsudik, G.: Authentication for paranoids: Multi-party secret handshakes. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 325–339. Springer, Heidelberg (2006)
5. Tsudik, G., Xu, S.: A Flexible Framework for Secret Handshakes (Multi-party Anonymous and Un-observable Authentication). Cryptology ePrint Archive (2005)
6. Tsudik, G., Xu, S.: Brief announcement: A flexible framework for secret handshakes. In: ACM Symposium on Principles of Distributed Computing (2005)
7. Yamashita, N., Tanaka, K.: Secret handshake with multiple groups. In: Lee, J.K., Yi, O., Yung, M. (eds.) WISA 2006. LNCS, vol. 4298, pp. 339–348. Springer, Heidelberg (2007)

Pseudorandom-Function Property of the Step-Reduced Compression Functions of SHA-256 and SHA-512

Hidenori Kuwakado[1] and Shoichi Hirose[2]

[1] Graduate School of Engineering, Kobe University
kuwakado@kobe-u.ac.jp
[2] Graduate School of Engineering, University of Fukui
hrs_shch@u-fukui.ac.jp

Abstract. Applications of an iterated hash function such as HMAC require that the compression function of the hash function is a pseudorandom function. However, the pseudorandom-function property of the compression function was not analyzed up to now. This paper shows that it is easy to distinguish between the 22 step-reduced SHA-512 compression function with the key-via-IV strategy and a random function. This is the first result on the pseudorandom-function property of the SHA-512 compression function with the key-via-IV strategy. A similar distinguishing attack is applicable to the SHA-256 compression function with the key-via-IV strategy.

1 Introduction

As SHA-256 and SHA-512 (so-called SHA-2) [14] are widely used in applications, the analysis of SHA-2 has been developed remarkably by the cryptanalysis community. For example, Mendel *et al.* [13] showed an 18-step collision of SHA-256 in 2006. Sanadhya and Sarkar [17] presented differential paths for 19 – 23 steps of SHA-256 in 2008. Indesteege *et al.* [11] showed collision attacks on SHA-256 reduced to 23 and 24 steps with complexities 2^{18} and 2^{50} in 2008. In addition, they also pointed out the non-random behavior of SHA-256 in the form of pseudo-near collision for up to 31 steps [11]. Recently, Sanadhya and Sarkar [16] have shown collision attacks against the 22-step SHA-512. Thus, previous results are related to the security of collision-resistance or that of non-random behavior based on the collision. We note that the collision-resistant property is important, but it is not all.

Applications such as HMAC require that a (keyed) hash function behaves as if it is a random function when the key is unknown. This property is called the *pseudorandom-function property*, which is closely related to the security of such applications. We notice that the security of HMAC based on the MDx family, SHA-0, or SHA-1 has been extensively studied [5,7,19], but the attacks are based on collision-finding attacks. Bellare [1] has shown that HMAC is a pseudorandom function if the compression function is a pseudorandom function with two keying strategies. That is, the proof of HMAC does not require that the hash function is collision-resistant. The pseudorandom-function property and the collision-resistant property are independent in the sense that there is no general reduction between them [18]. Indeed, the attack model on the collision-resistant property differs from the attack model on the pseudorandom-function property. In the attack model on the collision-resistant property, an adversary can search collisions only by himself, without other's help. Besides, in the attack model on the pseudorandom-function property, an adversary cannot obtain a hashed value without making queries to an oracle that knows a secret key. Accordingly, it is important to study not only the collision-resistant property but also the pseudorandom-function property.

The pseudorandom-function property of a hash function has been discussed in the context of domain extension [2,3,9]. Specifically, under the assumption that an underlying compression function is a pseudorandom function, methods for constructing a hash function that behaves as if it is the pseudorandom function have been studied. Hence, since the pseudorandom-function property of such a hash function is reduced to that of the underlying compression function, it is worth studying the pseudorandom-function property of the compression function.

When the compression function is the PGV mode [4,15], the security of the compression function is related to that of the underlying block cipher. From the viewpoint of block-cipher analysis, underlying block ciphers of SHA-1 and SHA-256, which are called SHACAL-1 and SHACAL-2 [8], have been analyzed [6,10,12]. In this case, the compression function with the key-via-IV strategy and the block cipher are different in the position of key input.

In this paper, we show that the 22 step-reduced SHA-2 compression function with the key-via-IV strategy is distinguishable from a random function. This is the first result on the pseudorandom-function property of the SHA-2 compression function. Our distinguishing attacks are practical in the sense that an adversary succeeds in distinguishing them with non-negligible probability in spite of the small number of queries and low complexity. Unlike collision-finding attacks, the adversary who performs distinguishing attacks can know only the hashed value, cannot know intermediate values. Hence, the adversary cannot use the message-modification technique that is effective in collision-finding attacks. Even though this paper represents a step forward in terms of security analysis on the pseudorandom-function property, the results do not threaten the security of applications using SHA-2.

This paper is organized as follows. Section 2 describes the algorithm of the SHA-2 compression function and the key-via-IV strategy to transform a non-keyed compression function into a keyed compression function, and defines the prf-advantage to measure the indistinguishability. Section 3 shows an algorithm for distinguishing between a 22 step-reduced SHA-512 compression function and a random function. We demonstrate that the prf-advantage of this algorithm is large. Furthermore, we show the differential path used by this algorithm and evaluate the probability that the algorithm succeeds in distinguishing them.

Section 4 describes a distinguishing algorithm for the 22 step-reduced SHA-256 compression function. Since the structure of the SHA-256 compression function is similar to that of the SHA-512 compression function, the distinguishing algorithm is applicable without a major modification. Section 5 concludes this paper.

2 Preliminaries

2.1 Pseudorandom-Function Property

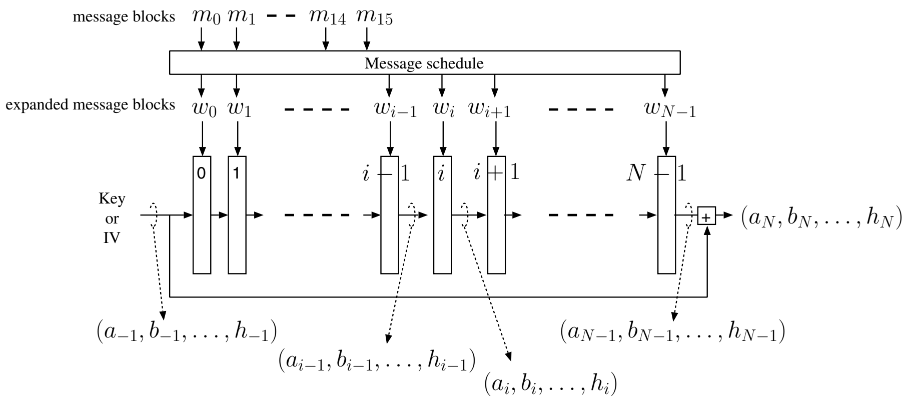
Let $\mathcal{F}_{\ell,n}$ be the set of all functions from $\{0,1\}^{\ell}$ to $\{0,1\}^{n}$. A function f is called a *random function* if f is randomly chosen from $\mathcal{F}_{\ell,n}$. Consider a function $\phi(u,x) : \{0,1\}^{\kappa} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{n}$. After u was fixed, $\phi(u,x)$ can be considered as a function in $\mathcal{F}_{\ell,n}$. Such a function is denoted by $\phi_u(x)$. The function $\phi(u,x)$ is called a *pseudorandom function* if it is hard for an adversary who does not know u to distinguish between $\phi_u(x)$ and a random function $f(x)$ in $\mathcal{F}_{\ell,n}$. Formally, the indistinguishability is measured by the *prf-advantage* of an adversary A that is defined as

$$\mathbf{Adv}^{\mathrm{prf}}_{\phi_k}(A) = \left|\Pr\left[u \xleftarrow{\$} \{0,1\}^{\kappa}; A^{\phi_u} = 1\right] - \Pr\left[f \xleftarrow{\$} \mathcal{F}_{\ell,n}; A^{f} = 1\right]\right| \qquad (1)$$

where A has access to ϕ_u (or f) and returns a bit [2]. Notice that A knows the description of $\phi(u,x)$, but A does not know the chosen key u.

2.2 SHA-2 Compression Function

We here describe the algorithm of the SHA-2 compression function (Fig. 1). In the following, all variables are L bits, an operation '+' denotes an addition



message blocks
m_0 m_1 m_{14} m_{15}
Message schedule
expanded message blocks
w_0 w_1 w_{i-1} w_i w_{i+1} w_{N-1}
0
1
$i-1$
i
$i+1$
$N-1$
Key or IV
+
$(a_N, b_N, \ldots, h_N)$
$(a_{-1}, b_{-1}, \ldots, h_{-1})$
$(a_{i-1}, b_{i-1}, \ldots, h_{i-1})$
$(a_i, b_i, \ldots, h_i)$
$(a_{N-1}, b_{N-1}, \ldots, h_{N-1})$

Fig. 1. SHA-2/N

modulo 2^L, an operation '$\oplus$' denotes the bitwise exclusive-or, and N represents the number of steps. Specifically, $L = 64$ and $N = 80$ for the full-step SHA-512 compression function, and $L = 32$ and $N = 64$ for the full-step SHA-256 one.

Prepare expanded message blocks w_i for given message blocks $m_0, m_1, \ldots, m_{15}$.

$$w_i = \begin{cases} m_i & \text{if } 0 \le i \le 15, \\ \sigma_1(w_{i-2}) + w_{i-7} + \sigma_0(w_{i-15}) + w_{i-16} & \text{if } 16 \le i \le N-1, \end{cases} \tag{2}$$

$$\sigma_0(x) = \begin{cases} \mathrm{ROTR}(1,x) \oplus \mathrm{ROTR}(8,x) \oplus \mathrm{SHR}(7,x) & \text{for SHA-512} \\ \mathrm{ROTR}(7,x) \oplus \mathrm{ROTR}(18,x) \oplus \mathrm{SHR}(3,x) & \text{for SHA-256,} \end{cases}$$

$$\sigma_1(x) = \begin{cases} \mathrm{ROTR}(19,x) \oplus \mathrm{ROTR}(61,x) \oplus \mathrm{SHR}(6,x) & \text{for SHA-512} \\ \mathrm{ROTR}(17,x) \oplus \mathrm{ROTR}(19,x) \oplus \mathrm{SHR}(10,x) & \text{for SHA-256,} \end{cases}$$

where $\mathrm{ROTR}(n,x)$ is a circular shift of x by n positions to the right and $\mathrm{SHR}(n,x)$ is a shift of x by n positions to the right.

Next, $(a_{-1}, b_{-1}, \ldots, h_{-1})$ are given as initial values. For step $i = 0$ to $N-1$, compute variables as Eq. (3) where $a_i, b_i, \ldots, h_i$ are called *chaining values*, and α_i, β_i are called *auxiliary values*.

$$\begin{array}{llll} \alpha_i = h_{i-1} + \Sigma_1(e_{i-1}) + \mathrm{Ch}(e_{i-1}, f_{i-1}, g_{i-1}) + k_i + w_i, & & & \\ \beta_i = \Sigma_0(a_{i-1}) + \mathrm{Maj}(a_{i-1}, b_{i-1}, c_{i-1}), & & & \\ a_i = \alpha_i + \beta_i, & b_i = a_{i-1}, & c_i = b_{i-1}, & d_i = c_{i-1}, \\ e_i = d_{i-1} + \alpha_i, & f_i = e_{i-1}, & g_i = f_{i-1}, & h_i = g_{i-1}, \end{array} \tag{3}$$

where k_i is a step-constant value, and $\Sigma_0, \Sigma_1, \mathrm{Ch}$, and Maj are defined as

$$\begin{aligned} \mathrm{Ch}(x,y,z) &= (x \wedge y) \oplus (\neg x \wedge z), \\ \mathrm{Maj}(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), \\ \Sigma_0(x) &= \begin{cases} \mathrm{ROTR}(28,x) \oplus \mathrm{ROTR}(34,x) \oplus \mathrm{ROTR}(39,x) & \text{for SHA-512,} \\ \mathrm{ROTR}(2,x) \oplus \mathrm{ROTR}(13,x) \oplus \mathrm{ROTR}(22,x) & \text{for SHA-256} \end{cases} \\ \Sigma_1(x) &= \begin{cases} \mathrm{ROTR}(14,x) \oplus \mathrm{ROTR}(18,x) \oplus \mathrm{ROTR}(41,x) & \text{for SHA-512,} \\ \mathrm{ROTR}(6,x) \oplus \mathrm{ROTR}(11,x) \oplus \mathrm{ROTR}(25,x) & \text{for SHA-256.} \end{cases} \end{aligned}$$

Finally, add the initial values to them.

$$\begin{array}{lll} a_N = a_{N-1} + a_{-1}, & b_N = a_{N-1} + b_{-1}, & c_N = c_{N-1} + c_{-1}, \\ d_N = d_{N-1} + d_{-1}, & e_N = e_{N-1} + e_{-1}, & f_N = f_{N-1} + f_{-1}, \\ g_N = g_{N-1} + g_{-1}, & h_N = h_{N-1} + h_{-1}. & \end{array}$$

The output H of the compression function is given by their concatenation.

$$H = a_N \parallel b_N \parallel c_N \parallel d_N \parallel e_N \parallel f_N \parallel g_N \parallel h_N. \tag{4}$$

Replacing $(a_{-1}, b_{-1}, \ldots, h_{-1})$ with a secret key allows us to use the SHA-2 compression function as a keyed compression function. The replacement is often called the *key-via-IV strategy*. This paper discusses the 22 step-reduced SHA-2 compression function with the key-via-IV strategy, which is denoted by *SHA-512/22* (or *SHA-256/22*).

3 22 Step-Reduced SHA-512 Compression Function

This section shows that SHA-512/22 is distinguishable from a random function. We first describe the algorithm for distinguishing them, and then analyze the prf-advantage of this algorithm.

3.1 Distinguishing Algorithm

Suppose that an adversary A has access to an oracle G that is SHA-512/22 ϕ_u or a random function f in $\mathcal{F}_{1024,512}$. The goal of A is to determine whether G is ϕ_u or f. The outline of A is given here. The adversary prepares four messages that are correlated with each other. After the adversary received their four hashes from the oracle G, the adversary computes two differences from four hashes. If two differences satisfy a condition simultaneously, then the adversary outputs 1, which implicitly means that G is guessed to be SHA-512/22, otherwise the adversary outputs 0. Notice that A has access to G only four times. The detail of A is described below.

A-1. Denote four messages $M^{(j)}$ for $j = 0, 1, 2, 3$ by

$$M^{(j)} = m_0^{(j)} \parallel m_1^{(j)} \parallel m_2^{(j)} \parallel m_3^{(j)} \parallel \cdots \parallel m_{14}^{(j)} \parallel m_{15}^{(j)}.$$

Choose message blocks $m_0^{(0)}, m_1^{(0)}, \ldots, m_{11}^{(0)}, m_{14}^{(0)}, m_{15}^{(0)}$ independently and randomly. Set other message blocks except for $m_{12}^{(j)}, m_{13}^{(j)}$ as follows.

$$\begin{aligned}
&m_0^{(0)} = m_0^{(1)} = m_0^{(2)} = m_0^{(3)}, \quad m_1^{(0)} = m_1^{(1)} = m_1^{(2)} = m_1^{(3)},\\
&\ldots\\
&m_{10}^{(0)} = m_{10}^{(1)} = m_{10}^{(2)} = m_{10}^{(3)}, \quad m_{11}^{(0)} = m_{11}^{(1)} = m_{11}^{(2)} = m_{11}^{(3)},\\
&m_{14}^{(0)} = m_{14}^{(1)} = m_{14}^{(2)} = m_{14}^{(3)}, \quad m_{15}^{(0)} = m_{15}^{(1)} = m_{15}^{(2)} = m_{15}^{(3)}.
\end{aligned}$$

A-2. Set message blocks $m_{12}^{(j)}$ for $j = 0, 1, 2, 3$ as follows.

$$\begin{aligned}
m_{12}^{(0)} &= \mathsf{0}, \quad & m_{12}^{(1)} &= m_{12}^{(0)} + \mathsf{1}_{63} = \mathsf{1}_{63},\\
m_{12}^{(2)} &= \mathsf{1}_{41}, & m_{12}^{(3)} &= m_{12}^{(2)} + \mathsf{1}_{63} = \mathsf{1}_{63,41},
\end{aligned}$$

where $\mathsf{1}_t$ means that the tth bit is one and the others are zero. Note that the least significant bit is the 0th bit. For example, $\mathsf{1}_{63}$ and $\mathsf{1}_{63,41}$ are

$$\mathsf{1}_{63} = 8000000000000000, \quad \mathsf{1}_{63,41} = 8000020000000000$$

in hexadecimal.

A-3. Set message blocks $m_{13}^{(j)}$ for $j = 0, 1, 2, 3$ as follows.

$$\begin{aligned}
m_{13}^{(0)} &= -k_{13}, \quad & m_{13}^{(1)} &= m_{13}^{(0)} + (-\Sigma_1(\mathsf{1}_{63})),\\
m_{13}^{(2)} &= -k_{13}, & m_{13}^{(3)} &= m_{13}^{(2)} + (-\Sigma_1(\mathsf{1}_{63})),
\end{aligned}$$

where k_{13} is the 13-step constant. Now, all message blocks have been determined.

A-4. Send four messages $M^{(0)}, M^{(1)}, M^{(2)}, M^{(3)}$ to the oracle G, and receive their hashes $H^{(0)}, H^{(1)}, H^{(2)}, H^{(3)}$. Here, we regard $H^{(j)}$ as the concatenation of eight 64-bit words that corresponds to Eq. (4).

$$H^{(j)} = a^{(j)} \parallel b^{(j)} \parallel c^{(j)} \parallel d^{(j)} \parallel e^{(j)} \parallel f^{(j)} \parallel g^{(j)} \parallel h^{(j)}$$

A-5. Compute the modular difference of $h^{(\ell+1)}$ and $h^{(\ell)}$ for $\ell = 0, 2$.

$$\delta h^{\langle \ell \rangle} = h^{(\ell+1)} - h^{(\ell)} \bmod 2^{64}. \tag{5}$$

A-6. If the following conditions are satisfied for all $\ell = 0, 2$, then A outputs 1, otherwise A outputs 0.

$$\delta h^{\langle \ell \rangle}[k] = \begin{cases} 0 & \text{for } k = 0, 1, \ldots, 5, \\ 1 & \text{for } k = 6, \end{cases} \tag{6}$$

where $\delta h^{\langle \ell \rangle}[k]$ denotes the kth bit of $\delta h^{\langle \ell \rangle}$.

We here evaluate the prf-advantage of A. Suppose that G is a random function f. Since $\delta h^{\langle \ell \rangle}$ satisfies Eq. (6) with probability 2^{-7} for each ℓ independently, A outputs 1 with probability 2^{-14}. Denoting by RF the event that G is the random function, we formally write the probability as

$$\Pr\left[A^G = 1 | \mathsf{RF}\right] = 2^{-14}. \tag{7}$$

Suppose that G is SHA-512/22 ϕ_u. As shown in Sect. 3.2, the probability that A outputs 1 is

$$\Pr\left[A^G = 1 | \mathsf{SHA}\right] > 2^{-11.749376} \tag{8}$$

where SHA denotes the event that G is SHA-512/22. Therefore, the prf-advantage of A is given by

$$\begin{aligned} \mathbf{Adv}^{\mathrm{prf}}_{\phi_u}(A) &= \left| \Pr\left[u \xleftarrow{\$} \{0,1\}^{512}; A^{\phi_u} = 1\right] - \Pr\left[f \xleftarrow{\$} \mathcal{F}_{1024,512}; A^f = 1\right] \right| \\ &= \left| \Pr\left[A^G = 1 | \mathsf{SHA}\right] - \Pr\left[A^G = 1 | \mathsf{RF}\right] \right| \\ &> 2^{-11.749376} - 2^{-14} \approx 2^{-12.089694}. \end{aligned}$$

This prf-advantage suggests that SHA-512/22 is easily distinguishable from the random function. The above algorithm A has access to G only four times. If an attacker performs this algorithm repeatedly, then the attacker can improve the probability that he succeeds in distinguishing between SHA-512/22 and the random function.

3.2 Analysis

This subsection explains why the probability that A outputs 1 is given by Eq. (8) when G is SHA-512/22. The algorithm in Sect. 3.1 is based on the differential path shown in Table 1. In Table 1, the first column indicates the step number of

Table 1. Differential path for SHA-512/22 ($\ell = 0, 2$)

Step i	$\Delta w_i^{\langle\ell\rangle}$	$\Delta\alpha_i^{\langle\ell\rangle}$	$\Delta\beta_i^{\langle\ell\rangle}$	$\Delta a_i^{\langle\ell\rangle}$	$\Delta b_i^{\langle\ell\rangle}$	$\Delta c_i^{\langle\ell\rangle}$	$\Delta d_i^{\langle\ell\rangle}$	$\Delta e_i^{\langle\ell\rangle}$	$\Delta f_i^{\langle\ell\rangle}$	$\Delta g_i^{\langle\ell\rangle}$	$\Delta h_i^{\langle\ell\rangle}$
0-11	0	0	0	0	0	0	0	0	0	0	0
12	$\Delta w_{12}^{\langle\ell\rangle}$	$\Delta w_{12}^{\langle\ell\rangle}$	0	$\Delta w_{12}^{\langle\ell\rangle}$	0	0	0	$\Delta w_{12}^{\langle\ell\rangle}$	0	0	0
13	$\Delta w_{13}^{\langle\ell\rangle}$	$\boxed{0}$	$\Delta\beta_{13}^{\langle\ell\rangle}$	$\boxed{\Delta a_{13}^{\langle\ell\rangle}}$	$\Delta w_{12}^{\langle\ell\rangle}$	0	0	0	$\Delta w_{12}^{\langle\ell\rangle}$	0	0
14	0	$\boxed{0}$	$\Delta\beta_{14}^{\langle\ell\rangle}$	$\Delta a_{14}^{\langle\ell\rangle}$	$\Delta a_{13}^{\langle\ell\rangle}$	$\Delta w_{12}^{\langle\ell\rangle}$	0	0	0	$\Delta w_{12}^{\langle\ell\rangle}$	0
15	0	$\boxed{0}$	$*$	$*$	$\Delta a_{14}^{\langle\ell\rangle}$	$\Delta a_{13}^{\langle\ell\rangle}$	$\Delta w_{12}^{\langle\ell\rangle}$	0	0	0	$\Delta w_{12}^{\langle\ell\rangle}$
16	0	$\boxed{1_{63}}$	$*$	$*$	$*$	$\Delta a_{14}^{\langle\ell\rangle}$	$\Delta a_{13}^{\langle\ell\rangle}$	0	0	0	0
17	0	$\boxed{0}$	$*$	$*$	$*$	$*$	$\Delta a_{14}^{\langle\ell\rangle}$	$\Delta e_{17}^{\langle\ell\rangle}$	0	0	0
18	0	$\Delta\alpha_{18}^{\langle\ell\rangle}$	$*$	$*$	$*$	$*$	$*$	$\Delta e_{18}^{\langle\ell\rangle}$	$\Delta e_{17}^{\langle\ell\rangle}$	0	0
19	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$\Delta e_{18}^{\langle\ell\rangle}$	$\Delta e_{17}^{\langle\ell\rangle}$	0
20	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$\Delta e_{18}^{\langle\ell\rangle}$	$\Delta e_{17}^{\langle\ell\rangle}$
21	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$\Delta e_{18}^{\langle\ell\rangle}$

SHA-512/22 (ref. Fig. 1), the second column indicates the difference of expanded message blocks given by Eq. (2), and the third column to the twelfth column indicate the difference of values given by Eq. (3).

The difference $\Delta^{\langle\rangle}$ in Table 1 is the bitwise exclusive-or difference. For example, $\Delta w_{12}^{\langle\ell\rangle}$ is the bitwise exclusive-or difference of $w_{12}^{(\ell+1)}$ and $w_{12}^{(\ell)}$, that is,

$$\Delta w_{12}^{\langle\ell\rangle} = w_{12}^{(\ell+1)} \oplus w_{12}^{(\ell)},$$

where $\ell = 0, 2$. The symbol 0 means that the bitwise exclusive-or difference is zero, and the symbol $*$ means an ignored difference because the difference has no effect on our analysis.

We here explain the relationship between each item in Table 1 and steps[1] of the algorithm A. First, the difference on expanded message blocks exists only in $\Delta w_{12}^{\langle\ell\rangle}$ and $\Delta w_{13}^{\langle\ell\rangle}$ because of step A-1 to step A-3.

Second, before step 11, expanded message blocks, auxiliary values, and chaining values have no difference.

$$\begin{aligned}
&w_{11}^{(0)} = w_{11}^{(1)} = w_{11}^{(2)} = w_{11}^{(3)},\\
&\alpha_{11}^{(0)} = \alpha_{11}^{(1)} = \alpha_{11}^{(2)} = \alpha_{11}^{(3)},\quad \beta_{11}^{(0)} = \beta_{11}^{(1)} = \beta_{11}^{(2)} = \beta_{11}^{(3)},\\
&a_{11}^{(0)} = a_{11}^{(1)} = a_{11}^{(2)} = a_{11}^{(3)},\quad b_{11}^{(0)} = b_{11}^{(1)} = b_{11}^{(2)} = b_{11}^{(3)},\\
&\ldots\\
&g_{11}^{(0)} = g_{11}^{(1)} = g_{11}^{(2)} = g_{11}^{(3)},\quad h_{11}^{(0)} = h_{11}^{(1)} = h_{11}^{(2)} = h_{11}^{(3)}.
\end{aligned}$$

[1] In this article, 'step A-?' indicates the step of the algorithm A, and 'step ?' indicates the step of SHA-512/22.

This is because step A-1 chooses message blocks in such a way that $m_i^{(0)} = m_i^{(1)} = m_i^{(2)} = m_i^{(3)}$ for $i = 0, 1, \ldots, 11$.

Third, let us consider step 12. Recall that

$$\alpha_{12}^{(\ell)} = h_{11}^{(\ell)} + \Sigma_1(e_{11}^{(\ell)}) + \mathrm{Ch}(e_{11}^{(\ell)}, f_{11}^{(\ell)}, g_{11}^{(\ell)}) + k_{12} + w_{12}^{(\ell)},$$
$$\beta_{12}^{(\ell)} = \Sigma_0(a_{11}^{(\ell)}) + \mathrm{Maj}(a_{11}^{(\ell)}, b_{11}^{(\ell)}, c_{11}^{(\ell)}).$$

Since step 11 does not produce any difference and $w_{12}^{(\ell+1)} = w_{12}^{(\ell)} + 1_{63}$ for $\ell = 0, 2$, we have $\alpha_{12}^{(\ell+1)} = \alpha_{12}^{(\ell)} + 1_{63}$ and $\beta_{12}^{(\ell+1)} = \beta_{12}^{(\ell)}$ for $\ell = 0, 2$. Since $a_{12}^{(j)} = \alpha_{12}^{(j)} + \beta_{12}^{(j)}$ and $e_{12}^{(j)} = d_{11}^{(j)} + \alpha_{12}^{(j)}$ for $j = 0, 1, 2, 3$, we obtain

$$\Delta a_{12}^{\langle \ell \rangle} = \Delta w_{12}^{\langle \ell \rangle} = 1_{63}, \quad \Delta e_{12}^{\langle \ell \rangle} = \Delta w_{12}^{\langle \ell \rangle} = 1_{63} \text{ for } \ell = 0, 2.$$

Notice that the modular addition of 1_{63} is equivalent to the exclusive-or at the 63rd bit position. In this paper, we often interchange these operations. In addition, other chaining values have no difference.

Fourth, skipping step 12 to step 21, we consider the final step. As illustrated in Fig. 1, the secret key $(a_{-1}, b_{-1}, \cdots, h_{-1})$ is added to chaining values at the final. Since the secret key does not depend on messages, the modular difference of Eq. (5) is transformed into

$$\begin{aligned}\delta h^{\langle \ell \rangle} &= h^{(\ell+1)} - h^{(\ell)} \bmod 2^{64} \\ &= (h_{21}^{(\ell+1)} + h_{-1}) - (h_{21}^{(\ell)} + h_{-1}) \bmod 2^{64} \\ &= h_{21}^{(\ell+1)} - h_{21}^{(\ell)} \bmod 2^{64}.\end{aligned}$$

Since the condition of Eq. (6) focuses only on the difference of the least significant seven bits of $h^{(\ell+1)}$ and $h^{(\ell)}$, it can be written using the bitwise exclusive-or difference as follows.

$$\Delta h_{21}^{\langle \ell \rangle}[k] = \begin{cases} 0 & \text{for } k = 0, 1, \ldots, 5, \\ 1 & \text{for } k = 6. \end{cases}$$

Since $h_{21}^{(j)} = e_{18}^{(j)}$ for $j = 0, 1, 2, 3$, the above condition is equivalent to

$$\Delta e_{18}^{\langle \ell \rangle}[k] = \begin{cases} 0 & \text{for } k = 0, 1, \ldots, 5, \\ 1 & \text{for } k = 6. \end{cases} \tag{9}$$

From now on, we will discuss Eq. (9) instead of Eq. (6).

Finally, to discuss the differential path from step 12 to step 18, we consider three conditions, which are boxed in Table 1.

C1: For $\ell = 0, 2$,

$$\Delta \alpha_{13}^{\langle \ell \rangle} = 0, \quad \Delta \alpha_{14}^{\langle \ell \rangle} = 0, \quad \Delta \alpha_{15}^{\langle \ell \rangle} = 0, \quad \Delta \alpha_{16}^{\langle \ell \rangle} = 1_{63}.$$

C2: For $\ell = 0, 2$,

$$\Delta a_{13}^{\langle\ell\rangle} = 1_{35,29,24},$$

where $1_{35,29,24}$ means that the 35th, 29th, 24th bits are one and the others are zero, that is, $1_{35,29,24} = 0000000821000000$ in hexadecimal.

C3: For $j = 0, 1, 2, 3$,

$$\alpha_{17}^{(j)}[k] = 0 \quad \text{for } 35 \leq \exists k \leq 40,$$

where k may depend on j. Unlike other two conditions, this condition is not on the difference.

We denote by $\mathsf{c1}$ ($\mathsf{c2}$, $\mathsf{c3}$) the event that the condition C1 (C2, C3, resp.) is satisfied. Using probabilities of these events, we can write the probability[2] that A outputs 1 as

$$\begin{aligned}\Pr\left[A^G = 1\right] &= \Pr\left[A^G = 1 | \mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}\right] \Pr\left[\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}\right] \\ &\quad + \Pr\left[A^G = 1 | \overline{\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}}\right] \Pr\left[\overline{\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}}\right] \\ &\geq \Pr\left[A^G = 1 | \mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}\right] \Pr\left[\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}\right]\end{aligned}$$

We firstly show $\Pr\left[A^G = 1 | \mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}\right] = 1$. Now, since the event of $A^G = 1$ is equivalent to Eq. (9), we consider $\Delta e_{18}^{\langle\ell\rangle}$ under the assumption that three conditions hold. Table 1 shows that $\Delta e_{18}^{\langle\ell\rangle}$ is produced by $\Delta a_{14}^{\langle\ell\rangle}$ $(= \Delta d_{17}^{\langle\ell\rangle})$ and $\Delta e_{17}^{\langle\ell\rangle}$ for $\ell = 0, 2$. Consider $a_{14}^{(j)}$ that is calculated by the following equation.

$$a_{14}^{(j)} = \alpha_{14}^{(j)} + \beta_{14}^{(j)} \quad \text{where } j = 0, 1, 2, 3.$$

Since $\Delta\alpha_{14}^{\langle\ell\rangle} = \mathsf{0}$ for $\ell = 0, 2$ because of C1, only $\Delta\beta_{14}^{\langle\ell\rangle}$ causes $\Delta a_{14}^{\langle\ell\rangle}$. Using C2, $\Delta b_{13}^{\langle\ell\rangle} = 1_{63}$, and $\Delta c_{13}^{\langle\ell\rangle} = \mathsf{0}$, we can write $\beta_{14}^{(\ell)}$ and $\beta_{14}^{(\ell+1)}$ as

$$\begin{aligned}\beta_{14}^{(\ell)} &= \Sigma_0(a_{13}^{(\ell)}) + \mathrm{Maj}(a_{13}^{(\ell)}, b_{13}^{(\ell)}, c_{13}^{(\ell)}), \\ \beta_{14}^{(\ell+1)} &= \Sigma_0(a_{13}^{(\ell+1)}) + \mathrm{Maj}(a_{13}^{(\ell+1)}, b_{13}^{(\ell+1)}, c_{13}^{(\ell+1)}) \\ &= \Sigma_0(a_{13}^{(\ell)} \oplus 1_{35,29,24}) + \mathrm{Maj}(a_{13}^{(\ell)} \oplus 1_{35,29,24}, b_{13}^{(\ell)} \oplus 1_{63}, c_{13}^{(\ell)}) \\ &= (\Sigma_0(a_{13}^{(\ell)}) \oplus \Sigma_0(1_{35,29,24})) + \mathrm{Maj}(a_{13}^{(\ell)} \oplus 1_{35,29,24}, b_{13}^{(\ell)} \oplus 1_{63}, c_{13}^{(\ell)}) \\ &= (\Sigma_0(a_{13}^{(\ell)}) \oplus 1_{59,49,7}) + \mathrm{Maj}(a_{13}^{(\ell)} \oplus 1_{35,29,24}, b_{13}^{(\ell)} \oplus 1_{63}, c_{13}^{(\ell)}).\end{aligned}$$

Comparing $\beta_{14}^{(\ell)}$ with $\beta_{14}^{(\ell+1)}$, we find that

$$\Delta\beta_{14}^{\langle\ell\rangle}[k] = 0 \quad \text{for } k = 0, 1, \ldots, 6.$$

Hence, we obtain

$$\Delta a_{14}^{\langle\ell\rangle}[k] = \Delta d_{17}^{\langle\ell\rangle}[k] = 0 \quad \text{for } k = 0, 1, \ldots, 6. \tag{10}$$

[2] We leave out SHA from $\Pr\left[A^G = 1 | \mathsf{SHA}\right]$.

Recall that another difference that produces $\Delta e_{18}^{\langle \ell \rangle}$ is $\Delta e_{17}^{\langle \ell \rangle}$. Here, $e_{17}^{(\ell)}$ and $e_{17}^{(\ell+1)}$ are given by

$$e_{17}^{(\ell)} = d_{16}^{(\ell)} + \alpha_{17}^{(\ell)}, \quad e_{17}^{(\ell+1)} = d_{16}^{(\ell+1)} + \alpha_{17}^{(\ell+1)}.$$

Using the following equations and the condition C3,

$$\begin{aligned}
d_{16}^{(\ell)} &= a_{13}^{(\ell)} \\
d_{16}^{(\ell+1)} &= a_{13}^{(\ell+1)} = a_{13}^{(\ell)} \oplus 1_{35,29,24} \\
\alpha_{17}^{(\ell)} &= h_{16}^{(\ell)} + \Sigma_1(e_{16}^{(\ell)}) + \mathrm{Ch}(e_{16}^{(\ell)}, f_{16}^{(\ell)}, g_{16}^{(\ell)}) + k_{17} + w_{17}^{(\ell)} \\
\alpha_{17}^{(\ell+1)} &= h_{16}^{(\ell+1)} + \Sigma_1(e_{16}^{(\ell+1)}) + \mathrm{Ch}(e_{16}^{(\ell+1)}, f_{16}^{(\ell+1)}, g_{16}^{(\ell+1)}) + k_{17} + w_{17}^{(\ell+1)} \\
&= h_{16}^{(\ell)} + \Sigma_1(e_{16}^{(\ell)}) + \mathrm{Ch}(e_{16}^{(\ell)}, f_{16}^{(\ell)}, g_{16}^{(\ell)}) + k_{17} + w_{17}^{(\ell)} \\
&= \alpha_{17}^{(\ell)}
\end{aligned}$$

we find that

$$\Delta e_{17}^{\langle \ell \rangle}[k] = \begin{cases} 0 & \text{for } k = 0, 1, \ldots, 23, 41, 42, \ldots, 63 \\ 1 & \text{for } k = 24. \end{cases} \tag{11}$$

because the condition C3 ensures that the carry caused by '$\oplus 1_{35,29,24}$' has no effect on the 41st bit. Next, $e_{18}^{(\ell)}$ and $e_{18}^{(\ell+1)}$ are given by

$$e_{18}^{(\ell)} = d_{17}^{(\ell)} + \alpha_{18}^{(\ell)}, \quad e_{18}^{(\ell+1)} = d_{17}^{(\ell+1)} + \alpha_{18}^{(\ell+1)}. \tag{12}$$

From Eq. (10), $d_{17}^{(\ell)}$ and $d_{17}^{(\ell+1)}$ produce no difference on the least significant seven bits of $e_{18}^{(\ell)}$ and $e_{18}^{(\ell+1)}$. Accordingly, consider difference on the least significant seven bits of $\alpha_{18}^{(\ell)}$ and $\alpha_{18}^{(\ell+1)}$.

$$\begin{aligned}
\alpha_{18}^{(\ell)} &= h_{17}^{(\ell)} + \Sigma_1(e_{17}^{(\ell)}) + \mathrm{Ch}(e_{17}^{(\ell)}, f_{17}^{(\ell)}, g_{17}^{(\ell)}) + k_{18} + w_{18}^{(\ell)} \\
\alpha_{18}^{(\ell+1)} &= h_{17}^{(\ell+1)} + \Sigma_1(e_{17}^{(\ell+1)}) + \mathrm{Ch}(e_{17}^{(\ell+1)}, f_{17}^{(\ell+1)}, g_{17}^{(\ell+1)}) + k_{18} + w_{18}^{(\ell+1)} \\
&= h_{17}^{(\ell)} + \Sigma_1(e_{17}^{(\ell+1)}) + \mathrm{Ch}(e_{17}^{(\ell+1)}, f_{17}^{(\ell)}, g_{17}^{(\ell)}) + k_{18} + w_{18}^{(\ell)}
\end{aligned}$$

Note that Ch() does not produce any difference on the 0th to 6th bit positions because of Eq. (11). Furthermore, Eq. (11) means that

$$\begin{aligned}
\Sigma_1(e_{17}^{(\ell+1)})[k] &= e_{17}^{(\ell+1)}[k+14] \oplus e_{17}^{(\ell+1)}[k+18] \oplus e_{17}^{(\ell+1)}[k+41] \\
&= e_{17}^{(\ell)}[k+14] \oplus e_{17}^{(\ell)}[k+18] \oplus e_{17}^{(\ell)}[k+41] \\
&= \Sigma_1(e_{17}^{(\ell)})[k],
\end{aligned} \tag{13}$$

for $k = 0, 1, \ldots, 5$, and

$$\begin{aligned}
\Sigma_1(e_{17}^{(\ell+1)})[6] &= e_{17}^{(\ell+1)}[20] \oplus e_{17}^{(\ell+1)}[24] \oplus e_{17}^{(\ell+1)}[47] \\
&= e_{17}^{(\ell)}[20] \oplus (e_{17}^{(\ell)}[24] \oplus 1) \oplus e_{17}^{(\ell)}[47] \\
&= \Sigma_1(e_{17}^{(\ell)})[6] \oplus 1.
\end{aligned} \tag{14}$$

Combining Eq. (13) and Eq. (14) yields

$$\Delta\alpha_{18}^{\langle\ell\rangle}[k] = \begin{cases} 0 & \text{for } k = 0, 1, \ldots, 5 \\ 1 & \text{for } k = 6. \end{cases}$$

Substituting the above equation into Eq. (12) gives

$$\Delta e_{18}^{\langle\ell\rangle}[k] = \begin{cases} 0 & \text{for } k = 0, 1, \ldots, 5 \\ 1 & \text{for } k = 6. \end{cases} \tag{15}$$

We have proved that $\Pr\left[A^G = 1 | \mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}\right] = 1$, that is, if three conditions C1, C2, C3 are satisfied, then A always outputs 1. Notice that satisfying three conditions is a sufficient condition of the event that A outputs 1, but is not its necessary condition.

We next evaluate the probability $\Pr[\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}]$ using the following formula.

$$\Pr[\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}] = \Pr[\mathsf{c3} | \mathsf{c2} \wedge \mathsf{c1}] \cdot \Pr[\mathsf{c2} | \mathsf{c1}] \cdot \Pr[\mathsf{c1}]$$

Appendix will show that

$$\Pr[\mathsf{c1}] > 2^{-6.007374}, \quad \Pr[\mathsf{c2}|\mathsf{c1}] > 2^{-5.696199}, \quad \Pr[\mathsf{c3}|\mathsf{c2} \wedge \mathsf{c1}] > 1 - 2^{-5.00},$$

respectively. Hence, we obtain

$$\begin{aligned} \Pr\left[A^G = 1\right] &\geq \Pr[\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}] \\ &> 2^{-11.749376}, \end{aligned}$$

which was used in Eq. (8).

Remark. Our argument assumes that the non-focused part of SHA-512/22 behaves as if it is a random function. To support the argument, we here show probabilities obtained by computer experiment. When G is SHA-512/22,

$$\begin{aligned} &\Pr\left[A^G = 1\right] = 2^{-8.484797}, \quad \Pr[\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}] = 2^{-11.749694}, \\ &\Pr[\mathsf{c1}] = 2^{-6.007577}, \quad \Pr[\mathsf{c2}|\mathsf{c1}] = 2^{-5.696744}, \quad \Pr[\mathsf{c3}|\mathsf{c2} \wedge \mathsf{c1}] = 2^{-0.045373}. \end{aligned}$$

The probability $\Pr\left[A^G = 1\right]$ is much larger than $\Pr[\mathsf{c1} \wedge \mathsf{c2} \wedge \mathsf{c3}]$ because there are many different paths to satisfy Eq. (9) other than the conditions C1, C2, C3. For example, the condition C2 can relax a little.

4 22 Step-Reduced SHA-256 Compression Function

Since the structure of SHA-256 is similar to that of SHA-512, the distinguishing algorithm described in Sect. 3.1 can be easily modified to apply to SHA-256/22. The modified algorithm B is described below.

B-1. This step is the same as step A-1 in Sect. 3.1.

B-2. Set message blocks $m_{12}^{(j)}$ for $j = 0, 1, 2, 3$ as follows.

$$m_{12}^{(0)} = \mathtt{0}, \quad m_{12}^{(1)} = m_{12}^{(0)} + \mathtt{1}_{31} = \mathtt{1}_{31},$$
$$m_{12}^{(2)} = \mathtt{1}_{19}, \; m_{12}^{(3)} = m_{12}^{(2)} + \mathtt{1}_{31} = \mathtt{1}_{31,19}.$$

B-3. Set message blocks $m_{13}^{(j)}$ for $j = 0, 1, 2, 3$ as follows.

$$m_{13}^{(0)} = -k_{13}, \;\; m_{13}^{(1)} = m_{13}^{(0)} + (-\Sigma_1(\mathtt{1}_{31})),$$
$$m_{13}^{(2)} = -k_{13}, \;\; m_{13}^{(3)} = m_{13}^{(2)} + (-\Sigma_1(\mathtt{1}_{31})).$$

B-4. Send four messages $M^{(0)}, M^{(1)}, M^{(2)}, M^{(3)}$ to the oracle G, and receive their hashes $H^{(0)}, H^{(1)}, H^{(2)}, H^{(3)}$. Here, we regard $H^{(j)}$ as the concatenation of eight 32-bit words that corresponds to Eq. (4).

$$H^{(j)} = a^{(j)} \parallel b^{(j)} \parallel c^{(j)} \parallel d^{(j)} \parallel e^{(j)} \parallel f^{(j)} \parallel g^{(j)} \parallel h^{(j)}$$

B-5. Compute the modular difference of $h^{(\ell+1)}$ and $h^{(\ell)}$ for $\ell = 0, 2$.

$$\delta h^{\langle \ell \rangle} = h^{(\ell+1)} - h^{(\ell)} \bmod 2^{32}.$$

B-6. If the following conditions are satisfied for all $\ell = 0, 2$, then B outputs 1, otherwise B outputs 0.

$$\delta h^{\langle \ell \rangle}[k] = \begin{cases} 0 & \text{for } k = 0, 1, 2, \\ 1 & \text{for } k = 3. \end{cases} \tag{16}$$

This algorithm is based on a differential path similar to Table 1. The difference $\mathtt{1}_{19}$ in step B-2 was chosen to make $\Pr\left[\Delta\alpha_{13}^{\langle \ell \rangle} = \mathtt{0}\right]$ large. Our experiment showed that Eq. (16) was satisfied with probability $2^{-5.584973}$. If G is a random function in $\mathcal{F}_{512,256}$, then Eq. (16) is satisfied with probability 2^{-8}. Hence, the prf-advantage of B is

$$\begin{aligned}\mathbf{Adv}_{\phi_u}^{\mathrm{prf}}(B) &= \left|\Pr\left[u \xleftarrow{\$} \{0,1\}^{256}; B^{\phi_u} = 1\right] - \Pr\left[f \xleftarrow{\$} \mathcal{F}_{512,256}; B^f = 1\right]\right| \\ &= 2^{-5.584973} - 2^{-8} \approx 2^{-5.884535}.\end{aligned}$$

5 Concluding Remarks

The previous analysis of hash functions focused on collision-resistance of underlying compression functions. However, applications often require that hash functions have not only the collision-resistant property but also the pseudorandom-function property. Except for HMAC security against collision-finding attacks, the pseudorandom-function property of compression functions has not been studied. This paper provided the first result on the pseudorandom-function property of the SHA-2 compression function. We showed practical attacks for distinguishing between the 22 step-reduced SHA-2 compression function with the key-via-IV strategy and a random function. In the case of the 22 step-reduced SHA-512 compression function with the key-via-IV strategy, there exists the adversary such that the prf-advantage is larger than $2^{-12.089694}$ when the number of queries is four. Increasing the number of queries allows the adversary to improve the prf-advantage.

Acknowledgments

The authors wish to acknowledge valuable discussions with Dr. Yoshida and Dr. Ideguchi at Hitachi, Ltd. and Prof. Ohta at The University of Electro-Communications. The authors also thank anonymous reviewers for their comments. This research was supported by the National Institute of Information and Communications Technology, Japan.

References

1. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006), http://eprint.iacr.org/2006/043.pdf
2. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
3. Bellare, M., Ristenpart, T.: Hash functions in the dedicated-key setting: Design choices and MPP transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007), http://eprint.iacr.org/
4. 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)
5. 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), http://eprint.iacr.org/
6. Dunkelman, O., Keller, N., Kim, J.: Related-key rectangle attack on the full SHACAL-1. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 28–44. Springer, Heidelberg (2007)
7. Fouque, P.-A., Leurent, G., Nguyenh, 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), ftp://ftp.di.ens.fr/pub/users/pnguyen/Crypto07.pdf
8. Handschuh, H., Naccache, D.: SHACAL (2000), http://www.nessie.eu.org/nessie/
9. Hirose, S., Park, J.H., Yun, A.: A simple variant of the Merkle-Damgård scheme with a permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)

10. Hong, S., Kim, J., Kim, G., Sung, J., Lee, C., Lee, S.: 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)
11. Indesteege, S., Mendel, F., Preneel, B., Rechberger, C., Rijmen, V.: Collisions and other non-random properties for step-reduced SHA-256. Cryptology ePrint Archive, Report 2008/131 (2008), http://eprint.iacr.org/
12. Kim, J., Moon, D., Lee, W., Hong, S., Lee, S., Jung, S.: Amplified boomerang attack against reduced-round SHACAL. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 243–253. Springer, Heidelberg (2002)
13. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of step-reduced SHA-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 126–143. Springer, Heidelberg (2006)
14. National Institute of Standards and Technology, Secure hash standard, Federal Information Processing Standards Publication 180-2 (August 2002), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
15. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: a synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
16. Sanadhya, S.K., Sarkar, P.: Collision attacks against 22-step SHA-512. Cryptology ePrint Archive, Report 2008/270 (2008), http://eprint.iacr.org/
17. Sanadhya, S.K., Sarkar, P.: Non-linear reduced round attacks against SHA-2 hash family. Cryptology ePrint Archive, Report 2008/174 (2008), http://eprint.iacr.org/
18. Simon, D.R.: Findings collisions on a one-way street: Can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
19. 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)

A Probability Pr [c1]

We evaluate Pr [c1] using the following formula.

$$\begin{aligned}\Pr[\mathsf{c1}] = \Pr\left[\Delta\alpha_{13}^{\langle\ell\rangle} = 0\right] \cdot \Pr\left[\Delta\alpha_{14}^{\langle\ell\rangle} = 0 | \Delta\alpha_{13}^{\langle\ell\rangle} = 0\right] \\ \cdot\Pr\left[\Delta\alpha_{15}^{\langle\ell\rangle} = 0 | \Delta\alpha_{14}^{\langle\ell\rangle} = 0 \wedge \Delta\alpha_{13}^{\langle\ell\rangle} = 0\right] \\ \cdot\Pr\left[\Delta\alpha_{16}^{\langle\ell\rangle} = 1_{63} | \Delta\alpha_{15}^{\langle\ell\rangle} = 0 \wedge \Delta\alpha_{14}^{\langle\ell\rangle} = 0 \wedge \Delta\alpha_{13}^{\langle\ell\rangle} = 0\right]\end{aligned}$$

This appendix provides only a part of Pr [c1] because of limitation of pages.

A.1 Probability of $\Delta\alpha_{13}^{\langle\ell\rangle} = 0$

Recall that

$$\begin{aligned}\alpha_{13}^{(0)} &= h_{12}^{(0)} + \Sigma_1(e_{12}^{(0)}) + \mathrm{Ch}(e_{12}^{(0)}, f_{12}^{(0)}, g_{12}^{(0)}) + k_{13} + w_{13}^{(0)}, \\ \alpha_{13}^{(1)} &= h_{12}^{(1)} + \Sigma_1(e_{12}^{(1)}) + \mathrm{Ch}(e_{12}^{(1)}, f_{12}^{(1)}, g_{12}^{(1)}) + k_{13} + w_{13}^{(1)}, \\ h_{12}^{(1)} &= h_{12}^{(0)}, \quad f_{12}^{(1)} = f_{12}^{(0)}, \quad g_{12}^{(1)} = g_{12}^{(0)}, \\ e_{12}^{(1)} &= e_{12}^{(0)} + 1_{63}, \quad w_{13}^{(1)} = w_{13}^{(0)} + (-\Sigma_1(1_{63})).\end{aligned}$$

We find that $\varDelta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}$ if and only if

$$\varSigma_1(e_{12}^{(0)} + 1_{63}) = \varSigma_1(e_{12}^{(0)}) + \varSigma_1(1_{63}) \ \wedge \ f_{12}^{(0)}[63] = g_{12}^{(0)}[63].$$

Since $\varSigma_1(1_{63}) = 1_{49,45,22}$, the above conditions are equivalent to

$$\varSigma_1(e_{12}^{(0)})[k] = 0 \ \text{ for } k = 49, 45, 22 \ \wedge \ f_{12}^{(0)}[63] = g_{12}^{(0)}[63]. \tag{17}$$

Suppose that other parts of SHA-512/22 behave as if it is a random function. From Eq. (17), we obtain

$$\Pr\left[\varDelta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}\right] = 2^{-4}.$$

Supposing that $\varDelta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}$, we evaluate $\Pr\left[\varDelta\alpha_{13}^{\langle 2\rangle} = \mathsf{0} | \varDelta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}\right]$.

$$\begin{aligned}
\alpha_{13}^{(2)} &= h_{12}^{(2)} + \varSigma_1(e_{12}^{(2)}) + \mathrm{Ch}(e_{12}^{(2)}, f_{12}^{(2)}, g_{12}^{(2)}) + k_{13} + w_{13}^{(2)} \\
&= h_{12}^{(0)} + \varSigma_1(e_{12}^{(0)} + 1_{41}) + \mathrm{Ch}(e_{12}^{(0)} + 1_{41}, f_{12}^{(0)}, g_{12}^{(0)}) + k_{13} + w_{13}^{(0)} \\
\alpha_{13}^{(3)} &= h_{12}^{(3)} + \varSigma_1(e_{12}^{(3)}) + \mathrm{Ch}(e_{12}^{(3)}, f_{12}^{(3)}, g_{12}^{(3)}) + k_{13} + w_{13}^{(3)} \\
&= h_{12}^{(0)} + \varSigma_1(e_{12}^{(1)} + 1_{41}) + \mathrm{Ch}(e_{12}^{(1)} + 1_{41}, f_{12}^{(0)}, g_{12}^{(0)}) + k_{13} + w_{13}^{(1)} \\
e_{12}^{(1)} &= e_{12}^{(0)} + 1_{63} \\
w_{13}^{(1)} &= w_{13}^{(0)} + (-\varSigma_1(1_{63}))
\end{aligned}$$

Since $f_{12}^{(0)}[63] = g_{12}^{(0)}[63]$, we have

$$\mathrm{Ch}(e_{12}^{(0)} + 1_{41}, f_{12}^{(0)}, g_{12}^{(0)}) = \mathrm{Ch}(e_{12}^{(1)} + 1_{41}, f_{12}^{(0)}, g_{12}^{(0)}).$$

Hence, $\varDelta\alpha_{13}^{\langle 2\rangle} = \mathsf{0}$ if and only if the following equation holds.

$$\varSigma_1(e_{12}^{(1)} + 1_{41}) = \varSigma_1(e_{12}^{(0)} + 1_{41}) + \varSigma_1(1_{63})$$

Since $\varSigma_1(1_{63}) = 1_{49,45,22}$, the above condition is equivalent to

$$\varSigma_1(e_{12}^{(0)} + 1_{41})[k] = 0 \text{ for } k = 49, 45, 22.$$

Since $\varSigma_1(1_{41}) = 1_{27,23,0}$, the probability that the difference on the 27th bit changes the 45th bit by the carry is 2^{-18}, and the probability that the other differences change the 49th bit or the 22nd bit by the carry is much small than 2^{-18}. Hence, we have

$$\begin{aligned}
\Pr\left[\varDelta\alpha_{13}^{\langle 2\rangle} = \mathsf{0} | \varDelta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}\right] &= \Pr\left[\varSigma_1(e_{12}^{(0)} + 1_{41})[k] = 0 \text{ for } k = 49, 45, 22\right] \\
&> 1 - 2 \cdot 2^{-18} = 1 - 2^{-17}.
\end{aligned}$$

Actually, we chose 1_{41} as the difference to make this probability large. This inequality is in agreement with the experimental value of $\Pr\left[\Delta\alpha_{13}^{\langle 2\rangle} = \mathsf{0} | \Delta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}\right]$, which is $2^{-0.000006}$. As a result, we obtain

$$\begin{aligned}\Pr\left[\Delta\alpha_{13}^{\langle\ell\rangle} = \mathsf{0} \text{ for } \ell = 0,2\right] &= \Pr\left[\Delta\alpha_{13}^{\langle 0\rangle} = \mathsf{0} \wedge \Delta\alpha_{13}^{\langle 2\rangle} = \mathsf{0}\right] \\ &= \Pr\left[\Delta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}\right] \cdot \Pr\left[\Delta\alpha_{13}^{\langle 2\rangle} = \mathsf{0} | \Delta\alpha_{13}^{\langle 0\rangle} = \mathsf{0}\right] \\ &> 2^{-4} \cdot (1 - 2^{-17}).\end{aligned}$$

A.2 Probability of $\Delta\alpha_{14}^{\langle\ell\rangle} = 0$ Under $\Delta\alpha_{13}^{\langle\ell\rangle} = 0$

Suppose that $\Delta\alpha_{13}^{\langle\ell\rangle} = \mathsf{0}$ for $\ell = 0,2$. It follows that $\Delta e_{13}^{\langle\ell\rangle} = \mathsf{0}$ for $\ell = 0,2$. Recall that

$$\begin{aligned}\alpha_{14}^{(\ell)} &= h_{13}^{(\ell)} + \Sigma_1(e_{13}^{(\ell)}) + \mathrm{Ch}(e_{13}^{(\ell)}, f_{13}^{(\ell)}, g_{13}^{(\ell)}) + k_{14} + w_{14}^{(\ell)}, \\ \alpha_{14}^{(\ell+1)} &= h_{13}^{(\ell+1)} + \Sigma_1(e_{13}^{(\ell+1)}) + \mathrm{Ch}(e_{13}^{(\ell+1)}, f_{13}^{(\ell+1)}, g_{13}^{(\ell+1)}) + k_{14} + w_{14}^{(\ell+1)}, \\ &= h_{13}^{(\ell)} + \Sigma_1(e_{13}^{(\ell)}) + \mathrm{Ch}(e_{13}^{(\ell)}, f_{13}^{(\ell)} + 1_{63}, g_{13}^{(\ell)}) + k_{14} + w_{14}^{(\ell)}.\end{aligned}$$

Hence, $\Delta\alpha_{14}^{\langle\ell\rangle} = \mathsf{0}$ if and only if $e_{13}^{(\ell)}[63] = 0$. The probability $\Pr\left[\Delta\alpha_{14}^{\langle\ell\rangle} = \mathsf{0} \text{ for } \ell = 0,2\right]$ is calculated as

$$\Pr\left[\Delta\alpha_{14}^{\langle\ell\rangle} = \mathsf{0} \text{ for } \ell = 0,2\right] = \Pr\left[e_{13}^{(0)}[63] = 0\right] \cdot \Pr\left[e_{13}^{(2)}[63] = 0 | e_{13}^{(0)}[63] = 0\right].$$

Supposing that other parts of SHA-512/22 behave as if it is a random function, we have $\Pr\left[e_{13}^{(0)}[63] = 0\right] = 2^{-1}$. Notice that the event of $e_{13}^{(0)}[63] = 0$ and that of $e_{13}^{(2)}[63] = 0$ are not independent. Recall that

$$\begin{aligned}e_{13}^{(0)} &= d_{12}^{(0)} + h_{12}^{(0)} + \Sigma_1(e_{12}^{(0)}) + \mathrm{Ch}(e_{12}^{(0)}, f_{12}^{(0)}, g_{12}^{(0)}) + k_{13} + w_{13}^{(0)}, \\ e_{13}^{(2)} &= d_{12}^{(2)} + h_{12}^{(2)} + \Sigma_1(e_{12}^{(2)}) + \mathrm{Ch}(e_{12}^{(2)}, f_{12}^{(2)}, g_{12}^{(2)}) + k_{13} + w_{13}^{(2)} \\ &= d_{12}^{(0)} + h_{12}^{(0)} + \Sigma_1(e_{12}^{(0)} + 1_{41}) + \mathrm{Ch}(e_{12}^{(0)} + 1_{41}, f_{12}^{(0)}, g_{12}^{(0)}) + k_{13} + w_{13}^{(0)}.\end{aligned}$$

Comparison $e_{13}^{(0)}$ with $e_{13}^{(2)}$ suggests that

$$\Pr\left[e_{13}^{(2)}[63] = 0 | e_{13}^{(0)}[63] = 0\right] > 1 - 2^{-19},$$

because the probability that '$+1_{41}$' in Ch() changes the 63rd bit is not larger than 2^{-20} and the probability that '$+1_{41}$' in $\Sigma_1()$ changes the 63rd bit is less than 2^{-20}. This inequality is supported by computer experiment, which showed that

$$\Pr\left[e_{13}^{(2)}[63] = 0 | e_{13}^{(0)}[63] = 0\right] = 2^{-0.000002}.$$

Therefore, we obtain

$$\begin{aligned}\Pr\left[\Delta\alpha_{14}^{\langle\ell\rangle} = 0 \text{ for } \ell = 0,2\right] &= \Pr\left[\Delta\alpha_{14}^{\langle 0\rangle} = 0 \wedge \Delta\alpha_{14}^{\langle 2\rangle} = 0\right] \\ &> 2^{-1} \cdot (1 - 2^{-19}).\end{aligned}$$

A Regression Method to Compare Network Data and Modeling Data Using Generalized Additive Model

Sooyoung Chae[1], Hosub Lee[2], Jaeik Cho[2],
Manhyun Jung[2], Jongin Lim[2], and Jongsub Moon[2]

[1] The Attached Institute of ETRI, Daejeon, Republic of Korea
sychae@ensec.re.kr
[2] Center for Information Security Technologies (CIST),
Korea University, Seoul, Republic of Korea
{leehosub,chojaeik,manhyun4,jilim,jsmoon}@korea.ac.kr

Abstract. This paper suggests a method to check whether the real network dataset and modeling dataset for real network has statistically similar characteristics. The method we adopt in this paper is a Generalized Additive Model. By using this method, we show how similar the MIT/LL Dataset and the KDD CUP 99' Dataset are regarding their characteristics. It provided reasonable outcome for us to confirm that MIT/LL Dataset and KDD Cup Dataset are not statistically similar.

Keywords: Statistical data analysis, Data set evaluation, Network data comparing.

1 Introduction

There are many modeling network datasets that are being used for researches on performance evaluation of Intrusion Detection and algorithms for Intrusion Detection Systems. Some of those modeling network dataset research institutes are located at MITRE [1], University of California at Davis [2], MIT Lincoln Laboratory (MIT/LL) [3,4,5,6], Air Force Research Laboratory [7], Neohapsis [8]. This kind of network dataset is used for performance evaluation of Intrusion Detection systems; thus, a comparison with the real network dataset is desirable. However, there is no known modeling network dataset that really reflects the characteristics of the real network data. This is because there is no known method to compare the similarities between the two network data groups.

In this paper, we suggest a method that effectively compares and contrasts the real network data and the modeling dataset. The method suggested is the Generalized Additive Model (GAM) [9,10,11,12,13,14,15], one of the regression analysis models, which compares a distribution of specified features. While researching with GAM, we compared the network dataset that was released by MIT/LL and the dataset of KDD CUP 99' [16,17].

This paper is organized as follows. In Chapter 2, we explain the Generalized Additive Model that we suggested in comparing network data and dataset. In

Chapter 3, we explain the data we used in our experiment; we discuss and analyze the preparation, the procedures and the test results. Finally, In Chapter 4, we draw a conclusion from the results of the experiment and discuss the possible research that could be done afterwards.

2 Generalized Additive Models(GAM)

In this section, we explain many characteristics of GAM and explain how to derive GAM.

2.1 Regression Analysis

Regression analysis is a statistical model that shows a relation between y, the dependent variable, and, x, the independent variable. This regression analysis is commonly used for these two purposes [18].

- An estimation model that is based on the information collected in the past.
- To find out how much the dependent variable y and the independent variable x are related.

This kind of regression analysis allows analysis of many multivariate variables such as the binary responses and count responses. For example, we use logistic regression analysis for binary response and Poisson regression for count response [18].

Regression analysis model has the following equation (1).

$$y = f(x_1, \cdots, x_p) + \varepsilon,\ \varepsilon \sim N(0, \sigma^2) \tag{1}$$

The equation (1) is the regression analysis of continuous response variable y and explanatory variable $X(x_1, \cdots, x_p)$. ε represents error, and it is the random variable that fits the normal distribution [10]. Function $f()$ represents the regression function, and it is linear or non-linear.

Also, the linear regression function with multi-variables are the following equation (2) [10].

$$f(x_1, \cdots, x_p) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \tag{2}$$

The equation (3) below was derived from equation (1) using the Least Square Method(LSM) to find the most satisfying regression equation [10].

$$\varepsilon^2 = \{y - f(x_1, \cdots, x_p)\}^2 \tag{3}$$

LSM is a way to find parameters which make ε^2 the smallest value.

The regression analysis model that was mentioned above is not appropriate to analyze non-linear network data because the model is linear regression. In other words, linear regression function like the equation (2) does not allow non-linear relations between x, the explanatory variable, and y, the response variable. This is because the slope of the function rapidly changes when the domain of ε is big. For this reason, the model could be different than what is expected due to the change in slope.

In this paper, that data that was used for the research is produced by many devices and users. Therefore, the network data is mingled complicating the results in a function with complicated distribution. Therefore, we compare two datasets to use the non-linear regression model, GAM, which fits many non-linear functions.

2.2 Generalized Additive Model

General Explanation of GAM. GAM is a method that was suggested by Hastie and Tibshirrani in 1990. It is a non-linear model that is explained by the summation of smooth functions of covariance [11]. This model is an additive non-linear model which follows the response variable y of the exponential family distribution, such as Normal distribution, Binomial distribution, Gamma distribution, Poisson distribution, Inverse Gaussian distribution, etc. The model is as equation (4). GAM is different from linear models in that it only allows X, the explanatory variable, to be applied in a closed domain. In other words, it only allows X in a closed domain in order to allow for a better fitting in closed intervals.

The following equation represents the GAM model [11].

$$\eta(x) = s_0 + \sum_{j=1}^{p} s_j(x_j),\ Where\ \eta = g(\mu),\ \mu = E(Y_i)$$
$$and\ Y_i \sim some\ exponential\ family\ distribution \qquad (4)$$

In the equation (4) above, Y_i is the set of response variable and x_j is the set of explanatory variable. s_j is a scatterplot smooth function.

How to get parameters of the GAM. The equation (5) represents penalized model, which solves parameters used for equation (4). Equation (5) uses similar method as the LSM. In other words, the purpose of equation (5) is to acquire $\eta(x)$ of equation (4). If we solve the ε with LSM, we can find the parameters of GAM. This function calculates and minimizes the difference between the observed response value and estimated response value like the first term in equation (5). However, we prevent generating a function with big changes in the slope by using the second term as penalty. In other words, we use the penalty term to minimize entropy because slopes with big changes are unstable [14].

$$\varepsilon = \sum_{i=1}^{n}(y_i - \eta(x_i))^2 + \lambda \int_a^b (f''(t))^2 dt,$$
$$\lambda > 0,\ a < x_1 < \cdots < x_n < b \qquad (5)$$

Penalized model makes the distribution smooth using parameter λ. As λ increases, the overall distribution in GAM is made smooth and becomes a straight line. In contrary, if λ approaches 0, the function in the graph develops a lot of slopes [9,10,15].

3 Experiment and Results

In this section, we compare the distribution of two network datasets by using GAM. This provides evidence to whether the dataset modeled from the network data reflects the original network's characteristics. In the experiment, we used the network data of MIT/LL and KDD CUP 99'. The procedures are as the following.

1. Extracting the frequency of protocol from two network dataset. The frequency is per minute.
2. Normalizing the frequency of protocols using scale function.
3. Deriving GAM from the two normalized network datasets.
4. Verifying whether the two datasets are reflecting the common protocols using GAM.

3.1 The Characteristic of Data

MIT/LL Data. MIT/LL's research of dataset started in 1998 with the support of Defense Advanced Research Project Agency (DARPA). The purpose of this research was to evaluate the intrusion detection [4].

In order to generate a dataset, the experiment for collecting network data is crucial. However, in consideration of the protection of private information and the release of classified information, the publicized collection of network data is impossible. For that reason, MIT/LL with the support of DARPA, collected the network data packet of the Air Force base [6].

For this experiment, MIT/LL analyzed the network environment of the Air Force base. In analyzing the environment, they analyzed the types of operating systems used by hosts, the number of hosts using each operating system and other internal factors etc. They did not consider any external factors. After analyzing the environment, they collected the real network data. In collecting the real network data, they had two major purposes. They collected data sent to external network from an internal network and they collected the data received by an internal network from the external network. For military security reasons, the MIT/LL erased and reformed each network packet's datagram [5]. Also, MIT/LL researched on a closed network to collect intrusion network data. They took the attack data from the network and analyzed the attack packets and experimented the intrusion another time. In other words, they took the regenerated network data, divided it into the normal and attack data. Then they retrieved the normal data for experiment network and analyzed the attack data. This allowed them to use automata which automatically attacked, simulating the real attack. They created hundreds of host when experimenting [3].

In 1998, the network packet data included the normal network data from the UNIX system, Stealth attack and 38 other kinds of external intrusion [3].

KDD CUP 99' data set. In 1998, MIT/LL collected network data of an Air Force base, deleted the datagram, and reproduced it. Later MIT/LL's data became a model for KDD CUP 99' data set [17].

Table 1. Partial features in KDD CUP 99' data set [16]

Feature	Description
Duration	Length (number of seconds) of the connection
Protocol	Type of the protocol, e.g. tcp, udp, etc.
Service	Network service on the destination, e.g, http, telnet, etc.
SRC_Byte	Number of data bytes from source to destination
DST_Byte	Number of data bytes from destination to source
Flag	Normal or error status of the connection
Land	If connection is from/to the same host/port; 0 otherwise
Wrong Fragment	Number of "WRONG" fragment
Urgent	Number of urgent packet

Each data in the KDD CUP 99' data was originated from a set of MIT/LL data. A sequence of the KDD CUP 99' data was produced with 41 features over a certain amount of time from the original MIT/LL dataset. They derived the MIT/LL data over a 2 second interval and used the most frequently appearing feature to represent the sequence of KDD CUP 99' data.

Table 1 shows feature of each TCP connection from KDD CUP 99' data set.

3.2 Experimental Method

Network Packet Data consists many characteristics such as protocol, Time to Live (TTL), sequence number, IP address and port number [19,20,21]. In this paper, we used protocols as a feature because protocols exist in every network layer, regardless of environment.

We used the data produced from MIT/LL and the data from KDD CUP 99', which was modeled on MIT/LL's data for experiment. The common protocols found in the two data are ARP, ICMP, TCP and UDP.

Making Frequency tables. In order to compare the frequency of the protocols between the two data (MIT/LL and KDD CUP 99'), we divided the data into one minute blocks. Within each block, we counted each protocol that appeared. Table 2 shows only a part of the result counting the number of protocols from MIT/LL's network packet data in each block.

Table 2. The frequency of protocols from MIT/LL's data network

Block sequence	ARP	ICMP	TCP	UDP
1	6	14	1152	77
2	4	0	712	50
3	16	0	312	43
⋮	⋮	⋮	⋮	⋮

Table 3. Number of protocols found in each block of KDD CUP data set

Block sequence	ARP	ICMP	TCP	UDP
1	0	0	30	0
2	0	0	30	0
3	0	0	30	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2 illustrates that ARP appeared 6 times, ICMP appeared 14 times, TCP appeared 1152 times and UDP appeared 77 times for first one minute (block 1).

With the same method, we counted the number of the protocols in KDD CUP 99' data set, which is shown in Table 3.

As mentioned earlier, KDD CUP 99' data set was generated by dividing MIT/LL's network packet data into 2 second intervals. There are 30 sequences of data in one block of experimental data for the KDD CUP 99'.

The most frequent protocol obtained from the blocks of KDD CUP 99' data set was TCP protocol; ARP did not appear in any of the blocks.

There was a big gap between the numbers of protocols of MIT/LL and those of KDD CUP 99'. Thus, we need to normalize the two tables. To do so, we used equation (6) (shown below) and scaled the frequency tables. Scaling method is as follows.

Normalizing the frequency tables. Table 4 shows the frequency table from Table 2 and Table 3. The scaling process for each d_{ij} is shown as the following equation (6) [14].

$$d_{ij}\prime = d_{ij} - min(d_{i1}, \cdots, d_{in})$$
$$d_{ij}\prime\prime = \frac{d_{ij}\prime}{max(d_{i1}\prime, \cdots, d_{in}\prime)}, i = 1, \cdots, p, j = 1, \cdots, n \quad (6)$$

p is the number of features, n is the number of blocks and i is the index of each feature and j is the index of each block. The function $min()$ selects the minimum value from the elements and $max()$ selects the maximum value from the elements. The calculated $d_{ij}\prime\prime$ has a maximum 1 and a minimum 0 within each feature, that is, a normalized value within each feature.

When two tables (frequency tables) are converted using this method, we produce tables that are normalized within each feature.

Table 4. Frequency of each feature according to the block sequence

Block sequence	D_1	D_2	$\cdots$	D_n
1	d_{11}	d_{21}	$\ldots$	d_{p1}
2	d_{12}	d_{22}	$\ldots$	d_{p2}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
n	d_{1n}	d_{2n}	$\ldots$	d_{pn}

Table 5. Normalized table from Table 2 assuming with only first 3 blocks

Block sequence	ARP	ICMP	TCP	UDP
1	0.167	1	1	1
2	0	0	0.476	0.206
3	1	0	0	0

Table 5 is a result of normalizing Table 2 assuming that there are only 3 blocks.

Fitting Generalized Additive Models. GAM uses the normalized table converted with equation (6). In this model, the independent variable is the block number and the dependent variable is the normalized value of each feature. Since the block sequence is represented per minute, response variable, y, follows Gamma distribution [22].

$$\begin{aligned} g(\mu) &= s(protocol_i) \\ protocol_i &: arp_j, icmp_j, tcp_j, udp_j \\ Link &: Inverse \end{aligned} \tag{7}$$

Parameters in (7) are used to calculate the GAM method. The link function adopts the "inverse" transformation because the response variable y is assumed to follow the Gamma distribution.

3.3 The Results of Experiments

Fig. 1., Fig. 2., Fig. 3., Fig. 4. are results after fitting the protocols of MIT/LL and KDD CUP 99' data sets for each protocol. The x-axis represents the normalized number of a protocols and the y-axis represents the value of the smooth function on each figure. The solid line shown on each graph represents the $s(protocol)$

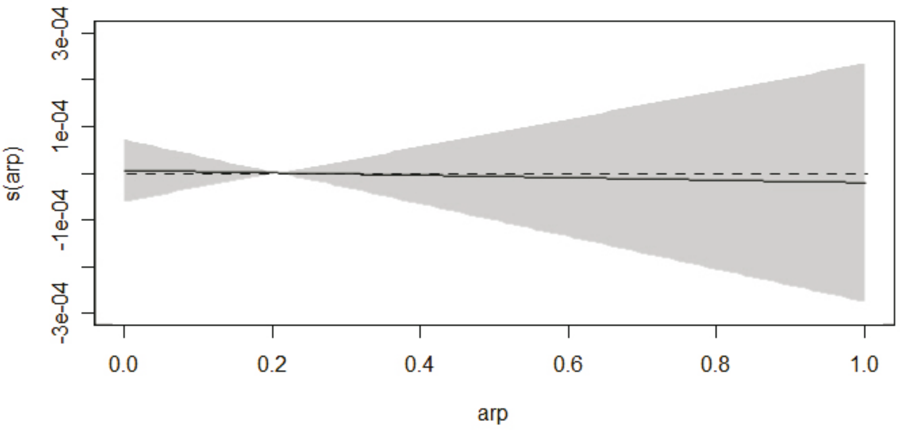


Fig. 1. $s(ARP)$ with ARP

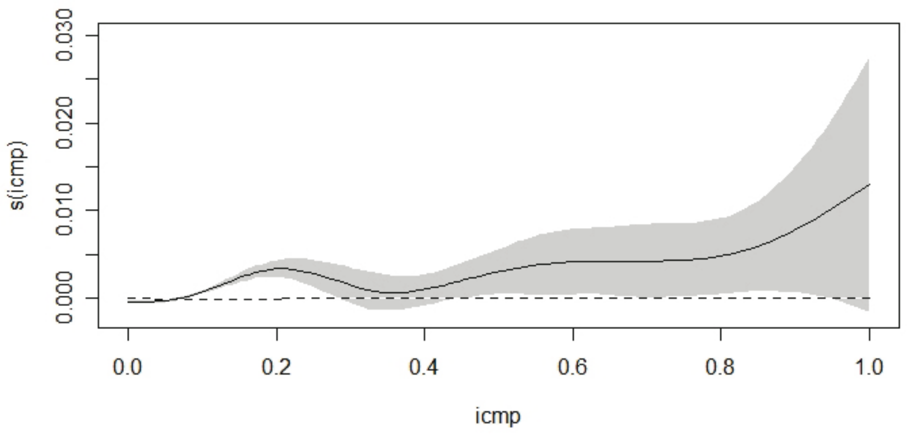


Fig. 2. $s(ICMP)$ with ICMP

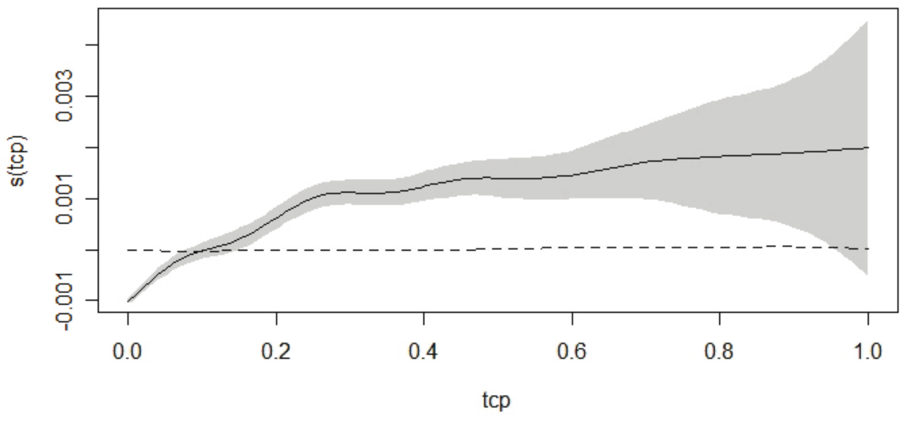


Fig. 3. $s(TCP)$ with TCP

value of the MIT/LL. The dotted line in all the graphs is $s(protocol)$ value of the KDD CUP 99'. The gray area is confidence interval of MIT/LL data. If a dotted line, fitted KDD CUP 99' data set, is inside the confidence interval of the MIT/LL network data, then we can say that the characteristics of the two data are statistically identical. In this case, we say that the KDD CUP 99' data reflects the characteristics of the MIT/LL data; therefore, we can use KDD CUP 99 instead of MIT/LL.

Fig. 1. shows the result of the GAM fitting regarding ARP protocol. In the case of KDD CUP 99' data set, the number of ARP protocol of the entire data is 0. So, the dotted line is horizontal over the x-axis.

Fig. 2. illustrates the results of GAM fitting with ICMP protocol. It shows that the graph of KDD CUP 99 is deviated from the confidence interval in an overall range.

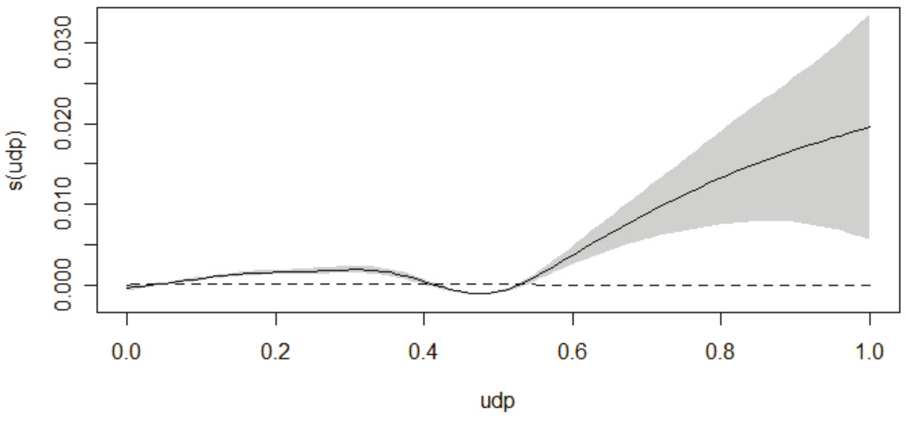


Fig. 4. $s(UDP)$ with UDP

Table 6. Fitting Gam for MIT/LL and Data cup'99 (10^{-3})

	0.0	0.2	0.4	0.6	0.8	1.0
GAM fitting for MIT/LL	-1.03	0.618	1.239	1.463	1.823	1.994
Upper Confidence	-0.95	0.833	1.525	1.98	2.927	4.496
Lower Confidence	-1.1	0.403	0.952	0.986	0.719	-0.51
GAM fitting for KDD CUP'99	-0.0112	-0.0084	-0.00968	0.0404	0.0462	0.0162

Also, Fig. 3. and Fig. 4. show $s(TCP)$ and $s(UDP)$ respectively. It is noticeable from the graphs that the dotted line for KDD CUP 99' data set diverges a lot from the confidence interval of the MIT/LL network data.

Table 6 shows numeric value of partial results from fitting TCP protocol with GAM. Fig. 3. represents the content of the Table 6 with a graph. The result shows that the characteristics of TCP for KDD CUP 99 data are not identical with that of MIT/LL.

In addition, we also checked the characteristics of both ICMP and UDP protocol also deviated from that of MIT/LL.

4 Conclusions

In this paper, we suggest a generalized adaptive model to compare the characteristics of two network data sets. Using this model, we can criticize whether the two data sets are recognized as identical groups or not in statistical view. This model may apply to another application. That is, this model can be applied to compare a simulated data with a real data such as a simulated Denial of Service (DoS) attack using real attack data [23,24,25].

The experimental data used in this paper is a frequency of protocols which resides in both MIT/LL data and KDD CUP 99' data. The results of the experiment show the distribution graph of each protocol of the MIT/LL with a confidence interval. Also the distribution graph of KDD CUP 99' data is overlapped with the corresponding graph. Thus, we can easily verify if protocols of the two data sets are identical or not. The experiment concludes that two experimental data groups are not identical.

In the future, we need to research different fitting models depending on each feature and not on GAM, because there are so many characteristics associated with the network data. We need to come up with more appropriate models depending on each feature. Therefore, we need to explore more appropriate research methods for each feature.

Acknowledgments. This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (KRF-2006-521-D00461).

References

1. Aguirre, S.J., Hill, W.H.: Intrusion Detection Fly-Off: Implications for the United States Navy, MITRE Technical Report 97W0000096 (1997)
2. Puketza, N., Chung, M., Olsson, R.A., Mukherjee, B.: A software platform for testing intrusion detection systems. IEEE Software 14, 43–51 (1997)
3. Haines, J.W., Laboratory, L.: 1999 DARPA intrusion detection evaluation: design and procedures, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Mass (2001)
4. Lippmann, R., Haines, J., Fried, D.J., Korba, J., Das, K.: Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. In: Debar, H., Mé, L., Wu, S.F. (eds.) RAID 2000. LNCS, vol. 1907, pp. 162–182. Springer, Heidelberg (2000)
5. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks 34, 579–595 (2000)
6. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K.: Evaluating intrusion detection systems: the 1998 DARPA off-lineintrusion detection evaluation. In: DARPA Information Survivability Conference and Exposition 2 (2000)
7. Durst, R., Champion, T., Witten, B., Miller, E., Spagnuolo, L.: Testing and evaluating computer intrusion detection systems. Communications of the ACM 42, 53–61 (1999)
8. Mueller, P., Shipley, G.: Dragon claws its way to the top. Network Computing 20, 45–67 (2001)
9. Wood, S.N.: The mgcv Package (2007), `http://cran.r-project.org/doc/packages/mgcv.pdf`
10. Faraway, J.J.: Linear Models With R. CRC Press, Boca Raton (2005)
11. Hastie, T., Tibshirani, R.: Generalized Additive Models. Statistical Science 1, 297–310 (1986)
12. Hastie, T., Tibshirani, R.: Generalized Additive Models: Some Applications. Journal of the American Statistical Association 82, 371–386 (1987)
13. Hastie, T., Tibshirani, R.: Generalized additive models. Chapman and Hall/CRC, Boca Raton (1990)

14. Wood, S.N.: Generalized additive models: an introduction with R. Chapman and Hall/CRC, Boca Raton (2006)
15. Xiang, D.: Fitting Generalized Additive Models with the GAM Procedure, SAS Institute Paper P 256 (2001)
16. Stolfo, S.: KDD-CUP-99 Task Description, http://kdd.ics.uci.edu/databases/kddcup99/task.html
17. Elkan, C.: Results of the KDD 1999 classifier learning. ACM SIGKDD Explorations Newsletter 1, 63–64 (2000)
18. Chatterjee, S., Hadi, A.S.: Regression Analysis by Example, 4th edn. Wiley-Interscience, Hoboken (2006)
19. Tanenbaum, A.S.: Computer Networks. Prentice Hall PTR, Englewood Cliffs (2002)
20. Stevens, W.R.: TCP/IP Illustrated, vol. I. Addison-Wesley Publishing Company, Reading (1995)
21. Kurose, J.F., Ross, K.W.: Computer networking: a top-down approach featuring the Internet. Pearson/Addison Wesley, Boston (2005)
22. Weisstein, E.W.: Gamma Distribution,WolframMathWorld (2005), http://mathworld.wolfram.com/GammaDistribution.html
23. Jin, S., Yeung, D.S.: A covariance analysis model for DDoS attack detection. In: 2004 IEEE International Conference on Communications, vol. 4, pp. 1882–1886 (2004)
24. Seo, J., Lee, C., Shon, T., Moon, J.: SVM approach with CTNT to detect DDoS attacks in grid computing. In: Zhuge, H., Fox, G.C. (eds.) GCC 2005. LNCS, vol. 3795, pp. 59–70. Springer, Heidelberg (2005)
25. Chen, Z., Gao, L., Kwiat, K.: Modeling the spread of active worms. In: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1890–1900. IEEE, Los Alamitos (2003)

A Visualization Technique for Installation Evidences Containing Malicious Executable Files Using Machine Language Sequence

Jun-Hyung Park[1], Minsoo Kim[2], and Bong-Nam Noh[3]

[1] System Security Research Center, Chonnam National University, Korea
werther@lsrc.jnu.ac.kr
[2] Dept. of Information Security, Mokpo National University, Korea
phoenix@mokpo.ac.kr
[3] School of Electronics and Computer Engineering,
Chonnam National University, Korea
bbong@jnu.ac.kr

Abstract. In the modern society the majority of information is stored and preserved on the digitalized storage medium. By the way, it is difficult to recognize that there are any adding, deleting, or changing of the records in the digitalized storage medium. In this paper, we suggest an evidence detection technique of malicious executable file installation on computer system using visualization of similarity between machine language sequences. Also suggested method can not only detect original malwares but also mutants of them. Besides, our method can help to reassemble the data blocks containing the fragments of the malicious file back into their proper sequences for securing legal evidences.

Keywords: Digital Forensic, Anti-Forensic, Machine language, Similarity, Malware, Malicious executables, opcode visualization.

1 Introduction

Computer system uses filesystems for storing and organizing files and the data on the data storage device such as hard disk. Also filesystem involves maintaining the physical location of the files on the data storage devices. By the way we cannot find the location of data blocks of files and understand the meaning of them, if the file is deleted or the metadata of file in the filesystem get damaged.

In digital forensic investigator's view, such kinds of problems are crucial. Though forensic investigators try to reconstruct the events of happened cyber crime and report to law enforcement agency, it is difficult to find traces and collect evidences following 5w1h (when, where, who, what, why, and how). For instance, the appearance of anti-forensic technology makes detecting malicious executables harder [1][2].

In this paper, we explain a detection technique for installation evidences of malicious executable files from the filesystem of computer system by comparing sequences of machine language instructions (especially, opcode sequence) [3].

The suggested technique can help us to detect data blocks containing similar opcode(operation code) sequence to known malicious executable files from the entire filesystem without the metadata of files. Additionally we can reassemble the data blocks containing the fragments of a malicious file back into their proper sequences automatically for securing legal enforcement [4].

The technique we propose can be used to depict to the jury in the courts for explaining the possibility by using visual graphs of similarity as evidences. Also we believe this technique can be used to protect copyright for program developed by individuals or industry.

The organization of this paper is as follow. In section 2, we discuss problems of present digital forensic investigation and the necessity to analyze executable files statically. In section 3, we explain our method to extract machine language sequence from filesystem for estimating similarities between sequences. Section 4 contains explanation of similarity graph and the experimental results. Section 5 concludes.

2 Related Work

2.1 The Data for Forensic Analysis

From the point of view of forensic investigators, we classified the data on the filesystem as above figure depicting. At first all data on the computer system must be written to text format or binary format. In the case of text file, forensic investigator can access directly without any specific applications because it follows American Standard Code for Information Interchange (ASCII) format generally and use keyword searching for strings or pattern matching for information such as internet protocol address or log data. Although text file is human-readable, there are some files we need to know the entire information for understanding the meaning of the data. That is why we classified text file into 2 groups as below.

The binary format data is unreadable by human and it can be an executable file itself or data to be used by executable programs. Therefore, binary data can

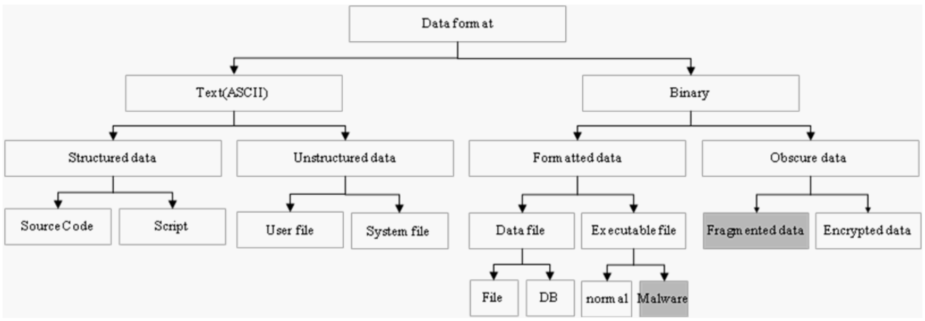


Fig. 1. Data classification on the filesystem for analysis the meaning of the data

Table 1. The groups and files of text file format

Group	File
Structured data	Script, source code for computer programs, Webpage, XML, markup language and other web standards-based file formats
Unstructed data	User data file, System data file(setup, configuration), Log files

Table 2. The groups and files of binary data

Group	File
Structured data	Archive and compressed, Computer-aided, Database, Document, Font file, Graphics, Object code, Executable files, Shared and dynamically-linked libraries, Presentation, Sound and music, Video, Financial Records
Unstructed data	Fragmented data, Encrypted data

be understood and executed exactly by operating system or application programs when it is written in appropriate format. That is why most binary data files (except some binary data files) have a file header to state their file format and tables to depict the structure of them. By the way, forensic investigators cannot understand the meaning of binary data if it is encrypted or fragmented. Our classification of binary data is as table 2.

Regardless of groups in text data, they are readable and can be analyzed easily. In the case of binary data, on the other hand, the most difficult problem to solve is that we must be able to understand the meaning of the data. That is why we need various tools to support all kinds of binary data files for the forensic investigation. Nevertheless, nowadays the capacity of data storage device is still getting bigger, it is impossible for human to analyze them if we must use appropriate applications for each of formats.

2.2 The Detection Mechanisms for Malicious Executable Files

Malicious code (malware) is any software developed for infiltrating or damaging a computer system without owner's consent [5]. Though the problem of malicious code has a long history, the detrimental effect by malware on society is still getting bigger. Furthermore, the protection or the evasion against malware is even more difficult than past by the creation and the spread of mutants of original malware through internet environment [6][7].

Moreover, the executable file that forensic investigators need to detect does not imply traditional malicious files such as virus, and worm. Detection executable files related to cyber crime cannot be solved by present malware detection technique using signatures of file, and mass storage device consumes much time for analyzing and detecting evidences [1].

Thereupon National Institute of Standards and Technology (NIST) started a project for developing National Software Reference Library (NSRL), is supported by United States Department of Justice's National Institute of Justice [8][9][10]. The NSRL project develops the Reference Data Set (RDS) which is the set of hash values of well-known files in the computer system. It is based on idea that most file on the computer systems is not related to cyber crime, and those files are used commonly and widely.

Forensic investigators can reduce the number of files to analyze by comparing the hash values of the files on the computer system with the values of the RDS. Though RDS is an effective method for forensic analysis, it is not sufficient to detect files related to cyber crime. The analyzing method using hash value or signature likes RDS has a limitation to detect specific file because hash value changes totally by even the difference of 1byte data of a file. That is why we need the detection technique for malicious executable files including mutants of them by using the sequential structure of file.

3 Executable Files and Machine Language Instruction

In this section we explain the method to abstract machine language instructions from a filesystem for detecting malicious executable files and the fragments of them. In real digital forensic environment, suggested technique can be used for only remaining data after the inspection with RDS of NSRL project. Also we can reduce time remarkably for investigation when there is some speculation about malicious code based on symptoms of victim system.

Computer system uses filesystems for storing and organizing files and the data on the data storage device such as hard disk. Also filesystem involves maintaining the physical location of the files on the data storage devices. By the way forensic investigators cannot find the location of data blocks of files and understand the meaning of them, if the file is deleted or the metadata of file in the filesystem get damaged [3]. For those cases, our method can abstract opcodes sequence consisting machine language instructions from the entire filesystem and data blocks record file fragments. After abstracting opcode sequences, we estimate the similarity between opcode sequences of abstracted sequences and suspicious malicious executable files. In section 3.1, we explain the method to abstract opcode sequence from the target data and the difference the frequencies of opcodes between executable data and others.

3.1 Opcode Abstraction

The executable file includes information of header, machine language instructions sequence for proper execution. The header section explains the file format and some tables for structure of the file. Opcode is the portion of a machine language instruction that specifies the operation to be performed. Though an instruction

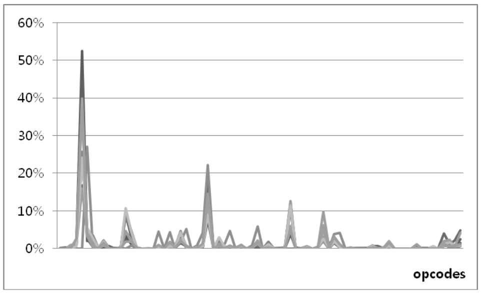


Fig. 2. The opcode frequency distribution of executable files

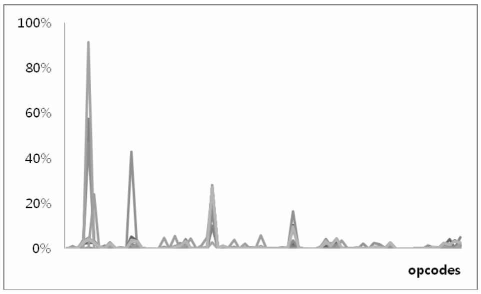


Fig. 3. The opcode frequency distribution of executable files

has one or more operands, they are changeable always, because operands represent the virtual address of variable.

That is why we used the order of opcodes only for calculating the similarity between executable files. The executable files can be estimated the degree of similarity with other executable files by using cmp section as a basic unit [3]. Below pictures show the frequencies distribution of abstracted opcodes from executable files and other formats.

Figure 2 depicts the frequencies of opcodes in executable files, and figure 3 is for other formats except executables. The file formats we tested for detect the frequencies of opcodes from other formats are multimedia file formats (jps, bmp, mp3, mp4, avi), MS office file formats (excel, powerpoint, MS-word), and system files (logs, configuration). By the way, opcodes abstracted from other formats are false positives because file formats except executable files do not contain machine codes. For this reason, we need to select opcodes detected from executable files in which not from nonexecutable files.

3.2 The Quantification of the Information Contained in Opcodes

It takes long time for calculation the similarity between files when all opcodes are used for estimation. That is why we need to select some opcodes for reducing

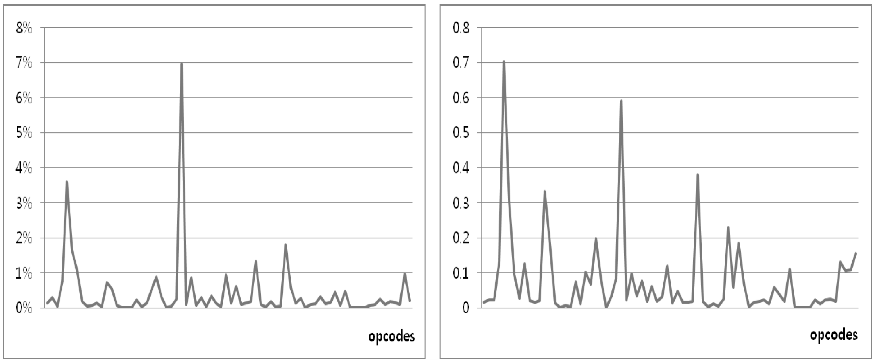


Fig. 4. The result graph of Chi-square distribution test and information entropy

complexity problem. Also there are several principles for the performance. At first we need to reduce the number of opcodes for calculating similarity of course. Second opcodes are detected occasionally must be eliminated from our abstraction for reducing calculation time. Third we must not eliminate opcodes, which are abstracted many from executable files and not from other formats.

Left side graph of below figure 4 shows the chi-square distance distributions of opcodes from executable files and others. According to our test results, several opcodes (add, jg, mov, and, pop) are abstracted frequently from executable files and nonexecutable files, and those opcodes are 1byte instructions for granted. Also those opcodes are detected more from other file formats than executable files.

Right side graph in figure 4 is the result of estimation of information entropy. The shannon entropy or information entropy is a measure of the uncertainty associated with a random variable. We estimated the quantification of the information to classification of file formats. In our test result, we decided the information entropy is more useful to select opcodes among entire opcodes.

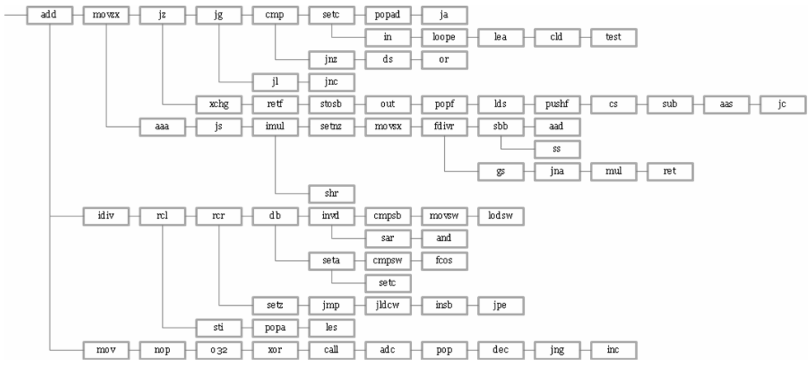


Fig. 5. The result graph of Chi-square distribution test and information entropy

3.3 Decision Tree for Selection of Opcodes by C4.5

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier. It uses the fact that each attribute of the data can be used to make a decision that splits the data into smaller subsets.

For the test, we used 1 Giga bytes of executable files and the same amount of other files. We selected 75 opcodes from entire opcodes by C4.5. By using selected opcodes only, the time for calculation decreased less than 35.

4 Acquisition Evidences of Malware Installation

In this section, we explain the similarity graph briefly, and then discuss about the changes of similarity graph according to the mutants of executable file. At the conclusion, we show the method for detect the evidences of malware installation from filesystem.

4.1 The Similarity Graph

The graphs of figure 6 are examples of similarity graphs. The vertical axis is a cmp sections sequence of the source file, such as malware and the horizontal axis is for the target file, such as filesystem. Also the brightness of each dot in the picture is showing the degree of the similarity of each cmp sections (between each of cmp sections of source file and target file) and it is getting darker, depending on the highness of the similarity.

In case of left side picture in figure 6, source file and target file are exactly same executable files. That is why we can find a diagonal line in the picture, which is the most important thing from our investigation. On the other side, the right side picture is the result of two different files, and there is not any diagonal line in the picture. Also we can find dark horizontal lines and dark vertical lines in the graphs. These lines are false positives because the lengths of the cmp sections of the dots are short.

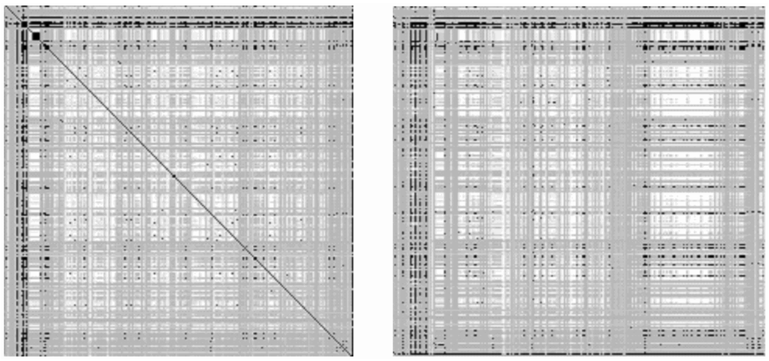


Fig. 6. The difference of similarity graphs between two files

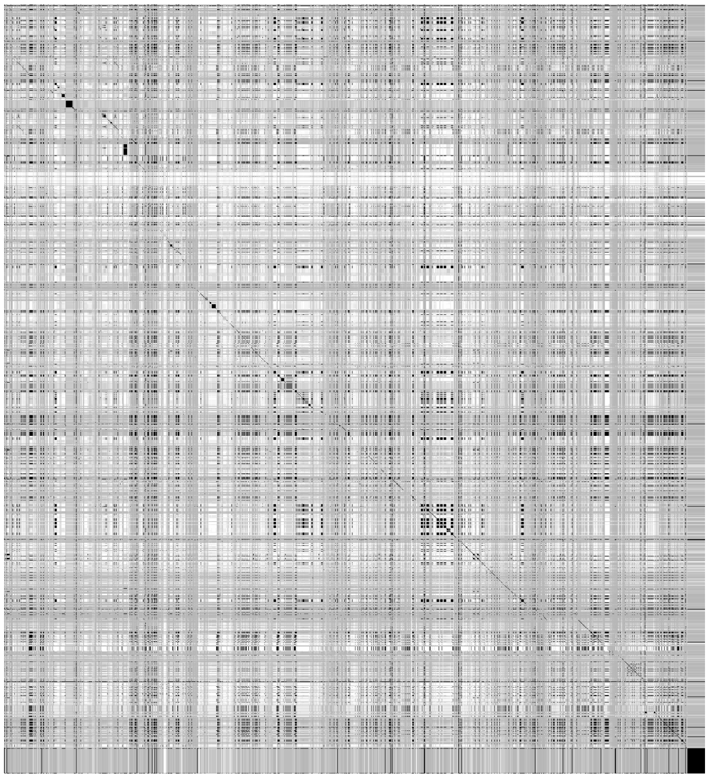


Fig. 7. The graph of similarity between version 1.6 and 1.7.02 of John the ripper

4.2 The Similarity Graphs between Diverse Versions of Executable File

We estimated similarities between several versions of malwares to know how the line is depicted in the similarity graphs. Figure 7 shows a form of similarity graph between version 1.6 and 1.7.02 of john the ripper which is a kind of well-know malware. John the ripper is a free password cracking software tool. It is one of the most popular password testing or breaking programs as it combines a number of password crackers into one package.

From this test results, we could infer above picture that there were several insertions and deletions. If there were some insertions in the original file, we can find some cutoff in the diagonal line and the line is moved to the right side. Also there must be some cutoffs and the line is moved to the down side if there were some deletion to the original file.

4.3 Reassembling Data Blocks in Order

Below figure 8 shows the test result for detection the data blocks containing malware from the entire filesystem. A malware was divided into 4 small fragments

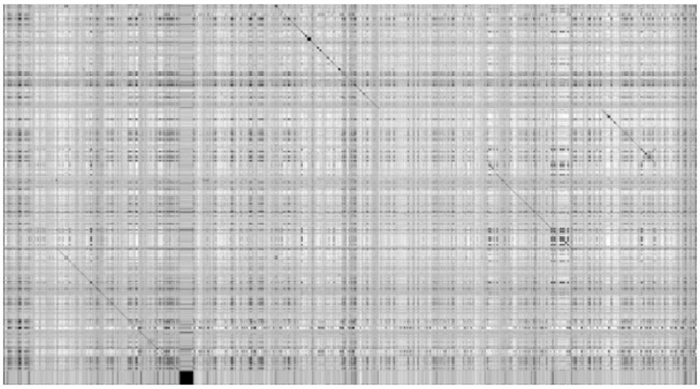


Fig. 8. The distribution of malware in the filesystem

and stored separately. By suggested detection test, we could find all of those fragments from filesystem and the result is as below picture.

In the former sections, we explained a detection technique for installation evidences of malicious executable files and mutants of them from filesystem. Below pseudo code shows the method for acquisition forensic evidences by reassembling data blocks containing malicious code in order.

```
Input : opcode sequence profiles of malware and filesystem
##### Similar data blocks detection #####
While(read next data block)
  while(read ith cmp[++i] on the data block)
    while(read jth cmp[++j] on a malware)
      If(similarity[i][j] is higher than critical value){
        write sim[i][j]
        if(there exist continuous section diagonally){
          detect current block as a candidate
          go to read next data block
        }
      }
##### Reassembling data blocks #####
While(find candidate, which is similar with former section
of malware)
  if(there are candidate for same section of malware)
    find the longest one
end.
```

At first, we detected data blocks containing similar opcode sequence with the malicious code sequence more than critical proportion from filesystem as candidate evidences. Next we reassembled the data blocks of candidate evidences by the order of malicious code. We selected the longest one with highest similarity as fragment of malware, if there are several data blocks of candidate evidences for same section of the malicious code. At last, we made as decision by comparing reassembled data blocks with the malicious code.

5 Conclusion and Future Work

In this paper, we presented a method to estimate similarity between files by visual similarity matrix. Also we suggested a detection technique for malware installation evidences from entire filesystem. Our method could be used for copyright protection by visualization the degree of similarity between two executable files. Besides, suggested technique can help to detect not only mutants of malware but also fragments of them. Lastly we can reassemble detected fragments in order for producing forensic evidences.

For future work, we will reduce the computational time for the similarity calculation. Also we have research plan for encrypted file by hooking machine language instructions for the execution. Besides we are interested in research for comparison between original file and the file embedded it.

Acknowledgments. This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD). (KRF-2005-214-D00359).

References

1. Christodorescu, M., Jha, S., Seshia, S.A., Song, D., Bryant, R.E.: Semantics-aware malware detection. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy, Oakland, CA (2005)
2. Chrisodorescu, M., Jha, S.: Static Analysis of Executables to Detect Malicious Patterns. In: Proceedings of the 12th USENIX Security Symposium (2003)
3. Park, J.-H., Kim, M., Noh, B.-N., Joshi, J.B.D.: A Similarity based Technique for Detection Malicious Executable files for Computer Forensics. In: IEEE International Conference on Information Reuse and Integration (2006)
4. Buskrik, E.V., Liu, V.T.: Digital Evidence: Challenging the Presumption of Reliability. Journal of Digital Forensic Practice 1, 19–26 (2006)
5. McGraw, G., Morrisett, G.: Attacking malicious code: report to the Infosec research council. IEEE Software 17(5), 33–41 (2000)
6. Garfinkel, S.: Anti-Forensics: Techniques, Detection and Countermeasure. In: Proceedings of the 2nd International Conference on i-Warefare and Security (ICIW), pp. 8–9 (2007)
7. McLaughlin, L.: Bot software spreads, causes new worries. IEEE Distributed Systems Online 5(6) (2004)
8. National Institute of Standards and Technology (NIST), National Software Reference Library (NSRL) Project, http://www.nsrl.nist.gov
9. Mead, S.: Unique File Identification in the National Software Reference Library. Digital Investigation (2006)
10. White, D., Ogata, M.: Identification of known files on computer systems. National Institute for Standards and Technology. Presented at American Academy of Forensic Sciences (2005)

Image-Feature Based Human Identification Protocols on Limited Display Devices*

Hassan Jameel[1], Riaz Ahmed Shaikh[1], Le Xuan Hung[1], Yuan Wei Wei[1], Syed Muhammad Khaliq-ur-rehman Raazi[1], Ngo Trong Canh[1], Sungyoung Lee[1], Heejo Lee[2], Yuseung Son[3], and Miguel Fernandes[3]

[1] Department of Computer Engineering, Kyung Hee University, 449-701 Suwon, South Korea
{hassan,riaz,lxhung,weiwei,raazi,ntcanh,sylee}@oslab.khu.ac.kr

[2] Department of Computer Science and Engineering, Korea University Anam-dong, Seongbuk-gu, Seoul 136-701, South Korea
heejo@korea.ac.kr

[3] Institute for Graphic Interfaces, Ehwa Womans University, Seoul 120-750, South Korea
{yssohn,mfernandes}@igi.re.kr

Abstract. We present variations and modifications of the image-feature based human identification protocol proposed by Jameel et al with application to user authentication on mobile devices with limited display capabilities. The protocols introduced are essentially reduced versions of the original protocol with a minor tradeoff between security and usability. However, the proposed protocols are not aimed for computation and memory restrained devices. A brief user survey highlights the usability. By employing realistic assumptions pertaining to mobile devices, we show that the protocols are secure under the conjectured difficulty of extracting the secret feature from the observation of images and their binary answers. The adversary considered is strictly passive.

1 Introduction

Secure user authentication (identification) protocols, in which a human user securely authenticates himself/herself to a remote server, are of utmost importance in today's increasingly connected world. Presently, the most prevailing form of user authentication is *password based* authentication. Alternative forms of authentication have been proposed but are not commonly deployed, generally due to their relative difficulty of use. Matsumoto [2], proposed a threat model in the user authentication scenario in which the adversary has more powers than conventional threat models for authentication. The adversary not only has passive and active access to the communication channel between the user and the server, but has also access to the computer terminal being used by the user and

* This work was supported by the IT R&D program of MIC (Ministry of Information and Communication)/IITA (Institute of Information Technology Assessment). [2005-S-604-02, Realistic Virtual Engineering Technology Development].

the alphanumerics being entered by the user. Furthermore, the user does not have access to any additional trusted hardware. Obviously, under such circumstances, traditional authentication protocols fail miserably. In accordance with the terminology used in [2], we will call user authentication protocols in which an *unassisted* human user authenticates to a remote server as *Human Identification Protocols.*

Human identification protocols secure under Matsumoto's threat model have been proposed in literature but are too impractical for humans due to high memory requirements and difficulty of use. One such protocol was proposed by Jameel et al in [1]. Their protocol is built on the diversity of things (features) an image describes. One of these features as a secret shared between the user and the server and the user has to answer '1' if a given picture contains that feature and answer '0' otherwise. On an authentication session, a series of pictures are shown to the user who constructs a binary answer string according to a predefined secret order. The protocol, as it is, cannot be used on mobile devices with small display units. In this paper, we present significant variations of the protocol in [1], so as to make it practical on mobile devices. The protocols presented are secure under the conjectured difficulty of the problem of finding the secret feature among a given set of images, which is different from the one presented in [1]. We present arguments in support of this difference and also illustrate general weaknesses and shortcomings in the original protocol as well as argue its specific inadequacies if applied to mobile devices in its original form. Our additional contribution is a comprehensive description of the threat model in [2] and a small-scale user survey based on an implementation of one of the proposed protocols which aims at demonstrating its usability.

Related Work. The first attempt at constructing a secure human identification protocol dates back to Matsumoto [2]. After this scheme was broken in [4], Matsumoto proposed another scheme in [5]. A variation of this scheme whose security is based on a mathematically hard problem was proposed by Hopper and Blum in [7], which is commonly known as the HB protocol. But HB protocol is impractical for human users as is acknowledged by the authors themselves. Other attempts, which loosely resemble this category of protocols are proposed in [4], [6] and [8]. However, the tradeoff between security and usability remains intact. Weinshall [12] proposed a slightly different protocol in which the user has to remember images instead of alphanumeric secrets. The protocol was cryptanalyzed and broken in [13]. Usage of images as memory aids is also employed in other graphical authentication schemes such as DeJa Vu [9], Passface [10], Point & Click [11] and [3]. They require little or no numerical computation whatsoever. As an example, the basic theme of [10] is to present the user a series of images, a subset of which is the set of secret images. The user is authenticated if his/her selection of secret images among the given set of images is correct. On the other hand, in [11], the user is authenticated if he/she clicks on the correct secret location in the given picture. [3] works similarly by letting the user draw the secret symbol or figure on a display device. Evidently, these purely graphical schemes are not secure against shoulder-surfing also known as "peeping" attacks [8]. An observer noting the actions of the user can know the secret readily. For a comprehensive survey of all these protocols, see [8]. Jameel et al [1] proposed a slightly different concept in which the internal properties of images are used as secrets. After a secret has been chosen, pictures which satisfy the properties are presented randomly with pictures which do not. The user has to answer the pictures according to the secret property. It is conjectured that finding out the secret property is a hard problem for adversaries. This concept is employed in this paper with new protocols different from the one proposed in [1].

2 Matsumoto's Threat Model

Matsumoto proposed a threat model in [2] and attempted to devise a human identification protocol secure in this model. Figure 1 shows this threat model pictorially. We have a human user $\mathcal{H}$ who wants to authenticate to a remote server $\mathcal{C}$. $\mathcal{H}$ is equipped with a malicious computing device. $\mathcal{H}$ and $\mathcal{C}$ are connected via an insecure communication channel. The adversary $\mathcal{A}$ enjoys the following capabilities:

P-Channel : Passive access to the communication channel.
A-Channel : Active access to the communication channel.
P-Device : Passive access to the software and hardware of the computing device.
A-Device : Active access to the software and hardware of the computing device.
P-Display : Passive observation of the visual display unit of the computing device.
P-User : Passive observation of users' inputs and actions.

Notice that the adversary has all these capabilities *during* an authentication session. We shall call this model, Full-MTM. A real world example of this model can be visualized as a user attempting to authenticate to a remote server using a computer in a public internet cafe. The computer can potentially be infected by trojan horses and key-loggers. There can be an added risk of shoulder-surfers [8] peeping at the user's input. Hidden cameras, network snoops and spoof websites can complete a real world realization of the Full-MTM.

An interesting line of research is to find human identification protocols secure in relaxed versions of Full-MTM. As an example, consider a variant of the Full-MTM in which $\mathcal{A}$ has only two capabilities: P-Display and P-User. A user entering his/her PIN number on an Automated Teller Machine is a perfect candidate for this threat model. Needless to say, the traditional *PIN number protocol* succumbs completely in the said threat model. Indeed, this protocol is only secure when the adversary has just the P-Display capability. The variation of Full-MTM considered in this paper assumes the adversary to possess all but two capabilities: A-Channel and A-Device. We call this variant P-MTM, to acknowledge that the adversary is strictly passive.

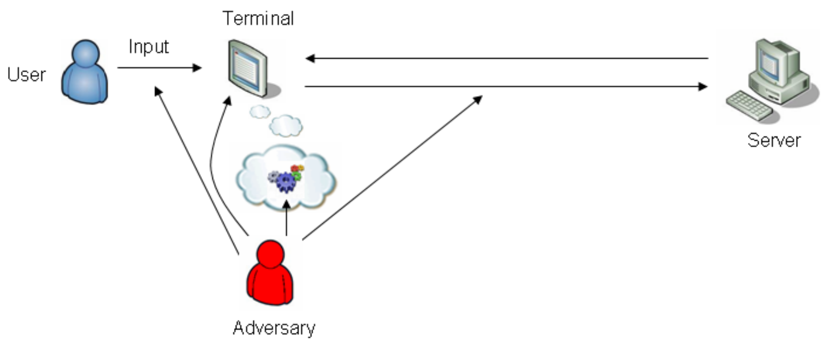


Fig. 1. Matsumoto's Threat Model

3 Human Identification Protocol Based on Images and their Features

Jameel et al [1] proposed a human identification protocol in which $\mathcal{H}$ and $\mathcal{C}$ share a common secret feature (question) which has a binary answer when applied to any image. An example secret feature can be 'an arrow'. So, by looking at the picture in Figure 1, $\mathcal{H}$ has to answer the following question: 'Does the image contain an arrow?'. During an authentication session, $\mathcal{C}$ sends a group of images such that with probability 1/2 each image satisfies the secret feature. $\mathcal{H}$'s response consists of a binary string which is constructed according to a secret permutation, also shared between $\mathcal{H}$ and $\mathcal{C}$. As an example, let the secret permutation be $**2**1345*$. The $*$'s represent the *don't care positions*. $\mathcal{C}$ sends a group of 10 images labeled from 0 to 9. $\mathcal{H}$ replies by constructing a binary string of length 10 by placing random bits in the positions of $*$'s and answering the images in the order specified by the secret permutation and in accordance with the secret question. Here, '1' means that the image satisfies the secret feature and '0' means that it does not.

The job of the adversary is to extract the secret feature. It is conjectured in [1], that since the answer string contains shuffled answers and random bits, it is "very hard" to extract the secret feature, although no concrete value for the hardness of this problem is given. The security of the protocol is reduced from the conjectured security of the said hard problem. Although the authors discuss the resiliency of the scheme against a couple of active attacks, they do not give a reductionist argument [14] against active adversaries. Therefore, we can only safely assume that the protocol in [1] is secure under P-MTM.

We notice the following weak points and deficiencies in the above mentioned protocol:

– The permutation string, if long, can be hard to remember for most of the people. Even with the length '10', as used in [2], remembering the location of the don't care positions is difficult.

- It is not known how a human user can generate random bits. Most users would just put all 0's or all 1's in the don't care positions of the answer string. This is also acknowledged by the authors in [1]. It is also not evident whether the inclusion of random bits adds to the security of the protocol or not.
- Two users with the same secret feature might differ when answering the same image. Thus, a feature/image pair is subject to $\mathcal{H}$'s interpretation which might differ from $\mathcal{C}$'s interpretation. As a result, we should allow for user error in the protocol.
- The suggested parameters in [2] are not suitable for mobile devices. Presenting 10 pictures at a time is very bothersome on the user even if he or she can *scroll* around all the images using a mobile device.
- In [2], whenever $\mathcal{C}$ presents an image to $\mathcal{H}$, it discards it from the repository and never uses it again. We can argue that if the repository of images for a given feature is large, the probability of the same picture being presented in succession is very small. We can thus get rid of this impractical requirement.
- The conjectured difficulty of extracting the secret features from images is justified due to the presence of the secret permutation. However, the underlying mechanism of shuffling the answer bits and employing random bits is very basic and can be cracked once the secret feature is found [15]. The adversary at least has the knowledge that on the average half of the images will contain the secret feature. We can similarly conjecture that the problem is hard even with out the presence of random bits and shuffled answers, although intuitively the hardness assumption will be stronger than in [1]. On the brighter side, this can allow us to construct more practical protocols on mobile devices.

In what follows, we will assume $\mathcal{H}$ to possess a mobile device and the goal is to authenticate $\mathcal{H}$ to $\mathcal{C}$ under P-MTM. We begin by rigorously defining the modified version of the conjecture presented in [1].

4 Preliminaries: Definitions and Conjectures

Denote the three parties by $\mathcal{H}$, $\mathcal{C}$ and $\mathcal{A}$ as in the previous section. $\mathcal{C}$ has a set of Images I and a set of features Q, whose contents are kept secret. Define the function: $f : Q \times I \rightarrow \{0, 1\}$ that takes as argument a $q \in Q$ and an $i \in I$ and maps the pair to 0 or 1. This function is implemented by $\mathcal{C}$. For all $q \in Q$ and $i \in I$, $b \leftarrow \mathcal{H}(q, i)$ represents $\mathcal{H}$ returning an answer bit b after taking as input q and i.

Conjecture 1. Let $q \overset{R}{\leftarrow} Q$ and $i \overset{R}{\leftarrow} I$. Then,

$$\Pr\left[b \leftarrow \mathcal{H}(q, i)\,; b = f(q, i)\right] \geq 1 - e \geq \frac{1}{2}$$

Here $0 \leq e \leq \frac{1}{2}$ is the average error probability. For the adversary $\mathcal{A}$ we have the following conjecture:

Conjecture 2. Let $q \overset{R}{\leftarrow} Q$ and $b_1 b_2 \cdots b_r \overset{R}{\leftarrow} \{0,1\}^r$. Let $i_1, \ldots, i_r$ be sampled from I, such that: $f(q, i_1) || \cdots || f(q, i_r) = b_1 \cdots b_r$. Let $b \overset{R}{\leftarrow} \{0,1\}$ and $i \leftarrow I$ such that $f(q, i) = b$ and $i \neq i_t$ for $1 \leq t \leq r$. Then,

$$\frac{1}{2} \leq \Pr[b' \leftarrow \mathcal{A}(i, (b_1, i_1), \ldots, (b_r, i_r)); b' = b] \leq \frac{1}{2} + \delta(r)$$

Here $0 \leq \delta(r) \leq \frac{1}{2}$ is a non-decreasing function of r. Notice that the adversary is shown corresponding pairs of images and their binary answers, except for the last image.

Notes. We have not defined $\mathcal{H}$ and $\mathcal{A}$ as turing machines as they could be either humans or a combination of human or machine in the case of the adversary. Conjecture 2 differs from the one presented in [2], in that we do not attribute the hardness of the problem to the extraction of the secret feature. Instead, we have related it to guessing the correct answer of an image after observing a few pairs of images and their binary answers. Furthermore, we do not involve the time consumed by $\mathcal{A}$ in conjecturing the hardness of the problem. The reason being that if $\mathcal{A}$ is a computer program, it is very hard to construct a program that will be able to solve the problem at hand. Instead, $\mathcal{A}$ needs a great deal of human assistance. We cannot say for certain that the longer the human adversary spends time in contemplating the images the better the result will be, since the time is not being used to do numerical computations.

We define the concept of human identification protocol on the basis of the definition of an identification protocol given in [7]. Define the result of the interaction between $\mathcal{H}$ and $\mathcal{C}$ with inputs x and y respectively, by $\langle \mathcal{H}(x), \mathcal{C}(y) \rangle$. For the sake of identification protocols, $\langle \mathcal{H}(x), \mathcal{C}(y) \rangle$ would be either `accept` or `reject`.

Definition 1. *A* human identification protocol *is an interactive protocol between a human $\mathcal{H}$ and a probabilistic program $\mathcal{C}$, both having auxiliary inputs, such that:*

$$\Pr[\langle \mathcal{H}(z), \mathcal{C}(z) \rangle = \texttt{accept}] \geq p$$
$$\Pr[\langle \mathcal{H}(z'), \mathcal{C}(z) \rangle = \texttt{reject}] < 1 - p$$

where $z' \neq z$.

For a human identification protocol to be useful, the probability p in Definition 1 should be high. We will say that a candidate protocol is *logically complete* if it satisfies the above definition. Of course, we need another definition for the human usability of such protocols. We take this definition from [7] again, as a reference:

Definition 2. *A human identification protocol is (α, β, τ)-*human executable*, if at least α proportion of the human population can perform the protocol computations unaided and correctly with probability greater than β and with an average time τ.*

5 Proposed Protocols

We begin with the first protocol[1] which is a direct consequence of the main idea. From here onwards, we assume the sets I and Q to be large enough so that the same image is not presented again in quick succession.

5.1 Protocol P1

Setup. $\mathcal{C}$ samples $q \overset{R}{\leftarrow} Q$. $\mathcal{C}$ and $\mathcal{H}$ share q as a secret.
Protocol.

- $\mathcal{C}$ initializes $j \leftarrow 0$.
- Repeat k times
  - $\mathcal{C}$ samples a bit $b \overset{R}{\leftarrow} \{0,1\}$. Randomly picks an image i from I such that $f(q,i) = b$. $\mathcal{C}$ sends i to $\mathcal{H}$.
  - $\mathcal{H}$ sends the response bit a to $\mathcal{C}$, where $a \leftarrow \mathcal{H}(q,i)$.
  - If $a = b$, $\mathcal{C}$ updates $j \leftarrow j+1$.
- If $j \geq s$, $\mathcal{C}$ outputs `accept`. Else $\mathcal{C}$ outputs `reject`.

Logical Completeness. We see that the protocol is logically complete if Conjectures 1 and 2 hold.

Claim 1. *If Conjectures 1 and 2 hold, Protocol* P1 *is logically complete with probability p, subject to the following conditions:*

$$p \leq \sum_{j=s}^{k} \binom{k}{j} (1-e)^j e^{k-j}$$

$$1 - p > \sum_{j=s}^{k} \binom{k}{j} \left(\frac{1}{2} + \delta(k)\right)^j \left(\frac{1}{2} + \delta(k)\right)^{k-j}$$

Proof. We see that if $\langle \mathcal{H}(q), \mathcal{C}(q) \rangle = \texttt{accept}$, then $\mathcal{H}(q,i)$ should be equal to $f(q,i)$ at least s times. From Conjecture 1, the probability that $\mathcal{H}(q,i)$ equals $f(q,i)$ is $\geq 1-e$. The sum of the probabilities of s or more successes, can be found through the binomial probability distribution. Thus for a given probability p of the event $\langle \mathcal{H}(q), \mathcal{C}(q) \rangle = \texttt{accept}$, the sum of the probabilities of s or more successes should be greater than or equal to p.
The event $\langle \mathcal{H}(q'), \mathcal{C}(q) \rangle = \texttt{accept}$ can happen with probability:

$$\sum_{j=s}^{k} \binom{k}{j} \left(\frac{1}{2} + \delta(0)\right)^j \left(\frac{1}{2} - \delta(0)\right)^{k-j}$$
$$\leq \sum_{j=s}^{k} \binom{k}{j} \left(\frac{1}{2} + \delta(k)\right)^j \left(\frac{1}{2} - \delta(k)\right)^{k-j}$$

Thus this probability has to be less than $1-p$, for Protocol P1 to be logically complete. □

Table 2 in Appendix A shows the values of s against different values of p, e and k.

[1] A similar protocol is described in an unpublished technical report [15].

5.2 Protocol P2

Denote by $\sigma[m]$, the set of all permutations from $\{0,1,\ldots,m\}$ to itself. Let $\sigma \in \sigma[m]$ be a generic permutation. $\sigma(j)$ denotes the jth element of σ. Let Grid be a data structure that holds an ordered sequence of m images. The operation '+' means the append operation when applied to Grid.

SETUP. $\mathcal{C}$ samples $q \overset{R}{\leftarrow} Q$ and $\sigma \overset{R}{\leftarrow} \sigma[m]$. $\mathcal{C}$ and $\mathcal{H}$ share q and σ as a secret.
PROTOCOL.

– $\mathcal{C}$ initializes $j \leftarrow 0$.
– Repeat k' times
  - $\mathcal{C}$ initializes Grid $\leftarrow \phi$. $\mathcal{C}$ samples $b_1 b_2 \cdots b_m \overset{R}{\leftarrow} \{0,1\}^m$.
  - For $1 \le t \le m$:
    * $\mathcal{C}$ randomly picks an image $i_{\sigma(t)}$ from I such that $f\left(q, i_{\sigma(t)}\right) = b_t$. $\mathcal{C}$ updates Grid $\leftarrow$ Grid $+\, i_{\sigma(t)}$.
  - $\mathcal{C}$ sends Grid to $\mathcal{H}$.
  - $\mathcal{H}$ initializes the answer string $a \leftarrow$ *null*.
  - For $1 \le t \le m$:
    * $\mathcal{H}$ updates $a \leftarrow a || \mathcal{H}\left(q, i_{\sigma(t)}\right)$.
  - $\mathcal{H}$ sends a to $\mathcal{C}$.
  - For $1 \le t \le m$, if $a(t) = b(t)$, $\mathcal{C}$ updates $j \leftarrow j+1$.
– If $j \ge s$, $\mathcal{C}$ outputs `accept`. Else $\mathcal{C}$ outputs `reject`.

Logical Completeness

Claim 2. *If Conjectures 1 and 2 hold, Protocol* P2 *is logically complete with probability* p*, subject to the following conditions:*

$$p \le \sum_{j=s}^{mk'} \binom{mk'}{j} (1-e)^j e^{mk'-j}$$

$$1 - p > \sum_{j=s}^{mk'} \binom{mk'}{j} \left(\frac{1}{2} + \delta(mk')\right)^j \left(\frac{1}{2} + \delta(mk')\right)^{mk'-j}$$

Proof. The first part of the proof is similar to the first part of the last claim with k replaced by mk'. For the second part, we see that the probability of the event $\langle \mathcal{H}(q'), \mathcal{C}(q) \rangle =$ `accept` would be less than or equal to the corresponding probability of the same event in Protocol P1, since now the introduction of the permutation σ adds extra error probability for someone without the knowledge of q and σ. The result follows immediately by once again replacing k by mk'. $\square$

For the allowable values of s, see Table 2 in Appendix A.

6 Security of the Protocols

Let $\mathcal{A}$ be an adversary in P-MTM. Let $\mathcal{A}'$ be another adversary with only the P-Channel capability. Informally speaking, in order to demonstrate the security of the protocols, we will first show that adversary $\mathcal{A}$ has no real advantage over adversary $\mathcal{A}'$. After that, we will attempt to reduce the security of the protocols to Conjecture 2. Once that is accomplished, we will say that the protocol is secure under P-MTM with an associated probability. Let $T\left(\mathcal{H}\left(.\right),\mathcal{C}\left(.\right)\right)$ denote the transcript of messages sent between $\mathcal{H}$ and $\mathcal{C}$ in a single run. In other words, it represents *one* challenge-response pair.

Definition 3. *We say that the adversary $\mathcal{A}$ is $\lambda\left(r\right)$-almost equivalent to the adversary $\mathcal{A}'$ for the human identification protocol $(\mathcal{H},\mathcal{C})$, if:*

$$\begin{aligned} &\Pr\left[\left\langle \mathcal{A}\left(T^r\left(\mathcal{H}\left(q\right),\mathcal{C}\left(q\right)\right)\right),\mathcal{C}\left(q\right)\right\rangle = \texttt{accept}\right] \\ \leq &\Pr\left[\left\langle \mathcal{A}'\left(T^r\left(\mathcal{H}\left(q\right),\mathcal{C}\left(q\right)\right)\right),\mathcal{C}\left(q\right)\right\rangle = \texttt{accept}\right] + \lambda\left(r\right) \end{aligned}$$

Definition 4. *A human identification protocol $(\mathcal{H},\mathcal{C})$ is (p',r) secure against the adversary $\mathcal{A}'$, if:*

$$\Pr\left[\left\langle \mathcal{A}'\left(T^r\left(\mathcal{H}\left(q\right),\mathcal{C}\left(q\right)\right)\right),\mathcal{C}\left(q\right)\right\rangle = \texttt{accept}\right] \leq p'$$

Definition 4 is taken from [7]. We can now relate the security of the protocol to Conjecture 2 in a straightforward manner. Notice that these proofs are not reductionist arguments in the formal sense, as we have not defined the adversary as being a turing machine.

Claim 3. *Protocol* P1 *is (p',r) secure under* P-MTM, *where:*

$$p' = \sum_{j=s}^{r} \binom{r}{j} \left(\frac{1}{2}+\delta\left(r\right)\right)^{j} \left(\frac{1}{2}-\delta\left(r\right)\right)^{r-j}$$

Proof. See Appendix B.1.

Claim 4. *Protocol* P2 *is (p'',r) secure under* P-MTM, *where $p'' \leq p' + \delta\left(r\right)$ and:*

$$p' = \sum_{j=s}^{r} \binom{r}{j} \left(\frac{1}{2}+\delta\left(r\right)\right)^{j} \left(\frac{1}{2}-\delta\left(r\right)\right)^{r-j}$$

Proof. See Appendix B.2.

7 Implementation

We implemented the protocols by employing a small set of test images and features. The server side maintained a database of features which contained a record of features, the corresponding question and the number of images pertaining to the features. The server also maintained an archive of images related to the features. The images used were scaled down to be more feasible for use on a mobile device using wireless communication. An example feature used was: “cartoon”. The corresponding question was: “Does the picture contain a cartoon?”. The image archive contained two disjoint sets of images for the feature “cartoon”. The images which satisfied the feature were labeled *yes-images* and the ones that did not were labeled as *no-images*. The administrator had the privilege to add, remove or update features and/or images in the feature database and the image archive.

The client side was implemented on a PDA. A small prototype for Protocol P2 was made, with $m = 4$, but was not used for subsequent study. Instead Protocol P1 was fully implemented and a user survey was made. Each pass of the protocol implemented, consisted of an image displayed on the PDA showing two buttons at the bottom, labeled “yes” or “no”, as shown in Figure 2. The values $k = 10$ and $s = 8$ were chosen, which meant that the user would be accepted if he or she answers correctly at least 8 times. A user was registered by first assigning a unique *ID* to the user after which he or she was given the secret feature. Once registration was done, the users could freely test the identification protocol.

The users used in the survey were all graduate school students. They were asked to attempt Protocol P1 3 times. The average time taken by the user as well as the number of errors was noted down. The number of times the users failed to authenticate was recorded. After the experiment, the users were asked to describe their experience of using the protocol in terms of the difficulty of usage

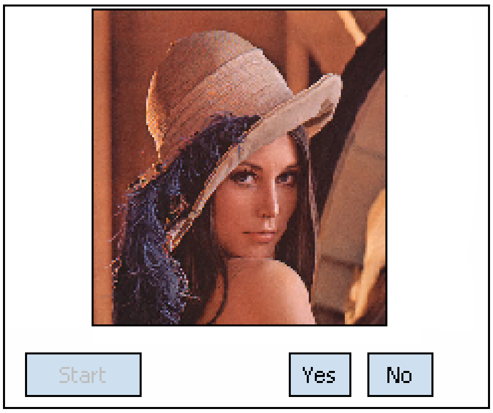


Fig. 2. One pass of Protocol P1 as implemented on the PDA

Table 1. User Statistics and Experience

Success Percentage and Average Time

Users	Attempts	Successes	Success %	e	Avg. Time
5	15	12	80	0.113	25.6 sec

Difficulty of Usage

Total	Easy	Normal	Difficult
5	4	1	0

Lengthiness

Total	Normal	Little Long	Too Long
5	1	4	0

and lengthiness. Both these characteristics were divided into three qualitative categories as shown in Table 1.

The average time taken by the users came out to be 25.6 seconds, with the best time being 21 seconds and the worst being 38 seconds. The maximum number of errors in an authentication session was 4. It should be noted that since the number of images were small, some images were repeated a small number of times. A user erring in one of the images would err again with the same image. This will certainly not be the case if the set of images is large. The average time of 25.6 seconds meant that the user on average spent 2.56 seconds per image. Since 80 percent of the users described executing the protocol as easy, we can roughly associate the probability 0.80 with the variable α in Definition 2. Also, the total successful attempts were 12 out of 15 which allows us to loosely associate a probability 0.80 with the parameter β. Thus we can state that Protocol P1 is $(0.80, 0.80, 2.56k)$-human executable, where k is the protocol's security parameter. We do acknowledge the extremely low number of test subjects due to resource limitations. Such a small portion might not be a complete representative of the human population. Nevertheless we can get a rough idea about the usability of the protocol. Finally it should be noted that most users described the protocol as a *little lengthy*. This opinion is sure to change into *very lengthy* if the value of k is chosen to be somewhere around 20. We acknowledge this as a drawback of the protocol.

8 Conclusion

While secure human identification protocols have been proposed in the literature for normal desktop computers, it is an interesting line of research to construct protocols that can be used on mobile devices. Mobile devices among other constraints have a smaller display unit. We have proposed variations of the protocol proposed in [1] that are practical for devices with smaller display units. However, these protocols are inherently heavy on devices with limited memory and computational power. Overcoming this limitation remains an open problem as most of the human identification protocols are graphics-based which inherently makes them unsuitable for resource constraint devices. The proposed protocols are secure under a slightly different conjecture from the original one. The protocols are usable if used sparingly, as the number of rounds in one authentication session is required to be high for strong security. Furthermore, it is not evident how the protocols will be secure under an active adversary without compromising the usability. It is therefore desirable to propose human identification protocols secure against active and passive adversaries alike as well as being practical on resource limited devices. The proposed protocols in this paper are a small step in that direction.

References

1. Jameel, H., Shaikh, R.A., Lee, H., Lee, S.: Human identification through image evaluation using secret predicates. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 67–84. Springer, Heidelberg (2006)
2. Matsumoto, T., Imai, H.: Human Identification through Insecure Channel. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 409–421. Springer, Heidelberg (1991)
3. Jermyn, I., Mayer, A., Monrose, F., Reiter, M., Rubin, A.: The design and analysis of graphical passwords. In: 8th USENIX Security Symposium (1999)
4. Wang, C.H., Hwang, T., Tsai, J.J.: On the Matsumoto and Imai's Human Identification Scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 382–392. Springer, Heidelberg (1995)
5. Matsumoto, T.: Human-computer cryptography: An attempt. In: 3rd ACM Conference on Computer and Communications Security, pp. 68–75. ACM Press, New York (1996)
6. Li, X.-Y., Teng, S.-H.: Practical Human-Machine Identification over Insecure Channels. Journal of Combinatorial Optimization 3, 347–361 (1999)
7. Hopper, N.J., Blum, M.: Secure Human Identification Protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
8. Li, S., Shum, H.-Y.: Secure Human-computer Identification against Peeping Attacks (SecHCI): A Survey. Unpublished report, available at Elsevier's Computer Science Preprint Server (2002)
9. Dhamija, R., Perrig, A.: Deja Vu: A User Study using Images for Authentication. In: Proc. of the 9th USENIX Security Symposium, pp. 45–58 (2000)
10. Passfaces Corporation: White Paper. The Science behind Passfaces (2005), http://www.passfaces.com
11. Sorensen, V.: PassPic - Visual Password Management (2002), http://www.authord.com
12. Weinshall, D.: Cognitive Authentication Schemes Safe Against Spyware (Short Paper). In: IEEE Symposium on Security and Privacy, pp. 295–300 (2006)
13. Golle, P., Wagner, D.: Cryptanalysis of a Cognitive Authentication Scheme. Cryptology ePrint Archive, Report 2006/258, http://eprint.iacr.org/
14. Bellare, M.: Practice-Oriented Provable-Security. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 221–231. Springer, Heidelberg (1998)
15. Jameel, H., Lee, H., Lee, S.: Using Image Attributes for Human Identification Protocols. Technical Report, CoRR abs/0704.2295 (2007), http://arxiv.org/abs/0704.2295

A Numerical Values of Parameters in Protocol P1 and P2

Table 2 illustrates different allowable values for the parameters in Protocol P1 and P2[2]. So, for instance, if $p \geq 0.90$ and $k = 10$ is desired, s should be equal to 8, with the error e being less than or equal to 0.1 and the adversary's advantage $\delta(10) \leq 0.05$. The '∞' symbol indicates that $\delta(k)$ is undefined for the corresponding choice of parameters. In other words, adversary's probability of success will always be higher than $1 - p$ for these choices. Hence Protocols P1 and P2 are not logically complete under these parameter values.

Table 2. Numerical Values of the Parameters

p	e	k	s	$\delta(k)$	p	e	k	s	$\delta(k)$	p	e	k	s	$\delta(k)$
0.90	0.10	10	8	$\leq$0.05	0.80	0.10	10	8	$\leq$0.11	0.70	0.10	10	9	$\leq$0.27
		15	12	$\leq$0.10			15	13	$\leq$0.23			15	13	$\leq$0.27
		20	16	$\leq$0.13			20	17	$\leq$0.24			20	17	$\leq$0.27
		25	21	$\leq$0.20			25	21	$\leq$0.24			25	22	$\leq$0.31
	0.20	10	6	∞		0.20	10	7	∞		0.20	10	7	$\leq$0.0655
		15	10	∞			15	11	$\leq$0.0923			15	11	$\leq$0.1322
		20	14	$\leq$0.0327			20	15	$\leq$0.1335			20	15	$\leq$0.1675
		25	17	$\leq$0.0327			25	18	$\leq$0.1176			25	19	$\leq$0.1895
	0.30	10	5	∞		0.30	10	6	∞		0.30	10	6	∞
		15	8	∞			15	9	∞			15	10	$\leq$0.0648
		20	11	∞			20	12	∞			20	13	$\leq$0.0658
		25	15	∞			25	16	$\leq$0.0358			25	16	$\leq$0.0672

B Security of the Protocols

B.1 Proof of Claim 3

There is no computation done by $\mathcal{H}$ or the computing device except for displaying the image. Now, if the display unit is *large enough* to display the whole image, then $\mathcal{A}$ has no advantage over $\mathcal{A}'$ whatsoever. Therefore, $\mathcal{A}$ is 0-almost equivalent to $\mathcal{A}'$, where $\lambda(r) = 0$ is the constant function. Let us assume that $\mathcal{A}'$ defeats the protocol with probability $> p'$. We construct an adversary $\mathcal{B}$ that uses $\mathcal{A}'$ to violate Conjecture 2. In the *training* phase, $\mathcal{B}$ simply provides $\mathcal{A}'$ the images being given to it and with probability $1 - e$ it provides the correct answers to these images to $\mathcal{A}'$. When $\mathcal{A}'$ completes the training phase, $\mathcal{B}$ provides the image i to $\mathcal{A}'$ whose answer $\mathcal{B}$ has to guess. Whenever, $\mathcal{B}$ gets an answer it outputs the answer and halts. Since:

$$p' = \sum\nolimits_{j=s}^{r} \binom{r}{j} \left(\frac{1}{2} + \delta(r)\right)^{j} \left(\frac{1}{2} - \delta(r)\right)^{r-j}$$

[2] Notice that for Protocol P2, $k = mk'$.

This means that the probability that $\mathcal{B}$'s guess is correct is $> \frac{1}{2} + \delta(r)$, where r is the number of runs the adversary $\mathcal{A}'$ needs in the training phase. This contradicts Conjecture 2 and the result follows. $\square$

B.2 Proof of Claim 4

The computing device has to display images in the correct order which is also visible to $\mathcal{A}'$. $\mathcal{H}$ just needs to recall σ and hence does not have to do any computation. If the display unit is large enough to display all m images at the same time then adversary $\mathcal{A}$ has no advantage over $\mathcal{A}'$ in this regard as well. However, if the computing device has a smaller display unit (a mobile device), then $\mathcal{H}$ has to scroll left and right to answer the images in the order specified by σ. Thus $\mathcal{A}$ has an advantage over $\mathcal{A}'$, but which cannot be more than $\delta(r)$, or else it will violate Conjecture 2. Therefore, $\mathcal{A}$ is $\delta(r)$-almost equivalent to $\mathcal{A}'$, where $\lambda(r) = \delta(r)$. Suppose now that $\mathcal{A}'$ defeats the protocol with probability $> p'$. We can construct an adversary $\mathcal{B}$ which uses $\mathcal{A}'$ in the same way as in the proof of Claim 3 except now it samples a random $\sigma \overset{R}{\leftarrow} \sigma[m]$ and feeds $\mathcal{A}'$ with the images and their answers in accordance with σ. By an argument similar to the previous section we can show that the probability that $\mathcal{B}$'s guess is correct is $> \frac{1}{2} + \delta(r)$, where $r \equiv 0 \bmod m$ is the number of images or answers shown to $\mathcal{A}'$ in the training phase. Since this contradicts Conjecture 2, we can say that Protocol $\mathsf{P2}$ is (p', r) secure against the adversary $\mathcal{A}'$ and (p'', r) secure under $\mathsf{P\text{-}MTM}$, where $p'' \leq p' + \delta(r)$. $\square$

Ternary Subset Difference Method and Its Quantitative Analysis

Kazuhide Fukushima[1], Shinsaku Kiyomoto[1], Toshiaki Tanaka[1], and Kouichi Sakurai[2,3]

[1] KDDI R&D Laboratories Inc.
[2] Faculty of Information Science and Electrical Engineering, Kyushu University
[3] Institute of Systems, Information Technologies and Nanotechnologies

Abstract. This paper proposes a ternary subset difference method (SD method) that is resistant to coalition attacks. In order to realize a secure ternary SD method, we design a new cover-finding algorithm, label assignment algorithm, and encryption algorithm. These algorithms are required to revoke one or two subtrees simultaneously while maintaining resistance against coalition attacks. We realize this two-way revocation mechanism by creatively using labels and hashed labels. Then, we evaluate the efficiency and security of the ternary SD method. We show that the upper bound of the average message length in the ternary SD method is smaller by about 12.2 percent than that of the conventional SD method, and the number of labels on each client device can be reduced by about 20.4 percent. On the other hand, the computational cost imposed on a client device stays within $O(\log n)$. Finally, we prove that the ternary SD method is secure against coalition attacks.

Keywords: Broadcast Encryption, Subset Difference Method, Ternary Tree.

1 Introduction

1.1 Background

Recently, high-speed Internet has been expanded to include 3G mobile services. Mobile content delivery services using broadcasting technology have become major services in the market, and pay broadcasting is expected to become a new service of particular importance in the near future. However, copyright protection is a serious issue for these services. As copies of digital content can be made easily with little effort, illegal content circulates widely. Thus, content must be encrypted for copyright protection in digital content distribution services so that the following two properties are satisfied: 1) only valid client devices can decrypt the content, 2) if the keys stored in a client device are revealed, the client device should be revoked so that it can no longer decrypt the content. A *broadcast encryption scheme* realizes these important requirements, and it is an essential technique for ensuring the security of broadcasting services. Naor et al. proposed the subset difference method (the SD method) [1]. Thereafter, many improved versions of the SD method have been proposed. However, no feasible broadcast encryption schemes that reduce both the average message length and storage have been proposed. An efficient broadcast scheme is desired in order to reduce the cost and load for key-management in broadcasting services.

Motivation: This paper proposes an SD method with a ternary tree that can reduce both message and storage size compared to the conventional SD method while imposing a feasible computational cost: $O(\log n)$ on client devices.

1.2 Previous Work

A broadcast encryption scheme was first proposed by Berkovits [2]. Fiat et al. [3] formalized the basic definition of a broadcast encryption scheme. Naor et al. [1] proposed the complete subtree method (the CS method). The scheme uses an a-array tree, and client devices are assigned to the leaf nodes of the tree. Valid client devices are covered by complete subtrees, and a key to encrypt the session key is assigned to each subtree. There is one problem associated with the CS method in that the message length increases in proportion to the number of revoked client devices. The average message length in the CS method is given by $O(r \log_a(n/r))$ for the number of total client devices n, the number of revoked client devices r and the degree of the tree a. Naor et al. [1] also proposed the subset difference method (the SD method). The SD method uses a logical binary tree to assign labels to client devices. A valid client device can derive the key to decrypt the message using its labels. The valid client devices are covered by subtrees that contain a complete subtree covering revoked client devices. A key to encrypt the session key is assigned to each subtree. The message length in the average and worst case scenarios is given by $2 \log 2 \cdot r$ and $2r - 1$, respectively. Furthermore, each client device stores $(\log^2 n/2 + \log n/2 + 1)$ labels.

Many improved versions of the SD method have been proposed. Halevy et al. [4], Goodrich et al. [5], Jho et al. [6], Hwang et al. [7] and Attrapadung et al. [8] proposed schemes based on a pseudo-random number generator. Asano [9], Attrapadung et al. [10] and Gentry et al. [11] proposed a scheme based on the RSA cryptosystem. Jho et al.'s scheme reduces the message length to r/c but increases the storage size, i.e., the number of keys/labels each client must store, to $O(n^c)$ where c is constant. Other schemes reduce the storage size to less than $O(\log^2 n)$ but increase the average message length to greater than $2 \log 2 \cdot r$. Boneh et al. [12] proposed a scheme based on Pairing in which the message length and storage size do not depend on r; however, this scheme imposes a heavy computational cost: $O(n)$ on client devices.

Challenge: The schemes based on an RSA cryptosystem and pseudo-random number generator involve a tradeoff between message length and storage size. Computational cost is heavy in the scheme based on pairing. Therefore, a broadcast encryption scheme with a smaller message and storage size, and a feasible computational cost is desired.

1.3 Our Contribution

This paper proposes a coalition resistant ternary SD method, which is based on the subset difference method with a logical ternary tree. In order to achieve a secure ternary array SD method, we design a new cover-finding algorithm, label assignment algorithm, and encryption algorithm. These algorithms are required to revoke one or two subsets simultaneously while maintaining resistance against coalition attacks. We realize this two-way revocation mechanism by creatively using labels and hashed labels. Next, we evaluate the efficiency and security of the ternary array SD method. We evaluate the upper bound of message length in the average and worst case scenarios based on a theoretical analysis of the cover-finding algorithms. Additionally, we evaluate the storage size and computational cost imposed on client devices based on analysis of the label assignment algorithm and decryption algorithm. Finally, we prove that the ternary SD method is secure against coalition attacks.

Results of the Efficiency Analysis: The message length in the ternary SD method is bounded by $n/3$ and $2r - 1$ in the worst case scenario and $3\log(3/2) \cdot r$ in the average case scenario. n denotes the total number of client devices, and r denotes the number of revoked client devices. The storage size, i.e., the number of labels and hashed labels, on client devices is given by $\log_3^2 n + \log_3 n + 1$. The computational cost imposed on client devices to derive the session key is $O(\log n)$.

Result of the Security Analysis: The ternary SD method is a collusion resistant broadcast encryption scheme.

Comparison with Conventional SD Method: The ternary SD method can be implemented using the same primitives as the conventional SD method. The upper bound of message length in the ternary SD method $n/3$ is lower than that in the conventional SD method, $n/2$, while the upper bound $2r - 1$ in terms of r coincides. In the average case scenario, the upper bound of message length in the ternary SD method is smaller by about 12.2 percent than that of the conventional SD method. The storage size imposed on each client device can be reduced by about 20.4 percent using the ternary SD method. Finally, the level of protection against coalition attacks provided by the ternary SD method is equal to that provided by the conventional SD method.

2 Preliminary

Let N be the set of all the client devices, $|N| = n$, and $R(\subset N)$ be the set of client devices, $|R| = r$, whose decryption privileges are revoked. The goal of a broadcast encryption scheme is to transmit message M to all client devices such that any device in $N \backslash R$ can decrypt the message correctly but none of the coalition of client devices in R can decrypt it. $N \backslash R$ denotes the set difference of N and R, or formally, $N \backslash R = \{x | x \in N \text{ and } x \notin R\}$.

A broadcast encryption scheme consists of a label assignment algorithm, encryption algorithm, and decryption algorithm.

Label Assignment Algorithm (by the center)
Assign labels to each client device.

Encryption Algorithm (by the center)
Set a family of disjoint subsets $\{S_1, \ldots, S_w\}$ such that $\cup_{t=1}^{w} S_t = N \backslash R$ using a cover-finding algorithm. Next, assign keys $L_1, \ldots, L_w$ to each subset. Then, encrypt message M with session key K and encrypt K with keys $L_1, \ldots, L_w$.

Decryption Algorithm (by each client device $d \in N \backslash R$)
Find subset S_t to which d belongs. Then, derive key L_t to decrypt K and obtain M.

The proposed method uses a logical ternary tree to assign labels to client devices. Each leaf node of the tree corresponds to each client device. D_i denotes the set of client devices assigned to leaf nodes in the complete subtree rooted at i. Labels and hashed labels are assigned to each client device using the label assignment algorithm in Sect. 5.1 and they are never updated. A client device in $N \backslash R$ can obtain the key to decrypt the message using these labels and hashed labels. Additionally, the proposed method uses the following primitives:

- An encryption function $F_K : \{0,1\}^* \rightarrow \{0,1\}^*$ to encrypt message M.
- An encryption function $E_L : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ to encrypt session key K.
- Cryptographic hash functions h, h_l, h_c, h_r, and h_{kg}: $\{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$. These hash functions must be independent. h_l, h_c, h_r, and h are used to derive label $l(u,w)$ from label $l(u,v)$ or hashed label $h(l(u,v))$ where node w is a child of node v. For example, $l(u,w) = h_l(h(l(u,v)))$ holds for any u, v and w such that w is the left child of v. The same holds for h_c or h_r if w is the center or right child of v, respectively. h_{kg} is used to derive a key from a label.

3 Trivial Expansion of the Conventional SD Method

We consider a straightforward expansion of the conventional SD method. In the SD method, a client device is assigned to a leaf node of the binary tree and given labels in the form of $l(i,j)$ defined by two node i and j. i is a node on the path from the root node to the node where the client device is assigned and j is a descendant node of i just hanging on this path. We can apply this label assignment algorithm to an a-array tree with $a \geq 3$. However, this label assignment algorithm has a drawback in that it cannot protect against coalition attacks.

We provide the following example. Figure 1 shows the label assignment for the ternary tree. Note that all the labels client device d_3 has, i.e., $l(1,3)$, $l(1,4)$, $l(1,5)$, $l(1,6)$, $l(2,5)$, $l(2,6)$, and $l(all)$, can be collected from the labels that client devices d_1 and d_2 have. Even if client devices d_1 and d_2 are revoked, a coalition of these devices can still obtain subsequent session keys by pretending to be a legitimate client device d_3. Thus, we need a new label assignment algorithm to protect against coalition attacks.

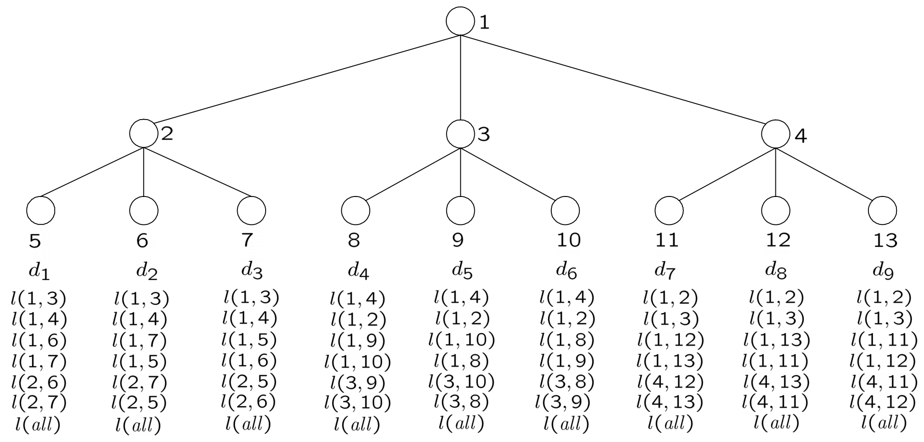

1
2
3
4
5
6
7
8
9
10
11
12
13
d_1 $l(1,3)$ $l(1,4)$ $l(1,6)$ $l(1,7)$ $l(2,6)$ $l(2,7)$ $l(all)$
d_2 $l(1,3)$ $l(1,4)$ $l(1,7)$ $l(1,5)$ $l(2,7)$ $l(2,5)$ $l(all)$
d_3 $l(1,3)$ $l(1,4)$ $l(1,5)$ $l(1,6)$ $l(2,5)$ $l(2,6)$ $l(all)$
d_4 $l(1,4)$ $l(1,2)$ $l(1,9)$ $l(1,10)$ $l(3,9)$ $l(3,10)$ $l(all)$
d_5 $l(1,4)$ $l(1,2)$ $l(1,10)$ $l(1,8)$ $l(3,10)$ $l(3,8)$ $l(all)$
d_6 $l(1,4)$ $l(1,2)$ $l(1,8)$ $l(1,9)$ $l(3,8)$ $l(3,9)$ $l(all)$
d_7 $l(1,2)$ $l(1,3)$ $l(1,12)$ $l(1,13)$ $l(4,12)$ $l(4,13)$ $l(all)$
d_8 $l(1,2)$ $l(1,3)$ $l(1,13)$ $l(1,11)$ $l(4,13)$ $l(4,11)$ $l(all)$
d_9 $l(1,2)$ $l(1,3)$ $l(1,11)$ $l(1,12)$ $l(4,11)$ $l(4,12)$ $l(all)$

Fig. 1. Label assignment in the trivial ternary SD method

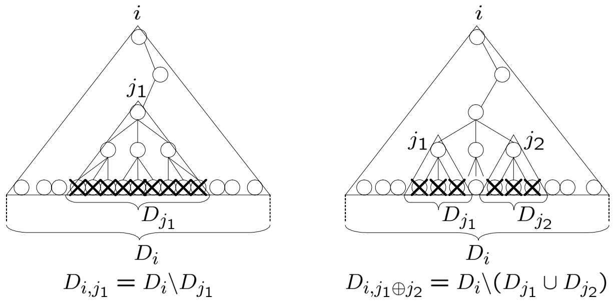

i
j_1
D_{j_1}
D_i
$D_{i,j_1} = D_i \backslash D_{j_1}$
i
j_1
j_2
D_{j_1}
D_{j_2}
D_i
$D_{i,j_1 \oplus j_2} = D_i \backslash (D_{j_1} \cup D_{j_2})$

Fig. 2. Subsets in the ternary SD method

4 Subsets in the Ternary SD Method

We define the subset description method and cover-finding algorithm in our proposed method.

4.1 Subset Description

In the ternary SD method, all valid client devices, i.e., client devices in $N \backslash R$, are covered by the collection of disjoint subsets $S_1, \ldots, S_w$. Each subset S_t is in the form of $D_i \backslash D_{j_1}$ or $D_i \backslash (D_{j_1} \cup D_{j_2})$. The former subset is also used in the conventional SD method and we denote it D_{i,j_1}. The latter subset is a newly introduced subset and we denote it $D_{i,j_1 \oplus j_2}$. In this subset, all the client devices in D_{j_1} and D_{j_2} are revoked. The nodes j_1 and j_2 must be siblings and the descendants of i. Figure 2 shows these two subsets.

```
**Input**: The set of revoked client devices $R$
**Output**: The collection of disjoint subsets $\{S_1, \ldots, S_w\}$ such that $\cup_{t=1}^{w} S_t = N \backslash R$
$T \leftarrow ST(R)$;   $C \leftarrow \phi$;
repeat do
  Find leaf nodes $j_1$, ..., $j_k$ whose sibling(s) are all leaf nodes;
  if $k = 3$ then
    Remove the nodes $j_1$, $j_2$, and $j_3$ from $T$;
  else then /* if $k = 1$ or $k = 2$ */
    Find the lowest ancestor node $i$ of $j_1$,...$j_k$ that has sibling(s);
    if not found then $i \leftarrow$ root; end if
    if $k = 1$ then
      $C \leftarrow C \cup \{D_{i,j_1}\}$;
    else then
      $C \leftarrow C \cup \{D_{i,j_1 \oplus j_2}\}$; /* if $k = 2$ */
    end if
    Remove all the descendant nodes of $i$ from $T$;
  end if
until $T = \{\text{root}\}$
return $C$;
```

Fig. 3. The cover-finding algorithm

4.2 Cover-Finding Algorithm

Figure 3 shows the cover-finding algorithm in the ternary SD method. This algorithm takes as input the set of revoked client devices R, and outputs the collection of disjoint subsets $\{S_1, \ldots, S_w\}$ that partitions $N \backslash R$. Let $ST(R)$ be the tree that consists of leaf nodes that correspond to revoked client devices and their ancestor nodes, and ϕ be the empty set. This algorithm is used in the encryption algorithm in Sect. 5.2.

5 Proposed Method

Here we present a detailed description of the ternary SD method. The ternary SD method consists of 1) label assignment algorithm, 2) encryption algorithm, and 3) decryption algorithm.

5.1 Label Assignment Algorithm

This algorithm is executed in the center:

1. Construct a ternary tree to manage client devices. These client devices are assigned to leaf nodes of the tree.
2. Generates initial labels with λ bits for all the nodes with the exception of the leaf nodes in the tree. Let the initial label for node u be $l(u, u)$. All of the other labels required in this scheme can be derived from these initial labels using hash functions h_l, h_c, h_r, and h according to the rule:

$$l(u, w) = h_l(h(l(u, v))),\ h_c(h(l(u, v))),\ \text{or}\ h_r(h(l(u, v)))$$

where w is the left, center, or right child of v, respectively.

3. Assign labels and hashed labels to client devices. The set $\mathit{Label}(u)$ that the client device at the node u has is given by:

$$\begin{aligned}\mathit{Label}(u) = &\{h(l(u,v)) | u \in \mathit{Path}_u,\ v \in \mathit{LeftHang}_u\} \\ &\cup \{l(u,v) | u \in \mathit{Path}_u,\ v \in \mathit{RightHang}_u\} \\ &\cup \{l(\mathit{all})\}.\end{aligned}$$

   Path_u is the set of nodes that are on the path from the root node to u. $\mathit{LeftHang}_u$ denotes the set of nodes that hang on the left of the Path_u. If Path_u contains the leftmost node, the rightmost sibling is in $\mathit{LeftHang}_u$. $\mathit{RightHang}_u$ denotes the set of nodes that hang on the right of the Path_u. If Path_u contains that rightmost node, the leftmost sibling is in $\mathit{RightHang}_u$. Furthermore, $l(\mathit{all})$ is a special label that is used when there are no revoked client devices.

5.2 Encryption Algorithm

The encryption algorithm is executed in the center on a message M:

1. Choose session key K and encrypt M.
2. Partition all the valid client devices into disjoint subsets $S_1, \ldots, S_w$ using the cover-finding algorithm in Sect. 4.2. Let $L_1, \ldots, L_w$ be the keys associated with these subsets. The key for subset D_{i,j_1} is given by:

$$h_{kg}(h(l(i, j_1))),$$

   and the key for subset $D_{i,j_1 \oplus j_2}$ where $j_2 = \mathit{right}_{j_1}$[1], is given by:

$$h_{kg}(l(i, j_1)).$$

3. Encrypt session key K with keys $L_1, \ldots, L_w$ and send broadcast message

$$\langle [S_1, \ldots, S_w, E_{L_1}(K), \ldots, E_{L_w}(K)], F_K(M) \rangle$$

   to all the client devices.

5.3 Decryption Algorithm

The decryption algorithm is executed in a client device on a received broadcast message

$$\langle [S_1, \ldots, S_w, E_{L_1}(K), \ldots, E_{L_w}(K)], F_K(M) \rangle :$$

1. Find S_t to which the client device belongs. The result is null when this client device is revoked.
2. Derive the corresponding key L_t using the labels and hashed labels assigned to the client device.
3. Decrypt $E_{L_t}(K)$ using the key L_t to obtain K.
4. Decrypt $F_K(M)$ using the key K to obtain and output M.

[1] right_u denotes the immediate right sibling of u. If u is the rightmost node, it means the leftmost sibling. Any two sibling nodes in a ternary tree can be described as "u and right_u".

6 Analysis

We analyze the efficiency and security of the ternary SD method.

6.1 Efficiency Analysis

We evaluate the ternary SD method from three perspectives:

1. Message length, i.e., the length of the header that is attached to $F_K(M)$, which can be evaluated by the number of subsets in the cover.
2. Storage size, which can be evaluated by the number of labels and hashed labels that a client device needs to store.
3. Computational cost, which is imposed on client devices to derive the session key that is used to decrypt a broadcast message.

The message length depends on the location to which revoked client devices are assigned, while the storage size and the computational cost is not dependent on location. Therefore, we present an analysis of the worst and average case scenarios.

Message Length. We evaluate the message length in the worst and average case scenarios.

Worst Case Analysis. We evaluate the maximum message length in the ternary SD method.

A trivial upper bound of the number of subsets is given by $n/3$, in the case where all the client devices are covered by ternary trees with height 1.

An upper bound in terms of r can be evaluated by the number of chains in the alternative description of the cover-finding algorithm in Naor et al.'s paper [1] (in Sect. 3.2, pp. 12). This alternative description is used to construct chains of nodes in $ST(\mathcal{R})$. In the conventional SD method, each chain is in the form $[u_1, \ldots, u_l]$ and satisfies the following condition.

1. u_1, …, u_{l-1} have out-degree 1.
2. u_l is either a leaf node or a node with out-degree 3.
3. The parent of u_1 is either a node with an out-degree 2 or 3, or the root node.

Subset $D_{i,j1}$ corresponds to chain $[i, \ldots, j]$. In the ternary SD method, each chain is in the form $[u_1, \ldots, u_l^{(1)}]$ or $[u_1, \ldots, u_{l-1}, u_l^{(1)}; u_l^{(2)}]$ and satisfies the following conditions:

1. u_1, …, u_{l-2} have out-degree 1, and u_{l-1} has out-degree 1 or 2.
2. $u_l^{(1)}$ and $u_l^{(2)}$ are leaf nodes or nodes with out-degree 3.
3. The parent of u_1 is a node with an out-degree 2 or 3, or the root node.

Subset D_{i,j_1} corresponds to chain $[i, \ldots, j_1]$ and subset $D_{i,j_1 \oplus j_2}$ corresponds to chain $[i, \ldots, j_1; j_2]$.

The head vertex of a chain must be the root node or a child of a node with an out-degree greater than 1. Thus, a branch node (with an out-degree smaller than 3) of $ST(R)$ is a parent node of the head vertices. Let the out-degree of the branch node be b. Then, the number of branch nodes is given by $r/b + r/b^2 + \cdots + 1 = (r-1)/(b-1)$, and, the number of chains is given by $b(r-1)/(b-1)+1$ (the root node is the additional head vertex), which takes the maximum value $2r-1$ when $b = 2$.

Therefore, the size of the broadcast messages is bounded by $n/3$ and $2r-1$.

Average Case Analysis. Naor et al. showed the upper bound of the average message length in the SD method. Their argument is also based on the alternative description of the cover-finding algorithm. They evaluated the expected number of subsets by counting the number of chains with an out-degree of 1 that are not empty (i.e., contain multiple vertices). Note that no subsets are added to the cover when a chain is empty. Consider a chain on which t client devices hang. Then, the probability that the chain is not empty is at most $2^{-(t-1)}$. For any $1 \le t \le r$, there are at most r/t chains on which t client devices hang since each chain contains distinct t devices. Therefore the expected number of non-empty chains is bounded by:

$$\sum_{t=1}^{r} \frac{r}{t} \cdot \frac{1}{2^{t-1}} = 2r \sum_{t=1}^{r} \frac{1}{t2^t} \le 2r \sum_{t=1}^{\infty} \frac{1}{t2^t} = 2\ln 2 \cdot r.$$

By the same logic, the average message length in the ternary SD method is bounded by:

$$\sum_{t=1}^{r} \frac{r}{t} \cdot \frac{1}{3^{t-1}} = 3r \sum_{t=1}^{r} \frac{1}{t3^t} \le 3r \sum_{t=1}^{\infty} \frac{1}{t3^t} = 3\ln\frac{3}{2} \cdot r. \qquad (1)$$

Note that the probability that a chain is not empty is replaced by $3^{-(t-1)}$.

Storage Size. There are $d_T - d$ labels of $l(u,*)$ and $d_T - d$ hashed labels of $h(l(u,*))$ ($*$ denotes *don't-care*) for a node u at depth d, where $d_T \sim \log_3 N$ is the height of the tree. Additionally, the label $l(all)$ is needed. The number of labels and hashed labels stored in client devices is given by:

$$\sum_{d=0}^{\log_3 n} 2(\log_3 n - d) + 1 = \log_3^2 n + \log_3 N + 1 \sim \ln^2 n / \ln^2 3. \qquad (2)$$

Table 1. Comparison between conventional SD method and ternary SD method

Method	Msg. Size (Max.)	Msg. Size (Ave.)	Stor. Size	Comp. Cost
The SD method	$n/2$, $2r-1$	$2\ln 2 \cdot r$	$\ln^2 n/2\ln^2 2$	$O(\log n)$
The ternary SD method	$n/3$, $2r-1$	$3\ln(3/2) \cdot r$	$\ln^2 n/\ln^2 3$	$O(\log n)$

Computational Cost. We evaluate the computational cost imposed on client devices to derive the session key. First, a client device finds the subset to which the device belongs. The subset can be found in $O(\log\log n)$ using the techniques for finding table structures in the CS method. Then, the client device derives the key. In this process, the device uses a hash function at most $2\log_3 n - 1$ times. Thus, the total computational cost is bounded by:

$$O(\log\log n) + 2(\log_3 n - 1) = O(\log n).$$

6.2 Security Analysis

We analyze the security of the ternary SD method.

Theorem 1. *The ternary SD method is secure against coalition attacks*

Proof. We show that for each subset S_t, any coalition of client devices that do not belong to this subset cannot obtain the corresponding key L_t. In the ternary SD method, a subset S_t may be D_{i,j_1} with the revoked subtree rooted at j_1 or $D_{i,j_1 \oplus j_2}$ with the two revoked subtrees rooted at j_1 and j_2 as defined in Sect. 4.1.

The case where S_t is D_{i,j_1}. In this case, the corresponding key is $L_t = h_{kg}$ $(h(l(i,j_1)))$, which can be derived by using label $l(i,j_1)$ or hashed label $h(l(i,j_1))$. We must show that none of the coalitions of client devices in $N\backslash D_i$ and D_{j_1} can obtain the label $h(l(i,j_1))$ or hashed label $h(l(i,j_1))$.

First, no coalition of devices in $N\backslash D_i$ can obtain the key. The label $l(i,*)$ can be derived only from the initial label $l(i,i)$ that is generated randomly and independently of other initial labels. Thus, a coalition of these client devices in $N\backslash D_i$ cannot obtain labels or hashed labels in the form of $l(i,*)$ and $h(l(i,*))$. Therefore, we have to consider only client devices in D_{j_1}.

Next, no coalition of client devices in D_{j_1} can obtain the key. No client device in D_{j_1} has labels or hashed labels in the form of $l(i,j_1)$, $h(l(i,j_1))$, $l(i,u)$, and $h(l(i,u))$ where u is an ancestor node of j_1. On the other hand, the coalition of all the client devices in D_{j_1} can collect all the labels in the form of $l(i,v)$ where v is a descendant of j_1. However, this coalition cannot derive $l(i,j_1)$ from these labels since it is computationally infeasible to find the inverse of the hash functions.

The case where S_t is $D_{i,j_1 \oplus j_2}$. In this case, the corresponding key is $L_t = h_{kg}(l(i,j_1))$, which can be derived by using label $l(i,j_1)$. We must show that none of the coalitions of client devices in $N\backslash D_i$, D_{j_1}, and D_{j_2} can obtain label $l(i,j_1)$.

First, no coalition of devices in $N\backslash D_i$ can obtain the key since none of the coalitions has any labels or hashed labels in the form of $l(i,*)$ and $h(l(i,*))$. Therefore, we have to consider only client devices in D_{j_1} and D_{j_2}.

Next, no coalition of client devices in D_{j_1} and D_{j_2} can obtain the key. No client device in D_{j_1} has labels or hashed labels in the form of $l(i,j_1)$, $l(i,u)$, and $h(l(i,u))$ where u is an ancestor node of j_1. Note that client devices in D_{j_2} have the label $h(l(i, j_1))$; however, it is computationally infeasible to derive $l(i, j_1)$ from $h(l(i, j_1))$.

On the other hand, a coalition of all the client devices in D_{j_1} and D_{j_2} can collect all the labels in the form of $l(i, v)$ where v is a descendant of j_1. However, this coalition cannot derive $l(i, j_1)$ from these labels since it is computationally infeasible to find the inverse of the hash functions. □

7 Discussion

7.1 Comparison with Conventional SD Method

Efficiency. The upper bound of message length in the ternary SD method, $n/3$, is lower than that in the SD method, $n/2$, while the upper bounds in terms of r coincide. The upper bound of the average message length in the ternary SD method is given by $3 \ln(3/2) \cdot r$, which is smaller by about 12.2 percent than that in the conventional SD method, given by $2 \ln 2 \cdot r$.

In the ternary SD method, the storage size on client devices is approximated by $\ln^2 n / \ln^2 3$, which is smaller by about 20.4 percent than that in the SD method, approximated by $\ln^2 n / 2 \ln^2 2$.

On the other hand, the ternary SD method imposes $O(\log n)$ computational cost on a client device, which is identical to the cost of the SD method.

We summarize the above discussion in Table 1.

Primitives and Algorithms. The ternary SD method can be implemented using the same primitives, encryption functions, and hash functions, as the conventional SD method.

However, the cover-finding algorithm, label assignment algorithm, and encryption algorithm in the SD method cannot be applied to the ternary SD method. The cover assignment algorithm in the SD method uses specific operations on the binary tree and it cannot be applied to the ternary SD method in a straightforward way. Thus, we proposed the new cover-finding algorithm that finds subsets of valid client devices by traversing the tree in post-order. The new label assignment algorithm and the encryption algorithm are required to revoke one or two subsets simultaneously while maintaining resistance against coalition attacks. We realize this two-way revocation mechanism by creatively using labels and hashed labels which are assigned to client devices with the label assignment algorithm.

7.2 Extension to Coalition Resistant *a*-Array SD Method

In the ternary tree, any one or two nodes are consecutive. Note that we assume that the leftmost node is adjacent to the rightmost sibling. Our label assignment algorithm based on a hash chain technique works in this situation. The key derived from a hashed label can revoke a single subtree, i.e., the client devices in the other two subtrees can obtain this key. The client devices in another subtree

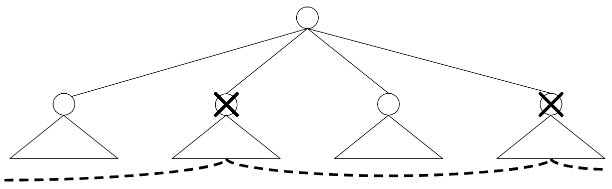


Fig. 4. Two inconsecutive nodes in the quaternary tree

can derive the key using the hashed label. Next, the client devices in the other subtree can derive the key using the label and hash function h. The key derived from a label can revoke two consecutive subtrees. The client devices in the other subtree can only derive the key using this label.

However, in a general a-array tree with $a \geq 4$, there exists sets of nodes that are inconsecutive. Figure 4 shows an example. Our hash chain approach fails with regard to these inconsecutive points.

Thus, the construction of a coalition resistant a-array SD method with reasonable communication, computation, and storage overhead is an open issue.

8 Conclusion

In this paper, we proposed a coalition resistant ternary SD method. The ternary SD method has the following properties: 1) it can be implemented using the same primitives as the conventional SD method, 2) both the average message length and storage size imposed are lower than those in the SD method, 3) the computational cost imposed on client devices to derive the session key is $O(\log n)$, 4) it is a coalition resistant broadcast encryption scheme. We presented quantitative analyses of efficiency and security of the ternary SD method; then, we compared our proposed method with the conventional SD method.

References

1. 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)
2. Berkovits, S.: How to Broadcast a Secret. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 535–541. Springer, Heidelberg (1991)
3. Fiat, A., Naor, M.: Broadcast Encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
4. Halevy, D., Shamir, A.: The LSD Broadcast Encryption Scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–161. Springer, Heidelberg (2002)
5. Goodrich, M.T., Sun, J.Z., Tamassia, R.: Efficient Tree-Based Revocation in Groups of Low-State Devices. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 511–527. Springer, Heidelberg (2004)

6. Jho, N.S., Hwang, J.Y., Cheon, J.H., Kim, M.H., Lee, D.H., Yoo, E.S.: One-Way Chain Based Broadcast Encryption Schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 559–574. Springer, Heidelberg (2005)
7. Hwang, J.Y., Lee, D.H., Lim, J.: Generic Transformation for Scalable Broadcast Encryption Schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 276–292. Springer, Heidelberg (2005)
8. Attrapadung, N., Imai, H.: Practical Broadcast Encryption from Graph-Theoretic Techniques and Subset-Incremental-Chain Structure. IEICE Transaction on Fundamentalof Electronics, Communications and Computer Sciences, Special Section on Cryptography and Information Security E90-A(1), 187–203 (2007)
9. Asano, T.: A Revocation Scheme with Minimal Storage at Receivers. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 433–450. Springer, Heidelberg (2002)
10. Attrapadung, N., Kobara, K., Imai, H.: Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 374–391. Springer, Heidelberg (2003)
11. Gentry, C., Ramzan, Z.: RSA Accumulator Based Broadcast Encryption. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 73–86. Springer, Heidelberg (2004)
12. 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)

A Toy Examples

This section show examples where the number of client devices is $N = 9$. Client devices d_1, d_2, ..., d_9 are assigned to the leaf nodes 5, 6, ..., 13. Then, the client devices are given the labels and hashed labels as shown in Fig. 5.

We consider the case where client devices d_3, d_4 and d_6 are revoked. The collection of subsets $\{D_{2,7}, D_{3,10\oplus 8}, D_{1,2\oplus 3}\}$ shown in Fig. 6 can be found using the cover-finding algorithm in Sect. 4.2. The center encrypts session key K with keys $L_{2,7}$, $L_{3,10\oplus 8}$, and $L_{1,2\oplus 3}$. The center sends the broadcast message

$$\langle [(2,7),(3,10\oplus 8),(1,2\oplus 3), \\ E_{L_{2,7}}(K), E_{L_{3,10\oplus 8}}(K), E_{L_{1,2\oplus 3}}(K)], F_K(M)\rangle$$

to all the client devices. Client devices d_1 and d_2 can derive key $L_{2,7}$ as:

$$L_{2,7} = h_{kg}(h(l(2,7)))$$

since they have $l(2,7)$ or $h(l(2,7))$. d_5 can derive $L_{3,10\oplus 8}$ as:

$$L_{3,10\oplus 8} = h_{kg}(l(3,10)).$$

d_7, d_8, and d_9 can derive key $L_{1,2\oplus 3}$ as:

$$L_{1,2\oplus 3} = h_{kg}(l(1,2)).$$

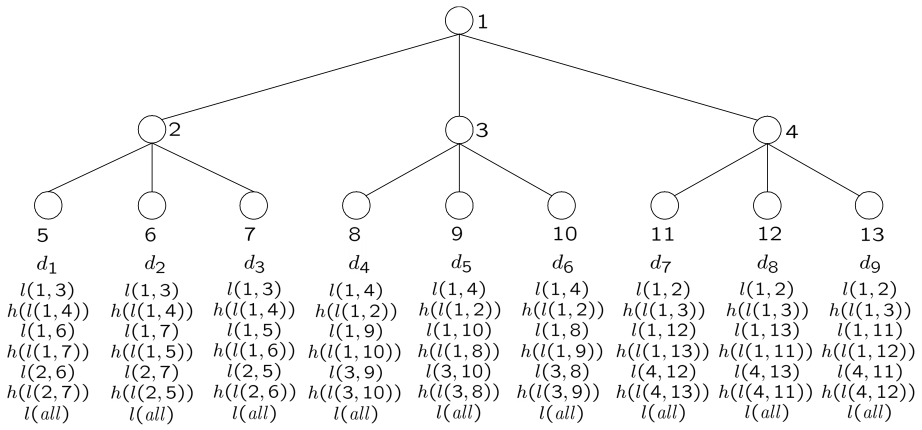

1
2
3
4
5 d_1 $l(1,3)$ $h(l(1,4))$ $l(1,6)$ $h(l(1,7))$ $l(2,6)$ $h(l(2,7))$ $l(all)$
6 d_2 $l(1,3)$ $h(l(1,4))$ $l(1,7)$ $h(l(1,5))$ $l(2,7)$ $h(l(2,5))$ $l(all)$
7 d_3 $l(1,3)$ $h(l(1,4))$ $l(1,5)$ $h(l(1,6))$ $l(2,5)$ $h(l(2,6))$ $l(all)$
8 d_4 $l(1,4)$ $h(l(1,2))$ $l(1,9)$ $h(l(1,10))$ $l(3,9)$ $h(l(3,10))$ $l(all)$
9 d_5 $l(1,4)$ $h(l(1,2))$ $l(1,10)$ $h(l(1,8))$ $l(3,10)$ $h(l(3,8))$ $l(all)$
10 d_6 $l(1,4)$ $h(l(1,2))$ $l(1,8)$ $h(l(1,9))$ $l(3,8)$ $h(l(3,9))$ $l(all)$
11 d_7 $l(1,2)$ $h(l(1,3))$ $l(1,12)$ $h(l(1,13))$ $l(4,12)$ $h(l(4,13))$ $l(all)$
12 d_8 $l(1,2)$ $h(l(1,3))$ $l(1,13)$ $h(l(1,11))$ $l(4,13)$ $h(l(4,11))$ $l(all)$
13 d_9 $l(1,2)$ $h(l(1,3))$ $l(1,11)$ $h(l(1,12))$ $l(4,11)$ $h(l(4,12))$ $l(all)$

Fig. 5. Assignment of the labels in the ternary SD method

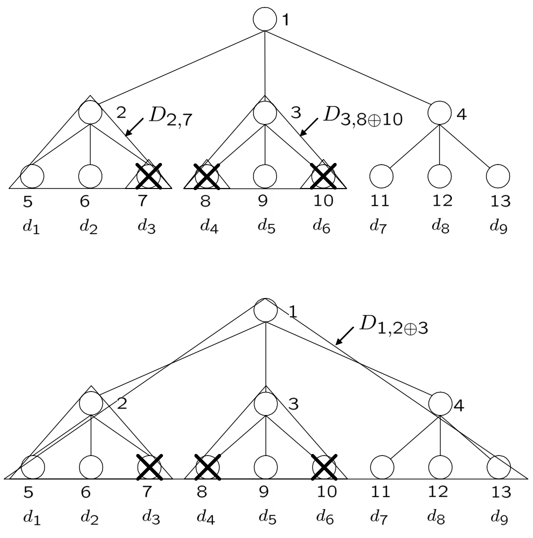

1
2
3
4
$D_{2,7}$
$D_{3,8\oplus10}$
5 6 7 8 9 10 11 12 13
d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9
1
2
3
4
$D_{1,2\oplus3}$
5 6 7 8 9 10 11 12 13
d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9

Fig. 6. The disjoint subsets in the first example

On the other hand, the coalition of d_3, d_4, and d_6 cannot derive $L_{2,7}$, $L_{3,10\oplus8}$, or $L_{1,2\oplus3}$ since neither $h(l(2,7))$, $l(3,10)$, nor $l(1,2)$ can be derived from all the labels and hashed labels they have, which are $h(l(1,2))$, $l(1,3)$, $l(1,4)$, $l(1,5)$, $h(l(1,6))$, $l(1,8)$, $l(1,9)$, $h(l(1,10))$, $l(2,5)$, $h(l(2,6))$, $l(3,8)$, $l(3,9)$, $h(l(3,10))$, and $l(all)$.

Then, we consider the case where d_7 and d_9 are revoked. The collection of subsets $\{D_{1,13\oplus11}\}$, shown in Fig. 7, can be found using the cover-finding algorithm in Sect. 4.2. The center encrypts session key K with keys $L_{1,13\oplus11}$. The center sends the broadcast message

$$\langle[(1,13\oplus11), E_{L_{1,13\oplus11}}(K)], F_K(M)\rangle$$

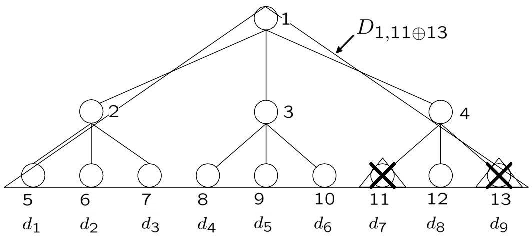


Fig. 7. The subset in the second example

to all the client devices. Client devices d_1, d_2, $\ldots$, d_6 can derive key $L_{1,13\oplus11}$ using $l(1,4)$ or $h(l(1,4))$ as:

$$L_{1,13\oplus11} = h_{kg}(l(1,13)) = h_{kg}(h_r(h(l(1,4)))).$$

d_8 also can derive $L_{1,13\oplus11}$ as:

$$L_{1,13\oplus11} = h_{kg}(l(1,13)).$$

On the other hand, the coalition of d_7 and d_9 cannot derive $L_{1,13\oplus11}$ since neither $h(l(1,4))$ nor $l(1,13)$ can be derived from all the labels and hashed labels they have, which are $l(1,2)$, $h(l(1,3))$, $l(1,11)$, $l(1,12)$, $h(l(1,13))$, $l(4,11)$, $l(4,12)$, $h(l(4,13))$, and $l(all)$.

Data Deletion with Provable Security*

Marek Klonowski, Michał Przykucki, and Tomasz Strumiński

Institute of Mathematics and Computer Science,
Wrocław University of Technology, Poland
ul. Wybrzeże Wyspiańskiego 50-370 Wrocław
{Marek.Klonowski,Michal.Przykucki,Tomasz.Struminski}@pwr.wroc.pl

Abstract. In many systems one of the most important and essential functionalities necessary for secure data processing is the permanent and irreversible deletion of stored bits. According to recent results, it is possible to retrieve data from numerous (especially magnetic) data storage devices, even if some erasing techniques like wiping have been applied. In fact, in many cases a powerful adversary is able to read information that has been many times overwritten.

In our paper we present a new approach to data storage for which a provably secure data deletion seems to be possible. The core idea is based on a particular way of data coding and is generally independent on physical features of the data storage device, so this approach is not limited to magnetic drives. Furthermore, it does not require any special-purpose "secure" device. The usage of a software drivers or installation of modified firmware is sufficient.

We provide rigid mathematical analysis of security of the proposed scheme for some scenarios even in the presence of an extremely powerful adversary. Our approach offers all of this for the price of speed and storage overhead. However, even under pessimistic assumptions this scheme remains fairly practical.

Keywords: Secure data deletion, provable security, cryptographic key storage.

1 Introduction

One of the most important security concerns is reliable erasing of stored data. This issue is being seemingly solved by using techniques such as wiping. It is commonly believed that data overwritten many times is permanently lost and cannot be derived from the data storage device [11]. However, this is not true generally. In many cases it is possible to derive at least some bits of the "erased" data. For instance, some recent studies show that for magnetic media (such as hard drives) even arbitrarily large number of data overwrites may not guarantee safety [2,4,7]. In fact, in some cases the adversary equipped with very sensitive devices is able to retrieve from a hard drive bits overwritten even many dozens

* The research was partially supported by MNiSW grant N N206 1842 33.

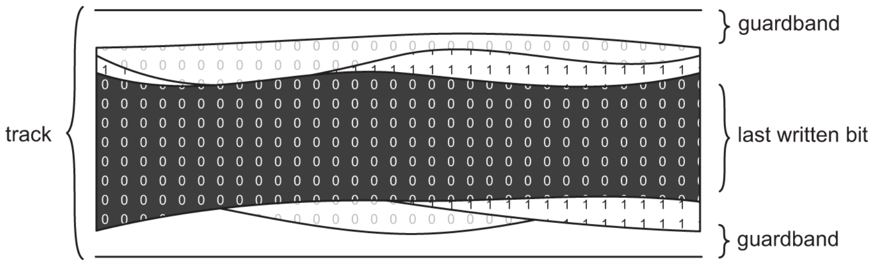


Fig. 1. Symbolical view of a magnetized track on a hard drive

of times. The secret information can be revealed because the new bit is not written to the same physical location as the previous one – the physical mark representing a particular bit can be slightly moved from its expected position whenever we overwrite an old bit. The overview of this situation is presented in figure 1. This enables the adversary to get the former overwritten bit if only he has got an access to a very sensitive device.

For this reason it is believed that there is no chance to completely remove data stored for example on a magnetic drive by simply overwriting it, especially when only limited number of overwrites is possible. Moreover, in many situations even partial physical destruction of a hard disk is not sufficient.

In our paper we present a new approach to data erasing that can be applied for many kinds of data storage devices and which is based on a special coding of data. As an example we analyze a case of magnetic disks, however this approach can be also applied for some other kinds of devices. Proposed approach provides very high level of security and immunity against an even extremely powerful adversary for the price of time and speed overhead. Even though, proposed scheme seems to be fairly practical, especially if we assume that we need to hide relatively short (but crucial for the security) bit-strings, for instance cryptographic material like private keys or seeds for pseudo-random generators.

1.1 Related Work

Erasing of data has been a common and popular subject of many papers and publications so far [4,5,10]. One of the most known is Gutmann's work, where author presents the problem of data deletion from a variety of media types taking magnetic drives under special consideration. Author describes some methods of recovering erased data and indicates a sequence of bit patterns for overwrite operations which according to him guarantees secure data deletion. Physical aspects of writing data on a magnetic media are covered in [2,7,9].

In some sense, spirit of the approach to erasing problem proposed by us is in some sense similar to paper [8] due to Rivest and Shamir. In that paper authors showed how to write several times on a "write-once" memory. It turned out to be possible by using a special representation of data that by-passes physical restrictions (of course in a very limited manner). Similarly, in our paper we suggest a special way of coding data on a magnetic drive. By using it, we make the retrival of large and consistent piece of previously stored data extremely problematic even for the powerful adversary.

1.2 Organization of This Paper

In section 2 we briefly outline the physical aspects of storing and deleting data from magnetic drives. We also present a mathematical and physical model of a special data encoding, which allows us to develop a new scheme for erasing data. Section 3 is devoted to the mathematical analysis of security of the proposed scheme. We start with security investigation of a single bit of a cryptographic key in section 3.3 and then, in section 3.4, we extend achieved results to longer bit-strings. Data wiping algorithms for proposed coding scheme are described in section 3.5. The promising ideas and concepts for future work are shortly presented in section 4. We conclude in section 5.

2 Secure Erasing via Special Coding

In this section we describe a new way of encoding data that allows us to remove almost perfectly data that we want to erase even in the presence of an extremely powerful adversary. We describe the idea in case of magnetic hard drives. However this approach can be also applied for other kinds of storage devices under the condition that some requirements are fulfilled.

2.1 Physical Data Representation on Magnetic Hard Drives and Its Weaknesses

On magnetic hard drives all data is arranged in concentric circles called tracks. Symbolic view of standard tracks on a hard drive is presented in figure 2. For modern hard drives the track pitch is about 200 nanometers wide and guard-bands (the space between tracks needed to provide margins in case of head misalignments) are less than 20 nanometers wide. Including the fact that the

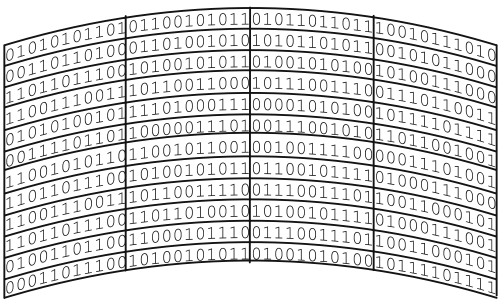


Fig. 2. Standard organization of data on a magnetic drive

head must be placed over the right track in an extremely short time there is no surprise that some inaccuracy of head placement appear (figure 1). This observation can make a successful data retrieval possible.

Techniques Used in Retrieving Old Data. Magnetic force microscopy (MFM) is a well known technique for imaging magnetization patterns with a high resolution. It uses a sharp magnetic tip placed close to the analyzed magnetic surface. A magnetization image is created by moving the tip across the surface and measuring the magnetic force as a function of position.

It has been proved [2,9] that MFM is a technique accurate enough to read erased data from hard drives, even if the desired data has been overwritten. Despite of the fact that magnetic force microscopy has got some serious limitations (low rate of image acquisition and special requirements for samples preparation), more powerful techniques such as spin-stand imaging are being developed [7]. This gives a chance to have a look at someone's erased files and therefore it became the motivation for our work.

In practice we need to assume that the equipped with special device adversary may be able to retrieve even several layers of overwritten bits. Moreover, remarkable is that the regular user is not able to find out if its device is exposed to such an attack without special diagnostic devices. In fact, it depends on the adversary's diagnostic devices sensitiveness and head misalignments which appears during the write operations.

We need to take into account that if one bit of previously stored (overwritten) information can be read, then there is a great chance that also other bits from its physical proximity have been written with the same track misalignment. We should presume that the adversary is usually able to get some amount of consecutive bits of data instead of independent bits spread all over the disk. This helps of course the adversary to retrieve a meaningful information.

2.2 Our Scheme for Data Coding

The coding is based on a following idea – the physical space of a magnetic disk is divided into rectangle-like areas that we call "boxes". Depending on the implementation, each box may contain from several to several dozens of regular bits. In our coding each box represents a single bit. Each box is divided into two subspaces – the first, inner one, represents the value of the box (which is 0 or 1) and the outer, surrounding the inner one, plays the role of a border and always consist 0-bits. This idea is depicted in the figure 3.

A box that in our coding is represented by 1, has got 1s in the inner part. Analogically a box representing 0 has 0s in the inner part. Since outer part of box has always got only 0s, then box representing 0 contains only 0s. To avoid ambiguity of notation we shall call a box that represent 0 a 0-box and a box representing 1 a 1-box.

Writing Data. The boxes in our system play a role of single-bit registers.

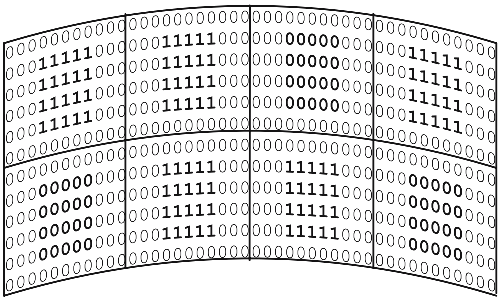


Fig. 3. Isolating bits on a magnetic drive

Preliminaries. At the beginning the whole space is divided (logically) into boxes and is written with 0s. At this moment whole data storage device contains some number of 0-boxes.

Putting new value in the box. If we need to get a particular value i ($i \in \{0, 1\}$) in a particular box, we first check this box by reading bits in its inner part. If it is $(1 - i)$-box we transform it into i-box by overwriting the inner part of this box. Otherwise, if it is already an i-box, we leave it as it is.

Thanks to this approach values in all boxes are independent. It is possible because we used outer parts surrounding the inner parts containing the true value. So we can assume that even significant track misalignment in one of the box does not give any information about the value in other boxes.

As previously assumed, the adversary equipped with sensitive devices can potentiall retrive all layers from a particular box. However, thanks to wide boarders of 0s, he does not learn any information about the history of one box from the history of other boxes.

2.3 From Physical to Mathematical Model

To be able to analyze this approach formally we need to transform physical representation of the proposed coding into a realistic but formal mathematical model. To achive it, firstly need to investigate what is the knowledge of the adversary as well as the knowledge of the regular user.

The Adversary's Perspective. An extremely powerful adversary introduced in our approach, can read layers of a stored data – i.e. given particular box, he is able to say what exactly was written in this box from the very beginning. Account for our special way of coding, he observes consecutive 0's and 1's beginning with 0 and with 0 or 1 at the end, depending on the last bit written in a particular box. In the example the adversary can observe the exact structure presented in the table. The adversary's perspective for n boxes can be represented as a vector $(d_1, d_2, \ldots, d_n)$ where d_i represents the number of changes of the i-th box's values. In our example, the illustration of adversary's perspective is presented in the table.

The Regular User's Perspective. The regular user can see in each box only the last written bit. This is because he can use only the techniques offered by a regular storage device that allows to read the last bit (bolded in our example). Therefore, we can say that the attacker is much stronger than the user.

In this model we can assume that all the adversary knows is $(d_1, \ldots, d_n)$ and nothing more. In the following sections we show that this way of coding in some scenarios will with significant probability not reveal any important information about the stored data to the adversary.

Example. Let us illustrate the described model. Assuming that we have only 8 boxes of information on a data storage device. At the beginning it is filled with 0's and then the following data is stored: 11010110, 01101101, 11010111, 00101111, 11001111, 00011001.

0							
1			**1**				
0	**0**	**0**	0			**0**	
1	1	1	1	**1**		1	
0	0	0	0	0	**0**	0	
1	1	1	1	1	1	1	**1**
0	0	0	0	0	0	0	0
bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8

In the worst case, the powerful adversary can retrieve the whole table depicted above from user's hard drive. However, thanks to our new coding scheme, there is no information leak about consecutive bits caused by head misalignments.

2.4 Analysis of Efficiency of Proposed Scheme

One of the most important issues of our solution is the overhead of time and storage derived by the usage of our coding scheme. If we decide to substitute one single bit with an $m \times n$ box then it is obvious that the storage overhead would be of the $m \cdot n$ order. On the other hand usage of our scheme might significantly reduce the number of bits needed to preserve the consistency of the stored data. According to [10], even 15% of the storage space might be spend on this purpose in modern hard disks. Therefore, the storage overhead might be significantly smaller.

It is more difficult to answer the question concerning the time overhead. We have to take under consideration that before writing we read the data from the disk first and then write only over those boxes, which are of a different value than expected. Still, it is quite intuitive, that if we decide that the inner part of our boxes should consist of $m \times n$, then the time overhead would be also of the $m \cdot n$ order.

Appropriate dimensions of our boxes are a crucial matter. They should derive from experiments aimed at minimizing both inner and outer dimensions with respect to keeping the correlation between neighbouring bits near to 0. Unfortunately, these experiments require a highly skilled personel in electronics and physics and very sophisticated measuring devices, thus are beyond the scope of this work.

We should also mention the fact, that just a small part of the storage device often contains a secret data and the rest is meant for everyday use. Taking it into account we could divide the storage device into two parts. One of them, very small, containing highly valuable and fragile data and another one for casual data. Then we could introduce our scheme of coding only for the hyper-secure part, reaching in this way a very good efficiency/security trade off.

3 Case of Erasing Cryptographic Material

In this section we investigate data deletion problem formally in the case when pseudo-random bit-strings are stored on the magnetic drive according to rules introduced in section 2.2. This covers a very important from practical point of view case when we need to store periodically-changed cryptographic material like private keys or seeds of pseudo-random generators. From mathematical point of view this case seems to be the simplest one (except for some trivial cases), since each bit of information we may treat as a $0-1$ random variable X such that $\Pr(X = 1) = \frac{1}{2}$. What is even more important such random variables representing different bits are stochastically independent.

Notation. Let us recall some standard notation. By $B^{(i)}$ we denote the i-th bit of a bit-string B. We assume also that the binomial coefficient $\binom{n}{m} = 0$ for $m > n$ and $m < 0$. By $\#A$ we understand the number of elements of a finite set A.

Definition 1. *For a bit-string $s = s_1 s_2 \dots s_l \in \{0,1\}^l$ let us define*

$$index(s) = \#\{0 < i < l | s_i \neq s_{i+1}\}$$

i.e. the number of bit flips in the bit-string.

Let us formalize considered scenario.

3.1 Formal Description of Considered Scenario

Writing data. We assume that the user has a register of n boxes and stores there keys (i.e. bit-strings) of the length n. The key is periodically changed $D-1$ times. According to the description included in section 2.2 at the beginning n zeros are placed in the register. At step t, key of the form

$$\mathbf{X}_t = X_t^{(1)} X_t^{(2)} \dots X_t^{(n)}$$

is placed in the register according to the described rules – i.e. some of the bits are flipped.

Moreover, as assumed $\Pr(X_t^{(i)} = 1) = \Pr(X_t^{(i)} = 0) = \frac{1}{2}$ for each $t > 0$ and i. For the convenience of notation we introduce also

$$X^{(i)} = X_1^{(i)} X_1^{(i)} X_2^{(i)} \dots X_D^{(i)}.$$

I.e. $X^{(i)}$ is a sequence of bits stored in the i-th box of the register in consecutive steps

Wiping procedure. In order to hide some information about stored data in the past, the user may perform some additional operations.

Inspection of data storage device by the adversary. The adversary takes the control over the device and he is allowed to observe the state of the data storage device using even very sophisticated methods. Since the special way of coding was used, the adversary is given for each i
- $d_i = \text{index}(X^{(i)})$,
- number of written keys D.

Let us note that for each i, the value d_i is a random variable with the values from the set $\{1, \dots, D\}$.

3.2 Knowledge of the Adversary

The adversary's aim is to retrieve the keys stored in the register using gained knowledge – i.e. parameters d_i for $1 \le i \le n$ and D.

Definition 2. *We define advantage of the adversary for the* $t - th$ *key with respect to the string* $b_1 \dots b_n \in \{0,1\}^n$ *as*

$$\mathcal{A}dv_t(b_1 \dots b_n) = \Pr[\mathbf{X}_t = b_1 \dots b_n | d_1, \dots, d_n, D].$$

If $\mathcal{A}dv_t(b_1 \dots b_n)$ is significant, it would mean that $b_1 \dots b_n$ was used as a t-th key with significant probability. Intuitively, data remains secure if $\mathcal{A}dv_t(b_1 \dots b_n)$ is small for any t and for any $b_1 \dots b_n$. In the ideal (from user's point of view) case advantage of the adversary is always 2^{-n} i.e.:

$$\left(\forall_{b_1 \dots b_n \in \{0,1\}^n}\right) \left(\mathcal{A}dv_t(b_1 \dots b_n) = \Pr[\mathbf{X}_t = b_1 \dots b_n | d_1, \dots, d_n, D] = \frac{1}{2^n}\right).$$

That would mean that the information gained by the adversary during physical inspection of the device does not give **any** information about the key. However from the practical point of view it would be enough to have $\mathcal{A}dv(b_1 \dots b_n) < \varepsilon$ for a small ε and for any $b_1 \dots b_n \in \{0,1\}^n$. This means that there are no strings that are specially likely to have been used as t-th key.

Let us also note that our scheme can provide it only with high probability, since parameters $d_1, \dots, d_n$ are random variables.

3.3 Single Bit Analysis

Since we assumed that stored data bits are independent, then our analysis boils down to analysis of a single bit. Let us concentrate on a single (say, the first) bit's analysis. To make notation more clear we skip some indices – i.e we use X, d instead of $X^{(1)}$ and d_1, respectively.

Now we need to find

$$P_t = \Pr[X_t = 0|d, D],$$

the probability that the first bit of the t-th key was 0, conditioned on the facts that D keys were stored and the bit was changed d times.

Definition 3. *We define the set of trajectories coherent $D-1$ overwrite operations with d bit flips as*

$$S_{D,d} = \{s \in \{0,1\}^D | index(s) = d \wedge s^{(1)} = 0\}.$$

This set can be intuitively understood as the set of "records" of a single box – consecutive bits represent values which are kept there.

Lemma 1. *For any natural D, d*

$$\#S_{D,d} = \binom{D-1}{d}.$$

Proof. This is a trivial fact, since we just need to choose d positions out of $D-1$, where bits are flipped from 0 to 1 or from 1 to 0 in the string of consecutive bits. □

Definition 4. *The set of trajectories coherent $D-1$ overwrite operations with d bit flips with 0 on the t-th position we define as*

$$S^t_{D,d} = \{s \in \{0,1\}^D | index(s) = d \wedge s^{(1)} = 0 \wedge s^{(t)} = 0\}.$$

The set $S^t_{D,d}$ represents "records" of the first box, such that the first bit of the t-th key was 0. Of course $S^t_{D,d} \subset S_{D,d}$.

Lemma 2. *For any natural D, t and any natural even d*

$$\#S^t_{D,d} = \sum_{j=1}^{(d+2)/2} \binom{t-1}{2(j-1)} \binom{D-t}{d-2(j-1)}.$$

Proof. Notice, that we are counting the trajectories of length D with 0 on the t-th position such that the total number of bits flips is d. Let us assume that there are j bit flips to t-th position included and $d-j$ flips after the t-th position. Each of such a trajectory can be seen as a following concatenation $l||r$ such that

$$l \in L^k_t = \{l \in \{0,1\}^t | \text{index}(l) = k \wedge l^{(1)} = 0 \wedge l^{(t)} = 0\}$$

and

$$r \in R^{d-k}_{D-t} = \{r \in \{0,1\}^{D-t} | \text{index}(0||r) = d-k\}.$$

An example trajectory is presented in a figure below.

$$\underbrace{000\boxed{1}11\boxed{0}0\boxed{1}1111\boxed{0}000\boxed{1}\overset{\overset{t}{\downarrow}}{\boxed{\boxed{0}}}}_{l\in L_t^{2(j-1)}}\underbrace{00\boxed{1}\boxed{0}0000\boxed{1}1\boxed{0}00\boxed{1}11\boxed{0}\boxed{1}111}_{r\in R_{D-t}^{d-2(j-1)}}$$

Fig. 4. An example trajectory of length 40 with 13 bit flips and its splitting into subtrajectories $l \in L_t^{2(j-1)}$ and $r \in R_{D-t}^{d-2(j-1)}$

Since we are interested in the strings with 0 on the t-th position and we have started with 0 then

$$\#S_{D,d}^t = \sum_{j=1}^{(d+2)/2} \left(\#L_t^{2(j-1)}\right)\left(\#R_{D-t}^{d-(2(j-1))}\right). \quad (\star)$$

Note that every element with even index is included in the sum. It is because of the restrictions that the number of flips up to the t-th bit is even.

Claim 1

$$\#L_t^{2(j-1)} = \binom{t-1}{2(j-1)}$$

Proof. All subtrajectories $L_t^{2(j-1)}$ are just bit-strings of length t with $2(j-1)$ bit flips, which start and finish with 0. They can be described, using notation from formal languages, as $0^+(1^+0^+)^{j-1}$. To compute $\#L_t^{2(j-1)}$ we must simply evaluate the number of such bit-strings of length i.

Let $\mathcal{A}^{(j)} = (\{w : w = 0^+(1^+0^+)^{j-1}\}, |\cdot|)$ be a combinatorial class, where $|w|$ is a length of bit-string w. Let $\{A_n\}_{n\geq 0}$, $A_n = \#\{a \in \{w : w = 0^+(1^+0^+)^{j-1}\} : |a| = n\}$ denote the counting sequence of a combinatorial class $\mathcal{A}$. The ordinary generating function of sequence $\{A_n\}$, after [1], is

$$A(z) = \frac{z}{1-z}\left(\left(\frac{z}{1-z}\right)^2\right)^{j-1} = \left(\frac{z}{1-z}\right)^{2j-1}$$

which we got from composition of simple combinatorial class and its sequences. By definition, the number we are interested in is just a coefficient near z^i. After applying some elementary transformation, which are described in [3], we finally get

$$A(z) = \left(\frac{z}{1-z}\right)^{2j-1} = \sum_{i=0}^{\infty}\binom{i+2j-2}{i} z^{2j-1+i}.$$

Thus, the number of elements of $L_t^{2(j-1)}$ is

$$L_t^{2(j-1)} = [z^t]A(z) = \binom{t-1}{t-2j+1} = \binom{t-1}{2(j-1)}. \qquad \square$$

Notice, that above result is very intuitive. Here is how we can achieve it in another way: the bit-string has a length of t and it must contain $2(j-1)$ bit flips. Notice, that the first bit must be 0 and cannot be the place where bit flip occurs. Thus, the number of possible bit-strings of this form is exactly $\binom{t-1}{2(j-1)}$.

By analogy, we can easily prove that

Claim 2

$$\#R_{D-t}^{d-2(j-1)} = \binom{D-t}{d-2(j-1)}.$$

Substituting these claims from the facts above into the formula ($\star$) finishes proof of the lemma 2. □

Lemma 3. *For any natural D, t and any natural odd d*

$$\#S_{D,d}^{t} = \sum_{j=1}^{(d+1)/2} \binom{t-1}{2(j-1)} \binom{D-t}{d-2(j-1)}.$$

Proof. Proof of the above lemma is entirely analogous to proof of lemma 2.

Let us prove following lemma:

Lemma 4. *From the adversary's point of view, every sequence of bits from $S_{D,d}$ is equally probable as a sequence of bits in consecutive keys if parameters D and d were observed.*

Proof. For any $s \in S_{D,d}$

$$\Pr[X_1 = s^{(1)}, \ldots, X_D = s^{(D)} | index(X) = d] =$$
$$= \frac{\Pr[X_1 = s^{(1)}, \ldots, X_D = s^{(D)}, \text{index}(X) = d]}{\Pr[index(X) = d]} = \frac{(\frac{1}{2})^{D-1}}{\binom{D-1}{d}(\frac{1}{2})^{D-1}} = \frac{1}{\binom{D-1}{d}}.$$

□

From this lemma we get immediately following corollary

Corollary 1. *For every t*

$$P_t = \frac{\#S_{D,d}^{t}}{\#S_{D,d}} \leq \frac{\sum_{j=1}^{(d+2)/2} \binom{t-1}{2(j-1)} \binom{D-t}{d-2(j-1)}}{\binom{D-1}{d}} = \frac{1}{2} + \varDelta(D, d, t).$$

Remark. We can consider a series $a_j = \binom{t-1}{j}\binom{D-t}{d-j}$. One can note that $0 \leqslant a_1 \leqslant \ldots \leqslant a_Q \geqslant a_{Q+1} \geqslant \ldots \geqslant a_d$ for some $0 \leq Q \leq d$, so in particular we have

$$|\Delta(D,d,t)| \leqslant \frac{a_Q}{S_{D,d}} = \max\left\{ \frac{\binom{t-1}{d}}{\binom{D-1}{d}}, \frac{\binom{D-t}{d}}{\binom{D-1}{d}}, \frac{\binom{t-1}{\left\lfloor \frac{dt+t-D-1}{D+1} \right\rfloor + 1}\binom{D-t}{d - \left\lfloor \frac{dt+t-D-1}{D+1} \right\rfloor - 1}}{\binom{D-1}{d}} \right\}$$

It is easy to see that the problem occurs when we are dealing with very small or very big (i.e. close to $D-1$) values of d, but let us note that d_i – i.e. the number of flips of the content of the i-th box – has binomial distribution $\mathrm{B}(D-1, \frac{1}{2})$. Then, using one of the Chernoff bounds [6] we can prove:

$$\Pr(0.25(D-1) \leq d_i \leq 0.75(D-1)) > 1 - 2\exp\left(-\frac{D-1}{16}\right),$$

so for big D with high probability we have $0 \ll d \ll D-1$. The value of $\Delta(D,d,t)$ is also significant when we have t close to 0 or to $D-1$. This is a serious issue as the last stored keys are usually the most valuable ones and retrieving them by the adversary could cause much trouble to the owner's data. We shall consider these problems in the following sections.

Notice that above inequality provides only a rough estimation of $\Delta(D,d,t)$ value. The exact values of $\Delta(D,d,t)$ are often significantly smaller. To illustrate it, the sample computation had been done and its results are shown in figure 5. Moreover, $\Delta(D,d,t)$ can be expressed precisely using complex hypergeometric series, but in this form it is hardly practical.

3.4 Analysis of a String of Random Data

Considerations from previous subsection combined with the fact that by the assumption on stored data all bits are independent we can formulate following theorem.

Theorem 1. *For any bit-string $b_1 \ldots b_n \in \{0,1\}^n$ and any t*

$$\mathcal{A}dv_t(b_1 \ldots b_n) \leq \prod_{i=1}^{n} \left(\frac{1}{2} + |\Delta(D,d_i,t)|\right)$$

The value $\Delta(D,d_i,t)$ from the above theorem is defined as in the Corollary 1 in the previous section.

Further analysis (and therefore also the user's strategies) strongly depend on the key's length and the value of the parameter D. For large D with high probability all the values d_i will lie in the interval $[\frac{D}{4}, \frac{3D}{4}]$.

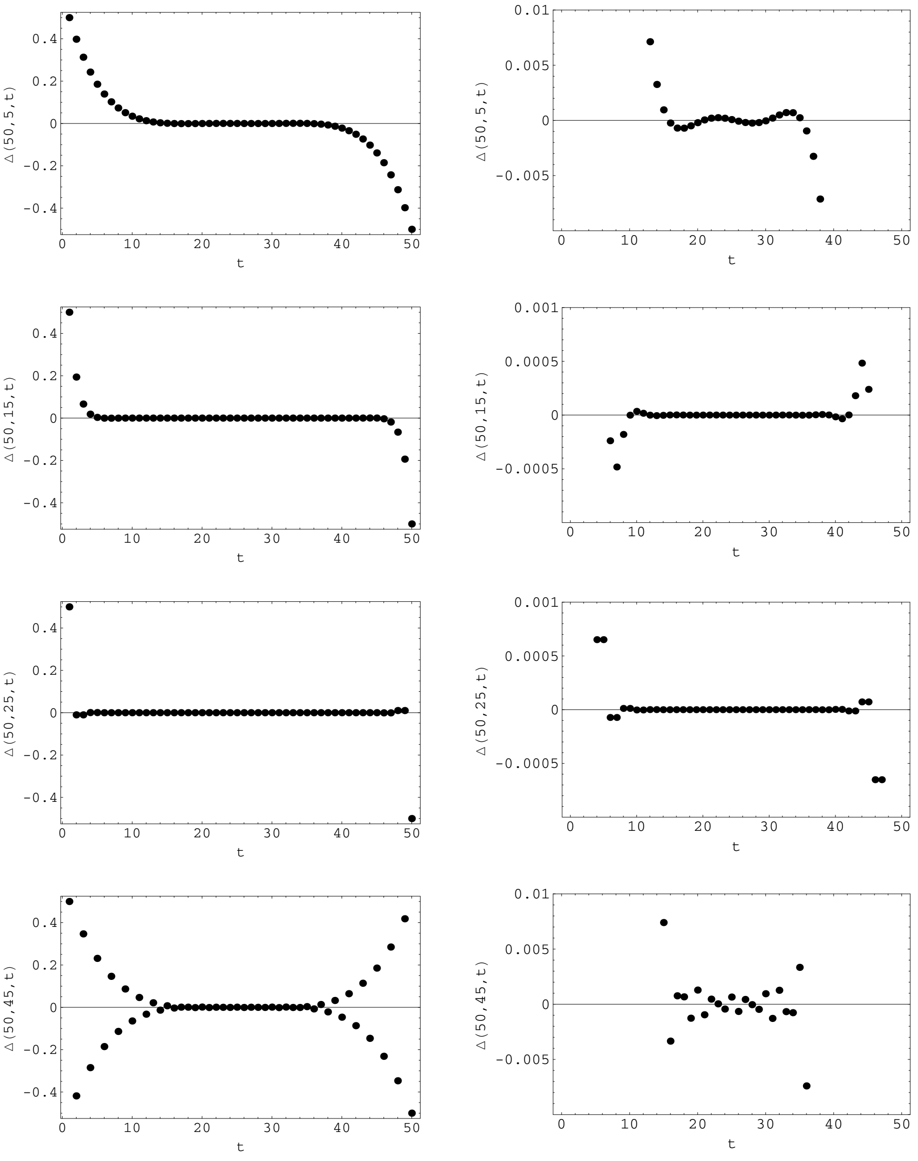


Fig. 5. The exact values of $\Delta(D, d, t)$ for $D = 50$, $d = 5, 15, 25, 45$ and $t \in \{1, \ldots, 50\}$

On the other hand, when the value D is relatively small, using Chernoff's inequality we can say, that in about $k = \lfloor (1 - 2\exp\left(-\frac{D-1}{16}\right))n \rfloor$ boxes the value d_i is close to $\frac{D}{2}$ and for those i the value $\Delta(D, d_i, t)$ is small. Formally speaking

$$\mathcal{A}dv_t(b_1 \ldots b_n) \leq \left(\frac{1}{2} + |\Delta|\right)^k,$$

where $|\Delta|$ is the largest absolute value among the k distinguished values $\Delta(D, d_i, t)$.

3.5 Data Wiping Algorithms

Observations and computations from the previous section lead us to following conclusions:

– for small t the advantage of the adversary is relatively high, therefore using one of t-th first keys are potentially exposed to be retrived by the adversary,
– for t close to D the situation is similar because $|\Delta(D, d_i, t)|$ is big and the adversary's advantage is once again high. What is more, the adversary's knowledge about the D-th key is full. It is of course natural – the last key is explicit written on a disk.

In order to ensure security for our storage scheme and to prevent key leaking in situations mentioned above, we propose to

Step 1. Write some (lets say t_{before}) pseudo-random bit-strings **before** any keys are stored on a magnetic drive (this can be done even by the disk manufacturer).
Step 2. When deletion is to be proceeded, one should overwrite data with particular (lets say t_{after}) number of pseudo-random bit-strings.

Notice that the particular values of parameters t_{before} and t_{after} can be computed with formulas from the previous sections. Although we assume that the regular user does not know the values d_i we can also assume that the value D is known to him (it is known to the adversary so keeping it stored in memory does not cause any additional leak of information). Therefore using Chernoff's inequality the user can estimate the values of d_i and use them to estimate Δ. With such knowledge user can calculate suitable value of t_{after}.

After choosing values of t_{before} and t_{after} the given security level can be achieved. It is possible, that for a very high security level, a significant number of data overwriting operations is necessary at the second step. Therefore, time and space overhead could be remarkable when data larger than cryptographic keys are to be protected by our scheme.

4 Future Work

In our paper we provided the solution of provably secure data deletion from hard drives in a limited domain. The natural extensions of this approach, which are worth taking into consideration are:

– Extended model with assumed a priori adversary's knowledge. In this model bits as well as consecutive layers of data do not have to be independent as in the case of cryptographic keys.
– In our analysis we have assumed that the adversary knows the number of layers of written data (parameter D). However it seems we could relax this assumption in real world scenarios. The adversary does not have to know anything except for the number of changed bits in a particular position. This may lead to stronger results and more effective solutions.

- It seems to be interesting to delete huge space in very short time by flipping minimal number of bits. This is a common problem – in many cases we need to delete data very fast in case of danger. Intuitively it can be applied when the system is based on the scheme proposed above.
- Other data wiping strategies could be examined in order to find one which would provide appropriate security level as well as fast data deletion.

5 Conclusions

When the data deletion is to be proceeded, there appear numerous questions about its security as well as its time complexity. For the model presented in our article, the security of erasing one bit of information can be computed using exact values. If we assume in addition the independence of consecutive bits, the exact result for one bit easily extends to larger data sets. This assumption is very realistic – such situation appears when cryptographic keys are to be stored. We have showed that secure deletion of crucial cryptographic data can be done in such a model, despite of the presence of a very powerful adversary.

Presented in our paper the idea of special data coding for secure data deletion presented in our paper seems to be fairly practical. Nevertheless, particular results are highly dependent on a probabilistic nature of data stored on magnetic drives.

To sum up, planning the possibility of a secure data deletion seems to be a good idea when developing new kinds (or versions) of storage devices. That can be done either at a very low level (hardware, physical persistence of data) as well as at a higher level when the firmware is used in order to encode bit streams. Of course, it would add a cost in the matter of time and space (a constant cost), but we strongly believe that the advantage of having a reliable storage device that enables us to delete data with a given security parameters is worth all of the efforts.

References

1. Flajolet, P., Sedgewick, R.: Analytic combinatorics. Cambridge University Press, Cambridge (2008)
2. Gomez, R.D., Burke, E.R., Adly, A.A., Mayergoyz, I.D., Gorczyca, J.A., Kryder, M.H.: Microscopic investigations of overwritten data. Journal of Applied Physics 73(10) (1993)
3. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics, 2nd edn. A Foundation for Computer Science (1994)
4. Gutmann, P.: Secure Deletion of Data from Magnetic and Solid-State Memory. In: The Sixth USENIX Security Symposium (1996)
5. James, D.G.: Forensically unrecoverable hard drive data destruction Whitepaper (2006), http://infosecwriters.com
6. Janson, S., Łuczak, T., Ruciński, A.: Random Graphs. John Wiley & Sons, Chichester (2001)

7. Mayergoyz, I.D., Tse, C., Krafft, C., Gomez, R.D.: Spin-stand imaging of overwritten data and its comparsion with magnetic force microscopy. Journal of Applied Physics 89(11) (2001)
8. Rivest, R.L., Shamir, A.: How to reuse a "write-once" memory, Information and Control, vol. 55, Belgium (1982)
9. Rugar, D., Mamin, H.J., Guethner, P., Lambert, S.E., Stern, J.E., McFadyen, I., Yogi, T.: Magnetic force microscopy: General principles and application to longitudinal recording media. Journal of Applied Physics 68(3) (1990)
10. Sobey, C.H.: Recovering unrecoverable data, A ChannelScience White Paper (2004)
11. US Department of Defense: National Industrial Security Program Operating Manual, disk sanitization, DoD 5220.22-M

A Probing Attack on AES

Jörn-Marc Schmidt[1,3] and Chong Hee Kim[2,⋆]

[1] Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
joern-marc.schmidt@iaik.at

[2] UCL Crypto Group, Université Catholique de Louvain, Belgium,
Place du Levant, 3, Louvain-la-Neuve, 1348, Belgium
chong-hee.kim@uclouvain.be

[3] Secure Business Austria (SBA),
Favoritenstraße 16, 1040 Vienna, Austria

Abstract. The Advanced Encryption Standard (AES) defines the most popular block cipher. It is commonly used and often implemented on smart cards. In this paper, we show how a 128-bit AES key can be retrieved by microprobing. Thereby, a probe is placed onto the chip to spy on inner values. Watching one arbitrary bit of the AES State during the first two rounds of about 210 encryptions is enough to reveal the whole key. For special positions of the probe, this number can be reduced to 168. The paper demonstrates that even few information is sufficient for a successful attack on AES.

Keywords: Probing Attack, AES, Smart Card.

1 Introduction

In order to provide security on a smart card or an embedded system, several different threats have to be taken into account. Besides traditional cryptanalytical methods, which attack the algorithm itself and treat the device as a black box, implementation attacks try to benefit from examining the device and its behavior. These attacks became more and more popular during the last decade.

Implementation attacks can be passive or active. Passive attacks collect information to undermine the security of a device. Such information can be the time a computation takes [1], the power it needs [2] or its electromagnetic emissions [3]. This information can be sufficient for a successful attack, if it correlates with the secret data used within the computations. In contrast to passive attacks, active ones try to influence the behavior of a device and determine sensitive information by examining the effects of a successful manipulation. As long as no modification of the device is needed, attacks are called non-invasive. Inserting peaks into the clock signal is a well known example for an active non-invasive attack. These peaks may corrupt data transferred between registers and memory [4]. It is also possible to apply semi-invasive and invasive attacks [5]. These attacks require

⋆ Supported by Walloon Region, Belgium / E.USER project.

a decapsulation procedure to have access to the chip surface. In this way, it is possible to modify the behavior of a computation in a more precise way.

Another serious threat to smart cards is microprobing [4]. Thereby, probes are used to spy on inner values of the device. Possible targets of such an attack can be the internal bus, registers or static memory. Probing is a passive but invasive attack, as direct electrical contact to the chip surface has to be established but no change of the behavior is intended. The idea of using probing to attack an algorithm was presented by Helena Hanschuh et al. [6]. They showed how to attack RSA and DES if an adversary can probe the value of a few RAM or address bus bits.

In order to perform such an attack, a probe has to be placed on the chip surface. Therefore, preparations are necessary. Before a needle can be placed onto the chip, the passivation from the position of interest has to be removed. For probing small structures, a test pad may be necessary. Such a pad can be placed by a focused ion beam. This procedure is time consuming and costly. Thus, it is favorable to minimize the number of pads on the chip. As the position of the secret data in the memory is usually unknown, not only a small number of needed, but also a wide range of possible positions for a successful attack is desired.

Our Contribution. In this paper, we present an attack that needs one probe that spies on an arbitrary bit within the 128-bit AES State. We show that observing the bit throughout the first two rounds of 210 encryptions of chosen plaintexts is enough to determine the whole key. For special positions of the probe this can be reduced down to a number of 168. In the case of known plaintext, three precisely-placed probes and 25 encryptions are sufficient.

The paper is organized as follows. After introducing notations and briefly summarize the AES algorithm in Section 2, the attack model is described in Section 3. In Section 4, the attack on AES for an arbitrary probed bit of the State is presented. Section 5 shows how the attack can be optimized under special conditions and briefly considers the known plaintext case. Conclusion is drawn in Section 6.

2 Preliminaries

2.1 Notations

Let S^i denote the intermediate State after the i^{th} operation. This is depicted in Figure 1. Hence, S^0 equals the input. The round key of the i^{th} round is called K^i. As the AES encryption processes the input in blocks of 4 times 4 bytes, the bytes of the i^{th} round key and of the intermediate State i are written as $K^i_{l,m}$ and $S^i_{l,m}$; l, $m \in \{0, \ldots, 3\}$. The e^{th} bit of a byte is denoted by $[\cdot]_e$.

2.2 AES

The Advanced Encryption Standard (AES) defines a symmetric block cipher. It uses a block size of 128 bit and includes key sizes of 128, 192 and 256 bit. This

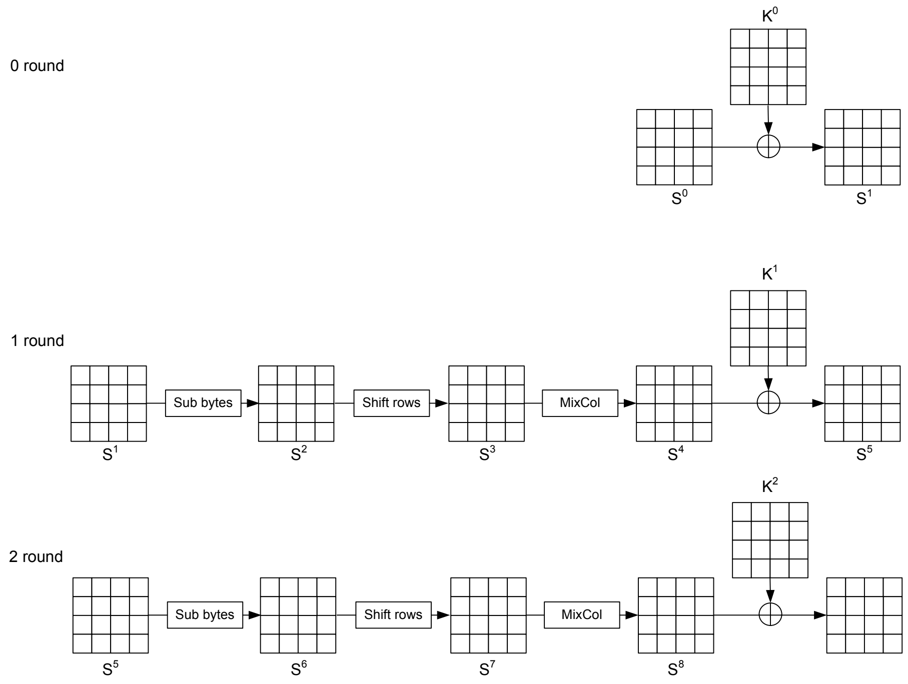


Fig. 1. AES process

paper uses the AES with a key size of 128 bit, denoted AES-128, which is briefly described here. A complete specification of the algorithm can be found in [7]. One encryption performs ten rounds. The first nine rounds consist of four operations in the following order:

- `SubBytes`: Applies a non-linear byte substitution to every byte of the block. This is denoted by $S^{k+1}_{i,j} = \mathrm{Sbox}(S^k_{i,j})$.
- `ShiftRows`: Shifts the rows 1, 2, and 3 of the State cyclically 1, 2, and 3 bytes to the left.
- `MixColumns`: Applies a linear operation on each column of the State. This can be written as a matrix multiplication for each column $i \in \{0, \ldots, 3\}$ of the State S^k:

$$\begin{pmatrix} S^{k+1}_{0,i} \\ S^{k+1}_{1,i} \\ S^{k+1}_{2,i} \\ S^{k+1}_{3,i} \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \cdot \begin{pmatrix} S^k_{0,i} \\ S^k_{1,i} \\ S^k_{2,i} \\ S^k_{3,i} \end{pmatrix}$$

- `AddRoundKey`: Xors the block with a 128-bit round key, which is derived from the initial key by a key expansion algorithm.

Three operations are performed in the tenth and last round: `SubBytes`, `ShiftRows` and `AddRoundKey`. `MixColumns` is skipped. At the beginning of an encryption an `AddRoundKey` operation is applied.

3 Attack Model

Our attack model is motivated by the possibilities of probing an internal data bus or a memory cell. Thereby, a probe is placed onto the chip and retrieves the value of the bus or the cell during all following operations until the probe is removed. Thus, the attack model assumes that an adversary can gain one bit e of the static memory of a device within the AES State throughout the whole computation, either by spying directly on it or by observing it on the bus. However, the knowledge of the bits for the first two rounds is sufficient for the attack. Inputs are chosen by the adversary. In the previous introduced notation, the adversary chooses different S^0 and learns $\{[S^i_{l_p,m_p}]_e|, i \in \{0,\ldots,40\}\}$ for a fixed position $((l_p, m_p), e)$, $l_p, m_p \in \{0,\ldots,3\}$, $e \in \{0,\ldots,7\}$.

4 Probing AES

In order to present the attack in a comprehensible way, we assume throughout this section that an adversary can probe the first byte of the AES State at the e^{th} position. Nevertheless, the method can be applied for every probe position within the State in the same way.

As the input is loaded into the memory during the initialization phase, an adversary can easily determine the position of the probe within the AES State by varying this input. Hence, an adversary knows the position of the probe.

For performing an attack, the four steps described in this section can be used. All steps were successfully simulated for numerous different keys and all possible positions of the probe. The given estimations of needed encryptions are based on these simulations. For showing the effort of each step, the needed encryptions are given separately. Nevertheless, the inputs can be chosen in a way that the results can be used for several steps, which reduces the needed ciphertexts as shown in Section 5.

In the considered case, the first step derives $K_{0,0}$, the second one $K^0_{1,1}$, $K^0_{2,2}$ and $K^0_{3,3}$. Afterwards, step three is performed three times. Thereby, the key bytes $K^0_{1,3}$, $K^0_{2,1}$ and $K^0_{3,0}$ are disclosed. Three executions of the fourth step finally reveal the remaining bytes of the secret key.

Main Idea. The attack uses the only non-linear part of the AES algorithm: the Sbox. For known bytes A and C, the relation $C = \text{Sbox}(A \oplus B)$ defines B, which can be easily found by exhaustive search over all possible 256 bytes. If only one bit of C is known, 128 possibilities for B are left using one equation. Thus, on average nine pairs of a known A and one bit of C will lead to a unique B. This technique is applied on different stages. However, the main principle is always changing the values the SubBytes operation is applied to and examining, which key byte can cause the observed bits. Due to the diffusion properties of the AES algorithm, every byte influences the probed bit somehow after the second round.

Step 1. First, the byte of the initial round key, corresponding to the probed byte, is calculated. One bit of the key is directly visible after the first AddRoundKey operation, because it is probed xor the input. After the first SubBytes operation, the probed bit $[S^2_{0,0}]_e$ fulfills

$$[S^2_{0,0}]_e = [\text{Sbox } (K^0_{0,0} \oplus S^0_{0,0})]_e. \tag{1}$$

Using on average 9 different inputs, only one possibility for the key byte $K^0_{0,0}$ is left.

Algorithm 1

1. Set up a list $\mathcal{L}$ containing all 2^8 candidates for $K^0_{0,0}$.
2. Choose a random input S^0, encrypt it and remove all candidates from $\mathcal{L}$ contradicting (1).
3. While $\mathcal{L}$ contains more than one candidate, repeat line 2.

Step 2. Next, the situation after the first MixColumns operation is considered. For the probed bit $[S^4_{0,0}]_e$ holds

$$[S^4_{0,0}]_e = [02 \cdot S^3_{0,0} \oplus 03 \cdot S^3_{1,0} \oplus 01 \cdot S^3_{2,0} \oplus 01 \cdot S^3_{3,0}]_e \tag{2}$$
$$S^3_{i,0} = \text{Sbox } (S^1_{i,i}) = \text{Sbox } (S^0_{i,i} \oplus K^0_{i,i}) \quad i \in \{0, \ldots, 3\}.$$

Varying only the input byte $S^0_{i,i}$, leaving the rest of S^0 untouched, results in a change of $S^3_{i,0}$ in (2), while all other values are constant. Thus, two different inputs S^0 and $\tilde{S}^0$, which differ only in $S^0_{i,i}$, deliver

$$\begin{aligned}[S^4_{0,0} \oplus \tilde{S}^4_{0,0}]_e &= [c_i \cdot (S^3_{i,0} \oplus \tilde{S}^3_{i,0})]_e \\ &= [c_i \cdot (\text{Sbox } (S^0_{i,i} \oplus K^0_{i,i}) \oplus \text{Sbox } (\tilde{S}^0_{i,i} \oplus K^0_{i,i}))]_e,\end{aligned} \tag{3}$$

where $c_i \in \{01, 02, 03\}$ denotes the appropriate MixColumns constant. For every pair $(S^0, \tilde{S^0})$ the equation (3) excludes wrong key bytes. Using about 10 inputs provide the right key byte. With this technique the key bytes $K^0_{1,1}$, $K^0_{2,2}$ and $K^0_{3,3}$ are determined. The following algorithm exemplary shows how to retrieve $K^0_{1,1}$.

Algorithm 2

1. Choose a random plaintext S^0.
2. Set up a list $\mathcal{L}$ containing all 2^8 candidates for $K^0_{1,1}$; $\mathcal{M}$ is an empty list.
3. Set $S^0_{1,3}$ to a random value, encrypt the new S^0 and add $(S^0_{1,3}, [S^4_{0,0}]_e)$ to $\mathcal{M}$.
4. For each pair $[(S^0_{1,3}, [S^4_{0,0}]_e), (\tilde{S}^0_{1,3}, [\tilde{S}^4_{0,0}]_e)]$ in $\mathcal{M}$ remove all candidates of $\mathcal{L}$ contradicting (3) with $c_i = 03$.
5. If $\mathcal{L}$ contains more than one candidate, go back to to line 3.

Step 3. With step three it is possible to determine an arbitrary key byte. However, the method requires a lot of different encryptions. Thus, it is only applied for a few bytes. The remaining bytes are revealed by step four to reduce effort and encryptions needed.

In order to demonstrate the method in an comprehensible way, it is assumed that the key byte $K^0_{1,3}$ should be determined. For retrieving another byte, the `MixColumns` constants have to be adjusted.

First, all input bytes except the byte $S^0_{1,3}$ are fixed. Let $E = 01 \cdot S^3_{0,2} \oplus 02 \cdot S^3_{2,2} \oplus 03 \cdot S^3_{3,2} \oplus K^1_{2,2}$ and $C = 02 \cdot S^7_{0,0} \oplus 03 \cdot S^7_{1,0} \oplus 01 \cdot S^7_{3,0}$ denote the constant parts in the computation. Thus, for the probed bit $[S^8_{0,0}]_e$ holds

$$\begin{aligned}
[S^8_{0,0}]_e &= [02 \cdot S^7_{0,0} \oplus 03 \cdot S^7_{1,0} \oplus 01 \cdot S^7_{2,0} \oplus 01 \cdot S^7_{3,0}]_e \\
&= [C \oplus 01 \cdot S^6_{2,2}]_e \\
&= [C \oplus 01 \cdot \mathrm{Sbox}\,(S^4_{2,2} \oplus K^1_{2,2})]_e \\
&= [C \oplus 01 \cdot \mathrm{Sbox}\,(01 \cdot S^3_{0,2} \oplus 01 \cdot S^3_{1,2} \oplus 02 \cdot S^3_{2,2} \oplus 03 \cdot S^3_{3,2} \oplus K^1_{2,2})]_e \\
&= [C \oplus 01 \cdot \mathrm{Sbox}\,(01 \cdot S^3_{1,2} \oplus E)]_e \\
&= [C \oplus 01 \cdot \mathrm{Sbox}\,(01 \cdot \mathrm{Sbox}\,(S^0_{1,3} \oplus K^0_{1,3}) \oplus E)]_e . \qquad (4)
\end{aligned}$$

As one bit is probed, only one bit of C has to be guessed. After about 22 different encryptions the probed bits form a pattern, which fits only to the right key byte $K^0_{1,3}$ and the right guess for E.

Algorithm 3

1. Choose a random input S^0.
2. Set up a list $\mathcal{L}$ containing all possible pairs $(K^0_{1,3}, E, [C]_e)$ for E, $K^0_{1,3} \in \{0, \ldots, 255\}$ and $[C]_e \in \{0, 1\}$.
3. Set $S^0_{1,3}$ to a random value, encrypt the new S^0 and remove all pairs contradicting (4) from $\mathcal{L}$.
4. While $\mathcal{L}$ contains more than one pair, repeat line 3.

Step 4. After the application of the last step, the byte $K^0_{1,3}$ is known. In order to improve the comprehensibility of the applied method, the explanation of step four is presented for retrieving $K^0_{0,2}$. However, with the key byte $K^0_{1,3}$ a whole column and with the bytes revealed by step three, all remaining key bytes are disclosed.

Therefore, the byte after the second `MixColumns` operation is reconsidered. The constant parts are denoted by E and C as in the previous step:

$$[S^8_{0,0}]_e = [01 \cdot \mathrm{Sbox}\,(E \oplus 01 \cdot \mathrm{Sbox}\,(S^0_{1,3} \oplus K^0_{1,3})) \oplus C]_e .$$

Varying the input byte $S^0_{1,3}$ to $\tilde{S}^0_{1,3}$ influences $S^2_{1,3}$. As $K^0_{1,3}$ is known, the new value of $S^2_{1,3}$, called $\tilde{S}^2_{1,3}$, is known. This results in a change of $S^4_{2,2}$, which is denoted by $F = S^4_{2,2} \oplus \tilde{S}^4_{2,2}$. Thus, for the probed bit of the old value $S^8_{0,0}$ and the new one $\tilde{S}^8_{0,0}$ holds

$$\begin{aligned}
[S^8_{0,0} \oplus \bar{S}^8_{0,0}]_e &= [01 \cdot \{\text{Sbox}\ (01 \cdot \text{Sbox}\ (S^0_{1,3} \oplus K^0_{1,3}) \oplus E) \\
&\quad \oplus \text{Sbox}\ (01 \cdot \text{Sbox}\ (\tilde{S}^0_{1,3} \oplus K^0_{1,3}) \oplus E)\}]_e \\
&= [01 \cdot \{\text{Sbox}\ (01 \cdot \text{Sbox}\ (S^0_{1,3} \oplus K^0_{1,3}) \oplus E) \\
&\quad \oplus \text{Sbox}\ (01 \cdot \text{Sbox}\ (S^0_{1,3} \oplus K^0_{1,3}) \oplus E \oplus F)\}]_e. \qquad (5)
\end{aligned}$$

All values in (5) except E are known. The value of E is determined using on average 8 encryptions by considering all possible combinations of values F and the corresponding probed bits.

Using the knowledge of E, the key bytes of the column are retrieved. For determining $K^0_{0,2}$, the corresponding input byte $S^0_{0,2}$ is changed to $\tilde{S}^0_{0,2}$ and the corresponding $\tilde{E}$ is determined. For E and $\tilde{E}$ holds:

$$\begin{aligned}
E \oplus \tilde{E} &= 01 \cdot \{S^3_{0,2} \oplus \tilde{S}^3_{0,2}\} \\
&= [01 \cdot \{\text{Sbox}\ (S^0_{0,2} \oplus K^0_{0,2}) \oplus \text{Sbox}\ (\tilde{S}^0_{0,2} \oplus K^0_{0,2})\}]_e. \qquad (6)
\end{aligned}$$

On average, three different constants will disclose the right key byte. In this way, the three key bytes of a column, remaining unknown after the third step, are calculated with 7 different values E. As one value E has been determined in step three, about 48 encryptions are needed for the whole column.

Algorithm 4

```
1. Chose a random input $S^0$.
2. Set up a list $\mathcal{L}$ containing all $2^8$ candidates for $K^0_{0,2}$; $\mathcal{M}$ is an empty list.
3. Set up a list $\mathcal{E}$ containing all $2^8$ candidates for $E$; $\mathcal{F}$ is an empty list.
   (a) Set $S^0_{1,3}$ to a random value, encrypt the new $S^0$ and add $(S^0_{1,3}, [S^4_{0,0}]_e)$
       to $\mathcal{F}$.
   (b) For each pair in $[(S^0_{1,3}, [S^4_{0,0}]_e), (\tilde{S}^0_{1,3}, [\tilde{S}^4_{0,0}]_e)]$ $\mathcal{F}$ remove all candidates
       from $\mathcal{E}$ contradicting (5).
   (c) If $\mathcal{E}$ contains more than one candidate, go back to to line 3a.
4. Add the remaining value of $\mathcal{E}$ together with the first $S^0_{1,3}$ of $\mathcal{F}$ to $\mathcal{M}$.
5. For each pair $(S^0_{1,3}, [S^4_{0,0}]_e)$and $(\tilde{S}^0_{1,3}, [\tilde{S}^4_{0,0}]_e)$ in $\mathcal{M}$ remove all candidates in
   $\mathcal{L}$ contradicting (6).
6. If $\mathcal{L}$ contains more than one candidate, go back to to line 3.
```

Summarization. Summing up all four steps, probing one bit of 250 encryptions on average is sufficient to determine a whole 128-bit AES key. Our simulation results showed that this neither depends on the probed byte within the State nor on the bit of the byte which is probed. A simple possibility to reduce the number of needed encryptions is to choose the plaintexts in a way, that they can be used by several steps. The messages used for step one can be reused in step two, if the whole column is changed. The difference in $S^3_{0,0}$ does not influenced step two, as $K^0_{0,0}$ and therefore $S^3_{0,0}$ is known from the previous step. As $S^4_{0,0}$ is known after step two, $K^1_{0,0}$ can be determined using the same method as in step one in the second round. Again, the probed bits from previous encryptions can be used. Now, $S^7_{0,0}$ can be calculated. Thus, the inputs can be chosen in a way that they can be used for step one, two, and three at the same time. Hence, 210 encryptions are sufficient in this way.

5 Variants

Depending on the probed byte, optimizations are possible. The following two methods allow reducing key bytes that have to be determined by step three. Therefore, the computational effort is reduced. In addition, the number of needed encryptions is decreased. It is also possible to attack AES successfully in the known plaintext scenario as shown by the third method.

5.1 Probing $S^i_{0,0}$

By probing $S^i_{0,0}$, step three can be skipped. Hence, step four must determine an additional E value for each column. Thus, the number of encryptions is reduced by $(22-8)*3 = 42$, resulting in 168 required encryptions, as the inputs for step one and two can also be used for the fourth.

In order to omit step three, the key schedule is used. After the second step, the whole column $S^3_{i,0}$, $i \in \{0\dots3\}$ and thus $S^4_{i,0}$ is disclosed. As this includes $S^4_{0,0}$, the first step is applied to the second round. This determines $K^1_{0,0}$, using the values for $S^6_{0,0}$, gained during the previous encryptions. In combination with $K^0_{0,0}$, another key byte of the first round is calculated using the key schedule, while 1 is its round constant *rcon*, see [7]:

$$K^1_{0,0} = K^0_{0,0} \oplus \text{Sbox}\ (K^0_{1,3}) \oplus 1$$
$$\Rightarrow K^0_{1,3} = \text{Sbox}\ ^{-1}((K^0_{0,0} \oplus K^1_{0,0} \oplus 1)).$$

After determining the column $S^3_{i,2}$, $i \in \{0,\dots,3\}$ using $K^0_{1,3}$ in step four, the values of $S^4_{i,2}$, $i \in \{0,\dots,3\}$ are known. Together with $E = S^4_{2,2} \oplus K^1_{2,2}$, gained in the fourth step, this leads to $K^1_{2,2}$. Thus, $K^0_{2,1}$ is calculated:

$$\begin{aligned} K^1_{2,1} &= K^0_{2,2} \oplus K^1_{2,2} \\ K^1_{2,0} &= \text{Sbox}\ (K^0_{3,3}) \oplus K^0_{2,0} \\ K^0_{2,1} &= K^1_{2,1} \oplus K^1_{2,0}. \end{aligned} \tag{7}$$

Repeating step four using $K^0_{2,1}$, followed by a similar use of the key schedule as in (7), leads to another 4 key bytes, including $K^0_{0,3}$. Thus, applying step four again will determine the last 3 key bytes. Summing up, 168 encryptions are necessary on average.

5.2 Probing Bytes of Rows 1-3

The `ShiftRows` operation can be used to improve the attack. If the `ShiftRows` operation is implicitly implemented by choosing the appropriate positions as input for the `MixColumns` operation, the result of `ShiftRows` is not explicitly written into the memory. Thus, it is not possible to probe a new value from S^3 and step two has to be applied for all four bytes in the column.

Otherwise, if the result of the `ShiftRows` operation is written into memory, step one is applied to State S^3 and determines the key byte of the value shifted to the position of the probe. Consider probing $S_{1,0}$, for example. Using step one, the key byte $K^0_{1,0}$ is revealed. After the first `ShiftRows` operation, a $S^2_{1,1}$ is probed. Thus, applying

$$[S^3_{1,0}]_e = [\text{Sbox}\ (K^0_{1,1} \oplus S^0_{1,1})]_e$$

on the ciphertexts of the first step will deliver two key bytes at the same time. Hence, one application of step three can be omitted which reduces the needed effort by 22 ciphertexts.

5.3 Known Plaintext Scenario

If the input cannot be chosen, three bits information about the State of the first round is necessary. Therefor, the probes are placed in a way that each of them can determine five different key bytes. This is fulfilled by the positions $(S_{1,i}, S_{2,i+1 \bmod 4}, S_{3,i+2 \bmod 4})$, $i \in \{0 \dots 3\}$. The first step can be applied to the probed bytes as in the chosen plaintext scenario.

As parts of the plaintext cannot be chosen, it is necessary to guess more than one byte at the same time in step two. The whole input and hence the column $j \in \{0, \dots, 3\}$: $S^3_{i,j}$ $i \in \{1, \dots, 3\}$ is varied in most cases. We illustrate the method for a probe at position $S_{1,1}$. Modifying (2), three bytes can be guessed at the same time:

$$[S^4_{1,1}]_e = [01 \cdot S^3_{0,1} \oplus 02 \cdot S^3_{2,1} \oplus 03 \cdot S^3_{2,1} \oplus 01 \cdot S^3_{3,1}]_e$$
$$S^3_{i,j} = \text{Sbox}\ (S^1_{i,l}) = \text{Sbox}\ (S^0_{i,l} \oplus K^0_{i,l}); k = j + i \bmod 4, i \in \{0, \dots, 3\}.$$

As the key bytes $K^0_{1,1}$ and $K^0_{1,2}$ were determined, as described in the previous subsection, using the first step and the `ShiftRows` operation, $S^3_{1,1}$ is known. Hence, 2^{24} combinations for the key bytes of the column are left. Using about 26 encryptions, the right bytes can be revealed. Applying this procedure for all probes leaves only $K^1_{0,i+3 \bmod 4}$ unknown, which can be easily guessed.

Concluding, for the known plaintext case, probing three positions of about 26 encryptions is necessary.

6 Conclusion

In this paper, we presented a probing attack on AES. For the attack, probing an arbitrary bit of the AES State is sufficient. As demonstrated, a 128-bit key can be determined by spying on the first two rounds of about 210 encryptions of chosen plaintexts. The number of encryptions can be reduced, by applying optimizations that depend on the probed byte, down to 168. We also considered the known plaintext scenario and showed that three probes and 26 ciphertexts are sufficient in this case.

Acknowledgements

The authors like to thank Stefan Tillich for helpful discussions. The information in this document reflects only the authors views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. 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)
2. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
3. 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)
4. Kömmerling, O., Kuhn, M.G.: Design Principles for Tamper-Resistant Smartcard Processors. In: USENIX Workshop on Smartcard Technology (Smartcard 1999), pp. 9–20 (May 1999)
5. Skorobogatov, S.P.: Semi-invasive attacks - A new approach to hardware security analysis. PhD thesis, University of Cambridge - Computer Laboratory (2005), http://www.cl.cam.ac.uk/TechReports/
6. Handschuh, H., Paillier, P., Stern, J.: Probing Attacks on Tamper-Resistant Devices. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 303–315. Springer, Heidelberg (1999)
7. National Institute of Standards and Technology (NIST): FIPS-197: Advanced Encryption Standard (November 2001), http://www.itl.nist.gov/fipspubs/

On Avoiding ZVP-Attacks Using Isogeny Volcanoes*

J. Miret[1], D. Sadornil[2], J. Tena[3], R. Tomàs[1], and M. Valls[1]

[1] Dept. de Matemàtica, Universitat de Lleida
{miret,rosana,magda}@eps.udl.es
[2] Dept. de Matemáticas, Estadística y Computación, Universidad de Cantabria
sadornild@unican.es
[3] Dept. de Álgebra, Geometría y Topología, Universidad de Valladolid
tena@agt.uva.es

Abstract. The usage of elliptic curve cryptography in smart cards has been shown to be efficient although, when considering curves, one should take care about their vulnerability against the Zero-Value Point Attacks (ZVP). In this paper, we present a new procedure to find elliptic curves which are resistant against these attacks. This algorithm finds, in an efficient way, a secure curve by means of volcanoes of isogenies. Moreover, we can deal with one more security condition than Akishita-Takagi method with our search.

1 Introduction

Smart card technologies are nowadays present in many activities in daily life, such as mobile communications, credit cards, identification systems, etc. Consequently, the usage of cryptographic techniques in smart cards has been a widely studied area of research during the last decade. Smart cards are devices which can basically execute commands and store data. This stored data can be protected using cryptographic algorithms, which must fit the memory and computational restrictions of these cards. Hence, the usage of elliptic curve cryptography turns out to be a good alternative, since it can offer the same security as conventional cryptosystems while using significantly shorter keys.

Apart from attacks to the cryptosystems, smart cards are also vulnerable under the so–called *Side Channel Attacks* (SCA) [9,11]. The attacker *listens* to the card while it is encrypting (for instance, using an oscilloscope). From its behavior, information about the performed computations can be obtained, from which the secret keys may be obtained. There are several proposals in the literature presenting methods to prevent these attacks.

In 2003, Goubin [8] detected an SCA for elliptic curve cryptography. It was shown there could be some points in an elliptic curve that, when used, could provide sensitive information to obtain the secret key of smart card. Goubin proved

* Partially supported by grants MTM2007-66842-C02-01, MTM2007-66842-C02-02 and TIN2006-15662-C02-02 from Spanish MCyT.

that, in order to avoid the existence of such points, the curve should fulfill some conditions. In the case a curve could not fulfill them, a new curve should be taken into account, while maintaining the same security levels (namely, same cardinal).

A first approach would consist of taking isomorphic curves [5,10]. But it can be shown that Goubin's conditions would not be fulfilled either. Later, Smart [18] introduced another countermeasure: the usage of isogenies of a given elliptic curve. The same cardinal would also be preserved, as well as some isogenous curve could fit Goubin's restrictions.

Akishita and Takagi [1,2] extended Goubin's work, considering that there were more points in the curve that could be exploited by an attacker (Zero-Value Point Attack, ZVP). Hence, they provided more conditions to be fulfilled by the curve. In their paper, they also look for curves isogenous to the ones proposed by SECG [17], and such that satisfy some of the restrictions (some others are not considered, since they seem difficult to be treated).

Both Smart's and Akishita-Takagi's procedures analyze if one of the adjacent ℓ–isogenies of the original curve fits the restrictions. Otherwise, the prime ℓ is incremented and a new curve is searched for. The more conditions we require to be satisfied, the higher is the value of ℓ. Since the computational cost increases rapidly when ℓ is high, these procedures turn out to be costly.

In this work, we present an alternative procedure which consists of analyzing more than one isogeny (in fact, as many as possible) before incrementing the value of ℓ. Consequently, this algorithm presents a lower computational cost and obtains results in less time. Besides, one more Akishita-Takagi's condition can be evaluated, so the curves obtained are more resistant to ZVP-attacks.

The rest of this paper is organized as follows. In Section 2, we review preliminary concepts on elliptic curve cryptography and isogenies of a given elliptic curve. Section 3 summarizes several results on SCA and ZVP attacks. Section 4 presents the new algorithm, analyzes its results and compares them to previous proposals. Finally, conclusions are exposed in Section 4.

2 Preliminaries on Elliptic Curves

Let E be an elliptic curve over a finite field $\mathbb{F}_p$, where p is a prime greater than 3, defined by a Weierstraß form:

$$E/\mathbb{F}_p : y^2 = x^3 + ax + b, \tag{1}$$

where a, $b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \neq 0$. We denote by $E(\mathbb{F}_p)$ the group of points $P = (x, y)$, x, $y \in \mathbb{F}_p$, that satisfy this equation, including the point at infinity, denoted as $\mathcal{O}_E$.

A point $P = (x, y) \in \mathbb{F}_p^2$ can be represented by means of Jacobian coordinates, $(X : Y : Z)$, where $x = X/Z^2$ and $y = Y/Z^3$. Using such a system of coordinates, $(X : Y : Z)$ and $(X' : Y' : Z')$ represent the same point if there is an element $r \in \mathbb{F}_p^*$ such that $X' = r^2 \cdot X$, $Y' = r^3 \cdot Y$ and $Z' = r \cdot Z$. Therefore, the Weierstraß equation (1) becomes:

$$E/\mathbb{F}_p : Y^2 = X^3 + aXZ^4 + bZ^6. \tag{2}$$

In elliptic curve cryptography it is usual to multiply a point $P \in E(\mathbb{F}_p)$ by a scalar $d \in \mathbb{F}_p$ to obtain the point dP. To compute this multiplication in an efficient way the standard methods involve two basic operations: adding and doubling points.

On the one hand, given a point $P = (X_1 : Y_1 : Z_1)$ of $E(\mathbb{F}_p)$, the coordinates of $2P = (X_2 : Y_2 : Z_2)$ can be expressed as follows:

$$X_2 = T, \quad Y_2 = -8Y_1^4 + M(S - T), \quad Z_2 = 2Y_1Z_1, \tag{3}$$

where $S = 4X_1Y_1^2$, $M = 3X_1^2 + aZ_1^4$ and $T = -2S + M^2$. On the other hand, given two points $P_1 = (X_1 : Y_1 : Z_1)$ and $P_2 = (X_2 : Y_2 : Z_2)$ of $E(\mathbb{F}_p)$, the coordinates of $P_1 + P_2 = (X_3 : Y_3 : Z_3)$ can be expressed as:

$$X_3 = -H^3 - 2U_1H^2 + R^2, \; Y_3 = -S_1H^3 + R(U_1H^2 - X_3), \; Z_3 = Z_1Z_2H, \tag{4}$$

where $U_1 = X_1Z_2^2$, $U_2 = X_2Z_1^2$, $S_1 = Y_1Z_2^3$, $S_2 = Y_2Z_1^3$, $H = U_2 - U_1$ and $R = S_2 - S_1$.

2.1 Isogenies

Given a subgroup $G \subseteq E(\mathbb{F}_p)$, for instance the cyclic subgroup $\langle P \rangle$ generated by a point $P \in E(\mathbb{F}_p)$, a rational map $\mathcal{I}$ can be constructed from the curve E with kernel G. Then the quotient E/G is a new elliptic curve E', which is called *isogenous curve* of E under isogeny $\mathcal{I}$. In general, given two elliptic curves, E and E', it is said that they are isogenous curves if there exists a rational map $\mathcal{I}$ between them that sends the point at infinity of E to this one of E'. Besides, the degree d of the isogeny $\mathcal{I}$ is the degree of the corresponding extension of function fields. If this extension is separable the degree d coincides with the cardinal of the subgroup $\ker \mathcal{I}$. Then, we can say that the isogeny $\mathcal{I}$ is an isogeny of degree d or a *d–isogeny*. In the case that d is not a prime number and its prime factors are ℓ_1,...,ℓ_n, the isogeny $\mathcal{I}$ can be expressed as a composition of n isogenies of degrees ℓ_i.

More precisely, given an elliptic curve of equation

$$E/\mathbb{F}_p : \; y^2 = x^3 + ax + b, \tag{5}$$

the coefficients of its isogenous curve of kernel G

$$E'/\mathbb{F}_p : \; y^2 = x^3 + a'x + b', \tag{6}$$

can be straightforwardly obtained by means of Vélu formulae [19]:

$$a' = a - 5t, \quad b' = b - 7w,$$

with

$$t = \sum_{T \in \mathcal{S}_G} t(T), \quad w = \sum_{T \in \mathcal{S}_G} \left(u(T) + x(T)t(T) \right),$$

being S_G a system of representatives of the orbits of G under the action of the subgroup $\{-1,1\}$, $x(T)$ is the abscissa of the point T and

$$t(T) = \begin{cases} 3x(T)^2 + a, & \text{if } T \in G \cap E[2] \\ 6x(T)^2 + 2a, & \text{if } T \in G \setminus E[2] \end{cases}$$

$$u(T) = 4x(T)^3 + 4ax(T) + 4b.$$

It is well known that, for a prime integer ℓ, the number of ℓ–isogenies that an elliptic curve $E/\mathbb{F}_p$ can have is 0, 1, 2 or $\ell + 1$ [3].

2.2 Isogeny Volcanoes and Cordilleras

Given an ordinary elliptic curve $E/\mathbb{F}_p$ and an ℓ–isogeny $\mathcal{I} : E \longrightarrow E'$, Kohel [12] introduced the notion of direction of the isogeny, according to the relation between the endomorphism rings $\mathcal{O}$ and $\mathcal{O}'$ of the curves. Actually, Kohel shows that $[\mathcal{O} : \mathcal{O}'] = 1$, ℓ or $1/\ell$, and depending on each case, it is said that the isogeny $\mathcal{I}$ is *horizontal, descending* or *ascending*, respectively. This notion of direction can be exploited to represent isogenous curves by means of graph structures.

An ℓ–volcano (see [6]) is a directed graph whose nodes are isomorphism classes of elliptic curves and whose edges represent ℓ–isogenies among them. These graphs consist of a unique cycle at the top level, called *crater*, and from each node of the cycle hang $\ell - 1$ trees which are ℓ–ary complete, except in the case where the volcano is reduced to the crater. The leaves of these trees are located at the same level, which form what is called the *floor* of the ℓ–volcano, while the remaining nodes of each tree constitute the volcano side. Each node of the ℓ–volcano (except the leaves) has $\ell + 1$ edges. More precisely, nodes in the volcano side have one ascending isogeny and ℓ descending ones, while nodes on the crater have two horizontal isogenies and $\ell - 1$ descending ones. The structure of an ℓ–volcano for $\ell = 3$ is given in Figure 1.

Given an elliptic curve $E/\mathbb{F}_p$, its volcano of ℓ–isogenies will be denoted by $V_\ell(E/\mathbb{F}_p)$. Then , for a given prime ℓ, the elliptic curves over a finite field $\mathbb{F}_p$ with the same cardinal can be distributed in several ℓ–volcanoes, the set of all these connex components will be named *ℓ–cordillera.*

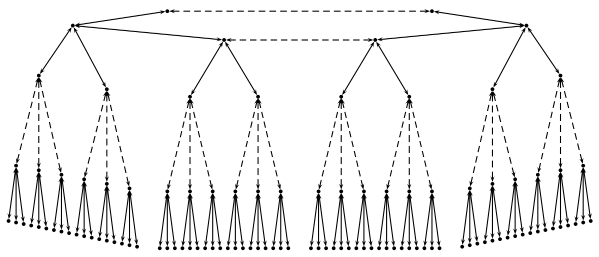


Fig. 1. Structure of a 3–volcano

The height of the volcano $V_\ell(E/\mathbb{F}_p)$ can be obtained considering the conductor f of the order $\mathbb{Z}[\pi]$, being π the endomorphism of Frobenius of $E/\mathbb{F}_p$. More precisely, it can be deduced that the height of the volcano of $E/\mathbb{F}_p$ coincides with the ℓ–adic valuation of the integer f, that is $h(V_\ell(E/\mathbb{F}_p)) = v_\ell(f)$. Nevertheless, there are efficient algorithms to determine the height of a volcano which do not need to obtain f (see [6,15]).

Concerning the study of the connex components of an ℓ–cordillera, we can take advantage of the following result [16]:

i) All connex components of an ℓ–cordillera of elliptic curves have the same height.
ii) Elliptic curves which are in different levels of an ℓ–volcano must belong to different connex components of any ℓ'–cordillera, when $\ell' \neq \ell$.

3 *Side Channel Attacks* in Smart Cards

Cryptographic techniques used in smart cards to protect its secret information have several vulnerabilities. One of them is due to the leak of information during the execution of the cryptosystem, which could be used to recover the secret key. This kind of attacks are known as *Side Channel Attacks* (SCA) [9].

In other words, cryptanalytic attacks look at the plaintext and the ciphertext and attempt to recover the key or some piece of information. Moreover, SCA also pretend to use some extra knowledge, such as the power consumption changes while the smart card is working. To be successful, an SCA–attacker requires considerable technical knowledge of the internal operations of the system on which the cryptography is implemented.

There are three main side channels to be exploited in these attacks: power consumption, timing consumption and radiation emissions, which are known as Power Analysis Attacks, Timing Attacks and Electromagnetic Attacks, respectively.

Power Analysis Attacks use information leaked by the power consumption of the smart card. This type of attack can be divided into two subtypes: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). On one hand, SPA extract secret keys and compromise the security of smart cards and other cryptographic devices by analyzing their power consumption. While these kind of attacks do not require statistical analysis, DPA uses statistical techniques to separate signal from noise. Timing Attacks are based upon the principle of dependence between the execution time of the cryptosystems and the value of the secret data. By measuring and analyzing small differences in processing time, an attacker can inffer secret data. Finally, Electromagnetic Attacks use the information leaked in the electromagnetic emanations from the card while it is running.

3.1 Zero–Value Point Attacks

The use of elliptic curves cryptography in smart cards revealed a new type of vulnerability not exploitable since that moment. Goubin [8] proposed a method to take advantage of this vulnerability: he showed that an attacker might generate points on the curve so that, after several calculations, a new point was obtained with some coordinate being zero. In that case, Goubin proved that the attacker would obtain information from the secret key of the card, and after several executions, all bits of the key could be obtained. These points with some null coordinate were called *special points*. It can be seen that a curve (2) where b is not a quadratic residue has no points with zero–valued abscissa. Whereas, curves with no points of order 2 have no zero–valued ordinate points. In this context, as a countermeasure, Smart [18] proposed to avoid these special points by searching for isogenous curves with these conditions.

Later, Akishita and Takagi [1,2] showed that this kind of attacks could be extended in the case that the doubling or adding processes involved some expressions equal to zero. This attack is known as *Zero–Value Point Attack* (ZVP attack). Akishita and Takagi deduced some extra conditions. These conditions include the ones given by Goubin and all of them are deduced from the expressions which appear when doubling (3) and adding (4) points.

A point $P = (x, y)$ on an elliptic curve $E/\mathbb{F}_p$ would be a ZVP with respect to the doubling process if some of these conditions is satisfied [1]:

ED1: $3x^2 + a = 0$;
ED2: $5x^4 + 2ax^2 - 4bx + a^2 = 0$;
ED3: The order of P is equal to 3;
ED4: $x(P) = 0$ or $x(2P) = 0$, i. e., b is a quadratic residue;
ED5: $y(P) = 0$ or $y(2P) = 0$, i. e., the order of P is equal to 2.

Hence Akishita and Takagi look for isogenous curves where none of these conditions are satisfied (using the same idea as Smart's countermeasure). In fact, they focus on conditions ED1 and ED4 (Goubin's condition): curves whose parameter b is not a quadratic residue and such that $-a/3$ is not a quadratic residue or there are no points whose abscissa is $\sqrt{-a/3}$. In particular, they performed experimental computations [2] searching for the minimal isogeny degree needed to find a good curve of the SECG curves [17].

Notice that conditions ED3 and ED5 do not vary when taking isogenies. Moreover, in their paper, Akishita and Takagi do not deal with condition ED2 neither conditions coming from the addition process.

4 Isogenies–Route over Cordilleras

In this section, we describe the proposed algorithm to find a resistant curve against ZVP attacks. It exploits the structure of isogeny cordilleras to travel across them until a good curve is found. Several results are then collected and compared with previous methods.

4.1 Algorithm

Smart and Akishita–Takagi used isogenies as a countermeasure against ZVP attacks. Their method to find a ZVP–resistant curve consists in finding a suitable

ℓ–isogeny of a given curve $E/\mathbb{F}_p$. More in detail, let $\ell_1 < \ell_2 < ... < \ell_n$ be prime numbers so that there exist ℓ_i–isogenies of $E/\mathbb{F}_p$, $1 \leq i \leq n$. Firstly, the ℓ_1–isogenies of $E/\mathbb{F}_p$ are obtained and it is also verified if these new curves are ZVP–resistant. If they are not, one proceeds similarly with ℓ_2, until a suitable curve is found. In these methods, the first level of adjacent isogenous curves in the volcano is the only one evaluated before increasing the value of ℓ. However, increasing this value, makes the computational cost rise sharply.

In this new proposal, the fact that the rest of curves in the volcano, and cordillera, also have the same cardinal is exploited. Hence, given an elliptic curve $E/\mathbb{F}_p$, its ℓ_1–volcano of isogenies is completely analyzed. If a suitable curve is not found, we then move to the ℓ_2–volcano of $E/\mathbb{F}_p$ and compute a new curve of it. For each curve which was not previously studied, we move to its ℓ_1–volcano, before going on with the ℓ_2–volcano. The goal is to analyze firstly the less costly volcanoes. If a good curve is not found in this volcano, the ℓ_2–volcano is explored again. When this volcano is completely explored, as well as the ℓ_1–volcanoes of its curves, we move to the ℓ_3–volcano, and proceed similarly. This algorithm follows similar ideas as the one presented in [16].

The details of the algorithm are sketched below.

```
Isogeny–route over cordilleras
Input: An elliptic curve $E$ over a field $\mathbb{F}_p$; IsogenyDegrees: a
       list of $n$ prime integers for which exist isogenies of $E$
Output: An isogenous curve $E'$ ZVP–resistant

IF ZVP–resistant($E$)
  RETURN $E$
FOR $i = 0$ to $n-1$
  Pendant[$i$]$\leftarrow E$
WHILE $\exists j$, $j = 0$ to $n-1$, s. t. Pendant[$j$]$\neq \emptyset$
  $\ell$ = IsogenyDegrees[$j$]
  $E_{act}$ =FirstElement(Pendant[$j$])
  Treat[$j$]$\leftarrow E_{act}$
  Delete($E_{act}$, Pendant[$j$])
  $E_{next}$ = $\ell$–isogeny($E_{act}$)
  IF ZVP–resistant($E_{next}$)
    RETURN $E_{next}$
  ELSE ZVP–resistant($E_{next}$)
    FOR $i = 0$ to $n-1$ s. t. $i \neq j$
      Pendant[$i$]$\leftarrow E_{next}$
```

where the functions called by the algorithm are the following:

– Pendant[i]: List of elliptic curves such that non ℓ_i–isogeny is already calculated. The algorithm could deal with them hereinafter.
– IsogenyDegrees[i]: List of prime integers $\ell_1 < \ell_2 < ... < \ell_n$ corresponding to possible isogeny degrees of E.

- Treat[i]: Stores a list of treated elliptic curves found to be vulnerable to ZVP attacks.
- ZVP–resistant(E): Verifies if E fulfill the given conditions.

Notice, this algorithm must be executed in advance. Indeed, when implementing the smart card, it will already have precalculated the best isogeny–degree for its curve. Hence, the computational cost for the smart card will only compute such an isogeny.

4.2 Implementation and Complexity

Let us show some computational results of the proposed algorithm, compared with the experimental results obtained by Smart and Akishita-Takagi, respectively. The test of their algorithm is performed over the set of curves listed in the SECG's document [17], which is considered a standard in elliptic curve cryptography. In particular, we also focus on several SECG curves, and MAGMA [13] is used for all implementations. The method to compute the isogenies of a given curve is to use division polynomials.

As we have seen, Smart and Akishita-Takagi method consists of increasing ℓ until finding a resistant ℓ–isogenous curve. The method proposed in this paper (isogeny–route) uses the concatenation of ℓ–isogenies, without increasing ℓ value, until obtaining a good curve, and in the case that all curves of this ℓ–volcano of a given curve E have been treated then the ℓ value increments. The reason why this method needs less time to obtain a resistant curve is because the cost of an ℓ–isogeny increases sharply when ℓ value grows and it makes this last method faster than the other.

We assume that the behavior of the curves is independent of the degree of the isogenies to compute them, i.e., there is the same probability to find a curve that fulfills some conditions using isogenies of a small degree compared to using a large one. Considering that the cost of computing an ℓ–isogeny in $\mathbb{F}_p$ is $\mathcal{O}(\ell^2(\ln p))$ [7], it is preferable that the degree of the isogeny is the lowest possible. In this way, the isogeny–route algorithm gives us better results.

Let us see the probability that an elliptic curve fulfills the different conditions. In this paper we will focus on conditions ED1, ED2 and ED4. We can compute the probabilities of these conditions being satisfied (assuming all the conditions being independent). In the case of ED4, the probability of a curve failing for this condition is the probability that the parameter b of the equation of the elliptic curve is a quadratic residue, i.e. 1/2. The probability of a curve failing for condition ED1, $3x^2 + a = 0$, is 1/4, because there is a probability of 1/2 that exists an x fulfilling the equation and another probability of 1/2 that this value of x corresponds to a valid abscissa of the elliptic curve.

Since the treated condition in Smart's case is ED4, the probability of finding a resistant elliptic curve is 1/2. In the case of Akishita and Takagi, the conditions that they treat are ED1 and ED4, i.e., $1/4 \cdot 1/2 = 1/8$. And finally, the probability of finding a curve that does not succumb to conditions ED1, ED2 and ED4 is, approximately, 3/40. This value comes from the probability to find an elliptic curve satisfying neither ED1 nor ED4 (which is 1/8) and from the probability to obtain a curve which does not fulfill ED2 (which is, approximately, 3/5). Indeed, this last one coincides with the probability of a degree four polynomial not having linear factors or, if it has, that none of its roots corresponds to a valid abscissa of the curve.

Using these probabilities and the cost to compute an ℓ–isogeny, if we try to find a resistant curve with the two initial methods, the ℓ values and the computational cost increase sharply. Nevertheless, if we try to do the same with the isogeny–route algorithm, these prime numbers and this cost remain stable. For this reason, searching for curves that fulfill ED1, ED2 and ED4 is fast and affordable using isogeny–route algorithm.

5 Experimental Results

In Table 1 we collect results obtained by Smart together with results obtained using our algorithm. In the second and fourth column we present the minimal and preferred isogeny degree (ℓ_{std} and ℓ_{pfr}) with respect to condition ED4 given by Smart, while the third and the fifth columns contain the degrees of the isogeny–route given by our algorithm in each case. More precisely, these degrees can be defined as:

– ℓ_{std}: the minimal isogeny degree for condition ED4;
– ℓ_{prf}: the minimal isogeny degree for condition ED4 and condition $a = -3$;

The integers ℓ_{std}–route and ℓ_{prf}–route correspond to the minimal and preferred isogeny degree obtained as a set of isogeny degrees using our algorithm. Thus, concerning curve secp192r1 from SECG the preferred isogeny degree is $\ell_{\mathrm{prf}} = 73$, while ℓ_{prf}–route $= 5 - 13 - 23$, which means the composition of three isogenies of degrees 5, 13 and 23. Notice that the cost of computing this composition is less than computing an isogeny of degree 73 (Table 2).

Table 2 provides the computing and searching time to find the suitable isogeny degrees of several SECG curves (those for which the time needed by Smart's algorithm and isogeny–route's algorithm are different). Notice that we distinguish

Table 1. Minimal isogeny degrees with respect to ED4 for SECG curves

ED4	ℓ_{std}	ℓ_{std}–route	ℓ_{prf}	ℓ_{prf}–route
secp112r1	1	1	1	1
secp128r1	7	7	7	7
secp160r1	13	13	13	13
secp160r2	19	19	41	19-19
secp192r1	23	5-13	73	5-13-23
secp224r1	1	1	1	1
secp256r1	3	3	11	3-5
secp384r1	19	19	19	19
secp521r1	5	5	5	5

Table 2. Time for computing minimal isogeny degrees with respect to ED4

Calculation / Search (sec.)	ℓ_{std}	ℓ_{std}–route	ℓ_{prf}	ℓ_{prf}–route
secp160r2	18.61 / 18.61	18.61 / 18.61	267.63 / 547.01	41.8 /46.78
secp192r1	44.30 / 51.24	6.01 / 6.99	3474.7 / 4788.15	50.31 / 59.22
secp256r1	0.012 / 0.012	0.012 / 0.012	5.93 / 6.43	0.035 / 0.043

Table 3. Minimal isogeny degrees with respect to ED1+ED4 for SECG curves

ED1+ED4	ℓ_{std}	ℓ_{std}–route	ℓ_{prf}	ℓ_{prf}–route
secp112r1	7	7	-	-
secp128r1	7	7	7	7
secp160r1	13	13	13	13
secp160r2	19	19	41	19-23-23
secp192r1	23	13-13	-	-
secp224r1	1	1	1	1
secp256r1	3	3	23	5-11
secp384r1	31	31	-	-
secp521r1	5	5	5	5

between search time and calculation time, that is, the time necessary to get the degree of a good isogeny and the time to calculate the isogenous curve when its degree or its minimal route are known, respectively.

As can be seen in Table 2, we obtain better calculation and search times with isogeny–route algorithm, specially when searching for a preferred curve. The obtained speedup even surpasses the value 80 in the search of a preferred and resistant isogeny of *secp192r1*.

Moreover, in Table 3 we compare results obtained by Akishta–Takagi concerning isogenous curves and conditions ED1 and ED4 with results obtained using the isogeny–route algorithm.

As we can see in Table 4, isogeny–route algorithm returns ZVP–resistant curves with less computational cost. For instance, for the curve secp160r2, Akishita and Takagi method computes a 41–isogenous curve, whereas the algorithm proposed computes three isogenies of degrees 19–23–23. Therefore, the time of calculation, using isogeny–route algorithm, will never be higher than the time needed using another method, in the worst of cases the results will be equal.

Another advantage of using isogeny–route algorithm is the fact that this algorithm obtains more curves in less time than other methods, and it allows us to do more restrictive searches without raising the calculation time. In that manner, we can approach another condition given by Akishita and Takagi, concretely ED2, without incrementing excessively the degree of the isogenies that we will need to compute. Using isogeny–route it is possible to treat this condition because the ℓ values keep relatively small, whereas using other method the ℓ values increase until high values.

Table 4. Time for computing minimal isogeny degrees with respect to ED1+ED4

Calculation / Search (sec.)	ℓ	ℓ–route
secp160r2	267.63 / 547.01	91.8 / 146.78
secp192r1	44.30 / 51.24	11.92 / 16.35
secp256r1	97.11 / 145.87	6.07 / 6.45

Table 5. Route and time for computing minimal isogeny degrees with respect to ED1+ED2+ED4

ED1+ED2+ED4	ℓ_{std}–route	time–route
secp112r1	7	0.22
secp128r1	7	0.332
secp160r1	13	4.93
secp160r2	19	16.58
secp192r1	13-13	13.44
secp224r1	3-3	0.06
secp256r1	3	0.02
secp384r1	19-19-19-19	327.93
secp521r1	7-7	8.38

Table 5 contains results to find isogeny curves of given ones by SECG's list that fulfill ED1, ED2 and ED4.

The equations of the isogenous curves of these examples have been computed using ℓ-division polynomials [3]. They could also be computed using modular polynomials $\Phi_\ell(x, y)$, which in MAGMA [13] are stored until a certain value of ℓ [4]. In that case, the obtained times are quite smaller. Nevertheless, the results of these tables, comparing the Smart and Akishita-Takagi method with isogeny–route method and using modular polynomials, maintain the same ratio of efficiency.

6 Conclusions

The use of elliptic curve cryptography in smart cards is not free of Side Channel Attacks' vulnerabilities. To avoid them, Smart gives resistant elliptic curves against such attacks by taking isogenous curves. Both Smart and Akishita–Takagi [18,2] use ℓ–isogenies to find a curve with same cardinal that does not succumb to ZVP attack, but they only deal with the isogenies of the initial curve, they do not look for compositions of isogenies. These curves have the same properties as the initial one and they are easier to find, because it uses minor ℓ's.

Due to the use of the algorithm proposed, we can obtain more curves with less effort and, working with a more extensive pool of elliptic curves, we have more possibilities of finding a resistant curve in less time. Because of this, we could deal with all the conditions with respect to the doubling process of a point. This is possible because proposed algorithm gives us more curves without increasing degrees of isogenies. Thus, we obtain more secure curves against ZVP attacks.

References

1. Akishita, T., Takagi, T.: Zero-Value point attacks on elliptic curve cryptosystem. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 218–233. Springer, Heidelberg (2003)
2. Akishita, T., Takagi, T.: On the optimal parameter choice for elliptic curve cryptosystems using isogeny. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 346–359. Springer, Heidelberg (2004)
3. Blake, F., Seroussi, G., Smart, N.: Elliptic Curves un Criptography. London Mathematical Society Lecture Notes, vol. 256. Cambridge University Press, Cambridge (1999)
4. Charles, D., Lauter, K.: Computing modular polynomials. Journal of Computation and Mathematics. London Mathematical Society 8, 195–204 (2005)
5. Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
6. Fouquet, M., Morain, F.: Isogeny volcanoes and the SEA algorithm. In: Fieker, C., Kohel, D.R. (eds.) ANTS 2002. LNCS, vol. 2369, pp. 276–291. Springer, Heidelberg (2002)
7. Galbraith, S.: Constructing isogenies between elliptic curves over finite fields. Journal of Computational Mathematics 2, 118–138 (1999)
8. Goubin, L.: A refined power-analysis attack on elliptic curve cryptosystems. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 199–211. Springer, Heidelberg (2002)
9. Joye, M.: Elliptic curves and side-channel analysis. ST Journal of System Research 4(1), 283–306 (2003)
10. Joye, M., Tymen, C.: Protections against differential analysis for elliptic curve cryptography - An algebraic approach. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 377–390. Springer, Heidelberg (2001)
11. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
12. Kohel, D.: Endomorphism rings of elliptic curves over finite fields. PhD thesis, University of California, Berkeley (1996)
13. Bosma, W., Canon, J.: Handbook of Magma functions. MAGMA Group. Sydney (2003), http://magam.maths.usyd.edu.au/
14. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004)
15. Miret, J., Moreno, R., Sadornil, D., Tena, J., Valls, M.: Computing the height of volcanoes of ℓ–isogenies of elliptic curves over finite fields. Applied Mathematics and Computation 196(1), 67–76 (2008)
16. Miret, J., Sadornil, D., Tena, J., Tomàs, R., Valls, M.: Isogeny cordillera algorithm to obtain cryptographically good elliptic curves. In: Australasian Information Security Workshop: Privacy Enhancing Tecnologies (AISW), CRPIT, vol. 68, pp. 127–131 (2007)
17. Standard for Efficient Cryptography (SECG). SEC2: Recommended Elliptic Curve Domain Parameters, Version 1.0 (2000), http://www.secg.org/secg_docs.htm
18. Smart, N.: An analysis of Goubin's refined power analysis attack. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 281–290. Springer, Heidelberg (2003)
19. Vélu, J.: Isogénies entre courbes elliptiques. C. R. Acad. Sci. Paris, Ser. I Math., Serie A 273, 238–241 (1971)

Security Analysis of DRBG Using HMAC in NIST SP 800-90

Shoichi Hirose

Graduate School of Engineering, University of Fukui
hrs_shch@u-fukui.ac.jp

Abstract. HMAC_DRBG is a deterministic random bit generator using HMAC specified in NIST SP 800-90. The document claims that HMAC_DRBG is a pseudorandom bit generator if HMAC is a pseudorandom function. However, no proof is given in the document. This article provides a security analysis of HMAC_DRBG and confirms the claim.

Keywords: NIST SP 800-90, pseudorandom bit generator, HMAC, pseudorandom function.

1 Introduction

Background. Random bits are indispensable to every cryptographic application. However, it is not easy to prepare sufficient amount of truly random bits in general. Thus, most applications use a cryptographic mechanism that is called a pseudorandom bit generator (PRBG). It stretches a short sequence of random bits to a long sequence of bits that are indistinguishable from truly random bits in practice.

HMAC_DRBG is a deterministic random bit generator (DRBG) specified in NIST SP 800-90 [3]. It is claimed in NIST SP 800-90 that HMAC_DRBG is a PRBG if HMAC is a pseudorandom function (PRF). However, no proof is made public as far as the authors know.

Contribution. This article presents a security analysis of HMAC_DRBG. The result supports the claim in NIST SP 800-90 mentioned above. This article does not provide new techniques and just uses well-known ones in the analysis. In spite of this fact, the contribution of this paper is still important since HMAC_DRBG is expected to be widely used in practice.

HMAC_DRBG consists of three algorithms. They are instantiate, reseed and generate algorithms. The instantiate/reseed algorithm is used to produce/refresh a secret key. The generate algorithm produces a binary sequence from a secret key given by the instantiate or reseed algorithms. This article gives a proof for the pseudorandomness of HMAC_DRBG on the assumption that the instantiate and reseed algorithms are ideal. Namely, a secret key given to the generate algorithm is selected uniformly at random by each of them.

Related Work. NIST SP 800-90 specifies four DRBG mechanisms: Hash_DRBG, HMAC_DRBG, CTR_DRBG and Dual_EC_DRBG. Hash_DRBG and HMAC_DRBG are based on hash functions. CTR_DRBG uses a block cipher in the counter mode. Dual_EC_DRBG is based on the elliptic curve discrete logarithm problem. A security analysis of these DRBGs was presented in [9]. However, the discussion was quite informal. A security analysis of CTR_DRBG was also presented in [7]. Brown and Gjøsteen [6] provided a detailed security analysis of Dual_EC_DRBG.

There also exist some other approved DRBGs in ANSI X.9.31 [2], ANSI X.9.62-1998 [1] and FIPS PUB 186-2 [11]. The security of these algorithms was discussed in [8] and [10].

HMAC was proposed by Bellare, Canetti and Krawczyk [5]. It was proved to be a PRF on the assumption that the compression function of the underlying iterated hash function is a PRF with two keying strategies[4]. Actually, HMAC is used as a PRF in many cryptographic schemes.

Organization. This article is organized as follows. Definitions of a pseudorandom bit generator and a pseudorandom function are given in Sect. 2. A description of HMAC_DRBG is presented in Sect. 3. Results on security analysis of HMAC_DRBG are shown in Sect. 4. Concluding remarks are given in Sect. 5.

2 Preliminaries

Let $a \xleftarrow{\$} A$ represent that an element a is selected uniformly at random from a set A.

2.1 A Pseudorandom Bit Generator

Let G be a function such that $G : \{0,1\}^n \to \{0,1\}^l$. Let $\mathcal{D}$ be a probabilistic algorithm which outputs 0 or 1 for a given input in $\{0,1\}^l$. The goal of $\mathcal{D}$ is to tell whether a given input is $G(k)$ for k selected uniformly at random or it is selected uniformly at random. The advantage of $\mathcal{D}$ against G is defined as follows:

$$\mathrm{Adv}_G^{\mathrm{prbg}}(\mathcal{D}) = \left|\Pr[\mathcal{D}(G(k)) = 1 \,|\, k \xleftarrow{\$} \{0,1\}^n] - \Pr[\mathcal{D}(s) = 1 \,|\, s \xleftarrow{\$} \{0,1\}^l]\right|,$$

where the probabilities are taken over the coin tosses by $\mathcal{D}$ and the uniform distributions on $\{0,1\}^n$ or $\{0,1\}^l$. G is called a pseudorandom bit generator (PRBG) if $l > n$ and $\mathrm{Adv}_G^{\mathrm{prbg}}(\mathcal{D})$ is negligible for any efficient $\mathcal{D}$.

2.2 A Pseudorandom Function

Let H be a keyed function such that $H : K \times D \to R$, where K is a key space, D is a domain, and R is a range. $H(k, \cdot)$ is denoted by $H_k(\cdot)$. Let $\mathcal{A}$ be a probabilistic algorithm which has oracle access to a function from D to R. The goal of $\mathcal{A}$ is to tell whether the oracle is H_k for k selected uniformly at random or it is a function selected uniformly at random. $\mathcal{A}$ first asks elements in D and obtains the corresponding elements in R with respect to the function, and then outputs 0 or 1. $\mathcal{A}$ makes the queries adaptively: $\mathcal{A}$ makes a new query after receiving an answer to the current query.

Let F be the set of all functions from D to R. The advantage of $\mathcal{A}$ for H is defined as follows:

$$\mathrm{Adv}_H^{\mathrm{prf}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{H_k} = 1 \mid k \overset{\$}{\leftarrow} K] - \Pr[\mathcal{A}^{\rho} = 1 \mid \rho \overset{\$}{\leftarrow} F]\right|,$$

where the probabilities are taken over the coin tosses by $\mathcal{A}$ and the uniform distributions on K or F. H is called a pseudorandom function (PRF) if $\mathrm{Adv}_H^{\mathrm{prf}}(\mathcal{A})$ is negligible for any efficient $\mathcal{A}$.

3 HMAC_DRBG

HMAC_DRBG is a DRBG using HMAC in NIST SP 800-90 [3]. It consists of three algorithms: an instantiate algorithm, a reseed algorithm and a generate algorithm. The instantiate algorithm is used to produce a secret key. The reseed algorithm is used to refresh it. These algorithms are out of the scope of the article. They also use HMAC to produce the secret keys. However, the security of the outputs is not based on the secrecy and randomness of the keys given to HMAC but the data given to HMAC via message input. From this viewpoint, we may say that they abuse HMAC.

We only assume that the keys produced by the instantiate and reseed algorithms are ideal. Namely, we assume that a secret key given to the generate algorithm is selected uniformly at random.

The generate algorithm produces a binary sequence for a given secret key. A description of this algorithm is given in the following part. The instantiate and reseed algorithms are given in Appendix A for reference.

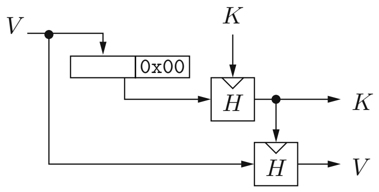
Notation. HMAC is simply denoted by H. Let n denote the output length of H. Let null denote an empty sequence. Concatenation of binary sequences x and y is denoted by $x \| y$. The symbol $\|$ is sometimes omitted.

3.1 Internal State

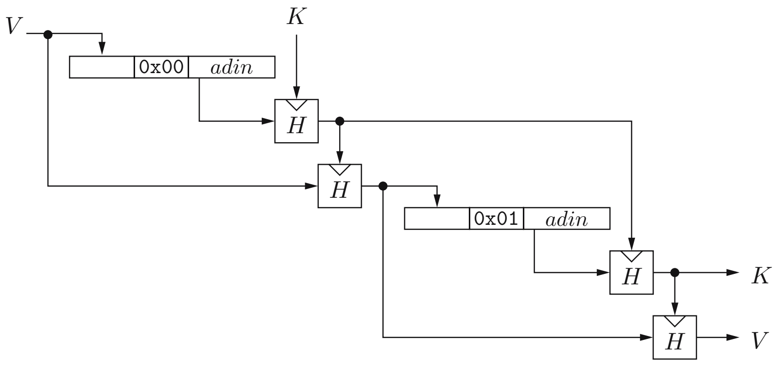
The internal state of HMAC_DRBG includes $K \in \{0,1\}^n$, $V \in \{0,1\}^n$, and a reseed counter $d \geq 1$. K and V are assumed to be secret. The reseed counter d is an integer variable indicating the number of requests for pseudorandom bits since instantiation or reseeding.

3.2 The Function Update

The function Update is used in the generate algorithm to produce a secret key (K, V) for the next invocation of the generate algorithm. It is described as follows.



(a) If *adin* is an empty sequence



(b) If *adin* is not an empty sequence

Fig. 1. The function Update

```
Update($K$, $V$, $adin$):
 1. $K = H(K, V\|\texttt{0x00}\|adin)$
 2. $V = H(K, V)$
 3. If $adin = \mathsf{null}$, then return $(K, V)$
 4. $K = H(K, V\|\texttt{0x01}\|adin)$
 5. $V = H(K, V)$
 6. Return $(K, V)$
```

adin is an optional additional input. Update is shown in Fig. 1.

3.3 The Algorithm Generate

The generate algorithm Generate produces a binary sequence s for given secret (K, V) and an optional additional input *adin*. It is described as follows.

```
Generate($K$, $V$, $adin$):
 1. If $d > w$, then return an indication that a reseed is required.
 2. $(K, V) = \mathsf{Update}(K, V, adin)$ if $adin \neq \mathsf{null}$
 3. $tmp = \mathsf{null}$
 4. While $|tmp| < \ell$ do:
    (a) $V = H(K, V)$
    (b) $tmp = tmp\|V$
 5. $s =$ the leftmost $\ell$ bits of $tmp$
```

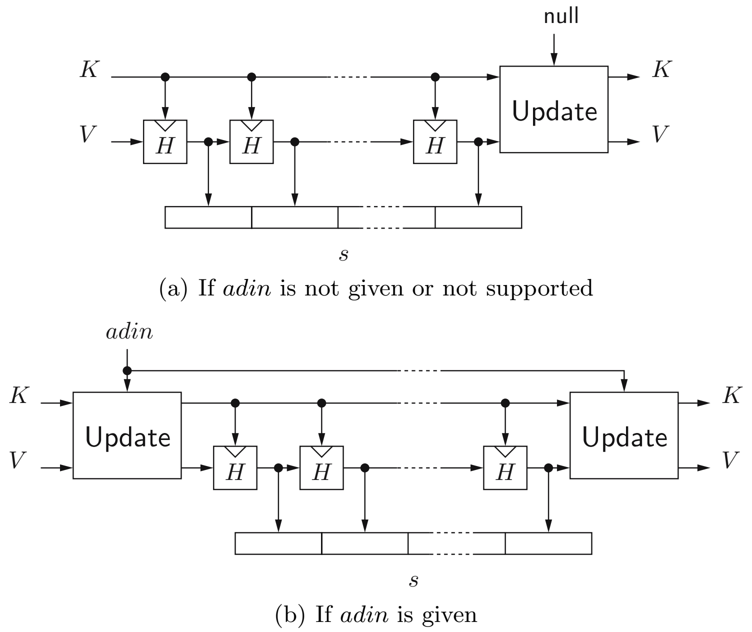


Fig. 2. The algorithm Generate. The update of the reseed counter d is omitted.

Table 1. The maximum sizes of parameters

parameter	maximum size
$\|adin\|$	2^{35} bits
ℓ	2^{19} bits
w	2^{48}

6. $(K, V) = \mathsf{Update}(K, V, adin)$
7. $d = d + 1$
8. Return s and (K, V)

The maximum sizes of parameters are given in Table 1. w is called a reseed interval. It represents the total number of requests for pseudorandom bits between reseeding. If *adin* is not supported, then the step 6 is executed with $adin = \mathsf{null}$. Generate is given in Fig. 2.

4 Security Analysis

For simplicity, we assume that ℓ is a multiple of n. Thus, the output length of Generate is $\ell + 2n$. We analyze the security of a generator $G : \{0,1\}^{2n} \to \{0,1\}^{w\ell}$ defined as follows:

```
$G(K, V)$:
 1. $(K_0, V_0) = (K, V)$
 2. $s = \mathsf{null}$
 3. for $i = 1$ to $w$ do
    (a) $(s^i, K_i, V_i) = \mathsf{Generate}(K_{i-1}, V_{i-1}, adin_{i-1})$
    (b) $s = s \| s^i$
    (c) Return $s$
```

We make $adin_i$ implicit in the notation of $G(K, V)$, because the analysis given in the remaining parts does not depend on the value of $adin_i$. It depends only on whether $adin_i = \mathsf{null}$ or not.

4.1 If $adin = \mathsf{null}$

First, notice that we cannot prove the pseudorandomness of G directly from that of Generate. Generate is not a PRBG if $adin = \mathsf{null}$. Let $q = \ell/n$ and $\mathsf{Generate}(K, V) = s_1 \| s_2 \| \cdots \| s_q \| K' \| V'$, where $s_j \in \{0,1\}^n$ for $1 \le j \le q$. Then, $V' = H_{K'}(s_q)$. Thus, it is easy to distinguish $s_1 \| s_2 \| \cdots \| s_q \| K' \| V'$ from a truly random sequence of length $\ell + 2n$.

We introduce two generators G_{01} and G_{02}. $G_{01} : \{0,1\}^{2n} \to \{0,1\}^{\ell+2n}$ is described as follows:

```
$G_{01}(K, V)$:
 1. $s = V$
 2. $tmp = \mathsf{null}$
 3. While $|tmp| < \ell$ do:
    (a) $V = H(K, V)$
    (b) $tmp = tmp \| V$
 4. $s = s \| tmp$
 5. $K = H(K, V \| \mathtt{0x00})$
 6. $s = s \| K$
 7. Return $s$
```

$G_{02} : \{0,1\}^{2n} \to \{0,1\}^{\ell+3n}$ is described as follows:

```
$G_{02}(K, V)$ :
 1. $s = V$
 2. $V = H(K, V)$
 3. $s = s \| G_{01}(K, V)$
 4. Return $s$
```

The only difference between G_{01} and G_{02} is that G_{02} calls HMAC more than G_{01} by one time.

Let $G_0 : \{0,1\}^{2n} \to \{0,1\}^{w(\ell+n)+n}$ be a generator which, for a given (K, V), produces a sequence

$$\begin{cases} s_0^1 s_1^1 \cdots s_{q+1}^1 & \text{if } w = 1 \\ s_0^1 s_1^1 \cdots s_{q-1}^1 \| s_{-1}^2 s_0^2 \cdots s_{q-1}^2 \| \cdots \| s_{-1}^{w-1} s_0^{w-1} \cdots s_{q-1}^{w-1} \| s_{-1}^w s_0^w \cdots s_{q+1}^w & \text{if } w \ge 2, \end{cases}$$

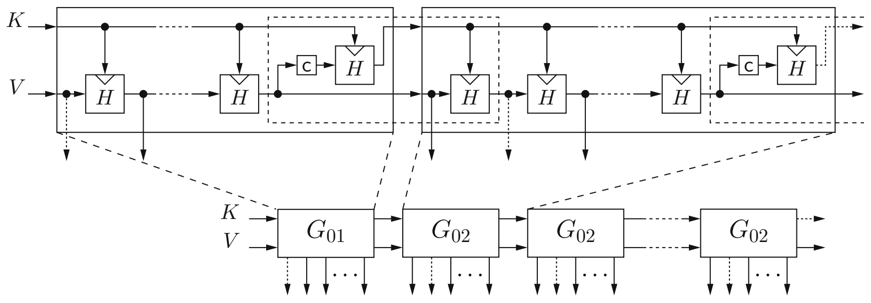


Fig. 3. A diagram of the generator G_0. c represents the concatenation with 0x00. The Update functions are surrounded by dashed rectangles.

where

1. $s^i_j \in \{0,1\}^n$,
2. $s^1_0 s^1_1 \cdots s^1_{q+1} = G_{01}(K, V)$, and
3. $s^i_{-1} s^i_0 s^i_1 \cdots s^i_{q+1} = G_{02}(s^{i-1}_{q+1}, s^{i-1}_q)$ for $2 \le i \le w$.

A diagram of G_0 is given in Fig. 3. Notice that $G(K, V)$ is a part of $G_0(K, V)$. It is obvious if $w = 1$. For $w \ge 2$,

$$
\begin{aligned}
&G(K, V) \\
&= s^1_1 s^1_2 \cdots s^1_{q-1} \| s^2_{-1} s^2_1 s^2_2 \cdots s^2_{q-1} \| \cdots \| s^{w-1}_{-1} s^{w-1}_1 s^{w-1}_2 \cdots s^{w-1}_{q-1} \| s^w_{-1} s^w_1 s^w_2 \cdots s^w_q \\
&= s^1_1 s^1_2 \cdots s^1_q \| s^2_1 s^2_2 \cdots s^2_q \| \cdots \| s^{w-1}_1 s^{w-1}_2 \cdots s^{w-1}_q \| s^w_1 s^w_2 \cdots s^w_q ,
\end{aligned}
$$

where $s^{i+1}_{-1} = s^i_q$ for $1 \le i \le w - 1$. Thus, G is a PRBG if G_0 is a PRBG. We will discuss the security of G_0 in the remaining part.

We first show that both G_{01} and G_{02} are PRBGs if HMAC is a PRF. For an algorithm A, let t_A be the running time of A.

Lemma 1. *Let $\mathcal{D}$ be a distinguisher for G_{01} which runs in $t_{\mathcal{D}}$. Then, there exists an adversary $\mathcal{A}$ for HMAC such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_{01}}(\mathcal{D}) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}) + \frac{q(q-1)}{2^{n+1}}.$$

$\mathcal{A}$ runs in $t_{\mathcal{D}} + O(\ell)$ and asks at most $q + 1$ queries.

Proof. Let $F_{*,n}$ be the set of all functions from $\{0,1\}^*$ to $\{0,1\}^n$. Let $\hat{G}_{01}(\rho, \cdot)$ be a generator obtained from G_{01} by replacing HMAC with a function $\rho \in F_{*,n}$. Let

$$
\begin{aligned}
P_0 &= \Pr[\mathcal{D}(s) = 1 \mid (K, V) \xleftarrow{\$} \{0,1\}^{2n} \wedge s \leftarrow G_{01}(K, V)], \\
P_1 &= \Pr[\mathcal{D}(s) = 1 \mid \rho \xleftarrow{\$} F_{*,n} \wedge V \xleftarrow{\$} \{0,1\}^n \wedge s \leftarrow \hat{G}_{01}(\rho, V)], \\
P_2 &= \Pr[\mathcal{D}(s) = 1 \mid s \xleftarrow{\$} \{0,1\}^{\ell+2n}].
\end{aligned}
$$

Then,

$$\mathrm{Adv}_{G}^{\mathrm{prbg}}(\mathcal{D}) = |P_0 - P_2| \le |P_0 - P_1| + |P_1 - P_2| .$$

Let $\hat{G}_{01}(\rho, V) = \hat{s}_0 \hat{s}_1 \cdots \hat{s}_{q+1}$, where $\hat{s}_j \in \{0,1\}^n$ for $0 \le j \le q+1$. If $\rho \xleftarrow{\$} F_{*,n}$, then $\hat{G}_{01}(\rho, V)$ and a random sequence of length $\ell + 2\,n$ is completely indistinguishable as far as $\hat{s}_{j_1} \ne \hat{s}_{j_2}$ for every j_1 and j_2 such that $0 \le j_1 < j_2 \le q-1$. Notice that $\hat{s}_j \ne \hat{s}_q \| \mathtt{0x00}$ for every $0 \le j \le q-1$. Thus,

$$|P_1 - P_2| \le \frac{q(q-1)}{2^{n+1}} .$$

On the other hand, for $|P_0 - P_1|$, it is easy to see that we can construct an adversary $\mathcal{A}$ for HMAC, using $\mathcal{D}$ as a subroutine, such that

$$\mathrm{Adv}_{\mathrm{HMAC}}^{\mathrm{prf}}(\mathcal{A}) \ge |P_0 - P_1| ,$$

where $\mathcal{A}$ runs in $t_{\mathcal{D}} + O(\ell)$ and asks at most $q+1$ queries. □

Lemma 2. *Let $\mathcal{D}$ be a distinguisher for G_{02} which runs in $t_{\mathcal{D}}$. Then, there exists an adversary $\mathcal{A}$ such that*

$$\mathrm{Adv}_{G_{02}}^{\mathrm{prbg}}(\mathcal{D}) \le \mathrm{Adv}_{\mathrm{HMAC}}^{\mathrm{prf}}(\mathcal{A}) + \frac{q(q+1)}{2^{n+1}} .$$

$\mathcal{A}$ runs in $t_{\mathcal{D}} + O(\ell)$ and asks at most $q+2$ queries.

Proof. The proof is similar to that of Lemma 1. □

For $w \ge 2$, let $G_{02}^{(w-1)}$ be a generator which calls G_{02} $(w-1)$ times successively. Namely,

$$G_{02}^{(w-1)}(s_{q+1}^1, s_q^1) = s_{-1}^2 s_0^2 \cdots s_{q-1}^2 \| \cdots \| s_{-1}^{w-1} s_0^{w-1} \cdots s_{q-1}^{w-1} \| s_{-1}^w s_0^w \cdots s_{q+1}^w .$$

Lemma 3. *Let $\mathcal{D}$ be a distinguisher for $G_{02}^{(w-1)}$ which runs in $t_{\mathcal{D}}$. Then, there exists a distinguisher $\mathcal{D}'$ of G_{02} such that*

$$\mathrm{Adv}_{G_{02}^{(w-1)}}^{\mathrm{prbg}}(\mathcal{D}) \le (w-1)\mathrm{Adv}_{G_{02}}^{\mathrm{prbg}}(\mathcal{D}') .$$

$\mathcal{D}'$ runs in $t_{\mathcal{D}} + (w-2)\, t_{G_{02}} + O(w\ell)$.

Proof. For a given input $s \in \{0,1\}^{\ell+3n}$, the distinguisher $\mathcal{D}'$ behaves as follows:

1. Select $2 \le r \le w$ uniformly at random.
2. If $r \ge 3$, then select $s_{-1}^2 s_0^2 \cdots s_{q-1}^2, \ldots, s_{-1}^{r-1} s_0^{r-1} \cdots s_{q-1}^{r-1}$ uniformly at random.
3. Let $s_{-1}^r s_0^r \cdots s_{q+1}^r = s$.
4. If $r < w$, $s_{-1}^i s_0^i \cdots s_{q+1}^i = G_{02}(s_{q+1}^{i-1}, s_q^{i-1})$ for $r+1 \le i \le w$.
5. Call $\mathcal{D}$ with
   $\boldsymbol{s} = s_{-1}^2 s_0^2 \cdots s_{q-1}^2 \| \cdots \| s_{-1}^{w-1} s_0^{w-1} \cdots s_{q-1}^{w-1} \| s_{-1}^w s_0^w \cdots s_{q+1}^w$.
6. Output $\mathcal{D}$'s output.

Then,

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{prbg}}_{G_{02}}(\mathcal{D}') \\
&= \left|\Pr[\mathcal{D}'(G_{02}(K,V)) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n}] - \Pr[\mathcal{D}'(s) = 1 \,|\, s \xleftarrow{\$} \{0,1\}^{\ell+3n}]\right| \\
&= \Big|\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n} \wedge s \leftarrow G_{02}(K,V)] \\
&\qquad - \Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, s \xleftarrow{\$} \{0,1\}^{\ell+3n}]\Big| .
\end{aligned}$$

$$\begin{aligned}
&\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n} \wedge s \leftarrow G_{02}(K,V)] \\
&= \sum_{u=2}^{w} \Pr[r = u \wedge \mathcal{D}(\boldsymbol{s}) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n} \wedge s \leftarrow G_{02}(K,V)] \\
&= \sum_{u=2}^{w} \frac{\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n} \wedge s \leftarrow G_{02}(K,V) \wedge r = u]}{w-1} \\
&= \frac{\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n} \wedge \boldsymbol{s} \leftarrow G_{02}^{(w-1)}(K,V)]}{w-1} + \\
&\qquad \sum_{u=3}^{w} \frac{\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n} \wedge s \leftarrow G_{02}(K,V) \wedge r = u]}{w-1} .
\end{aligned}$$

$$\begin{aligned}
\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, s \xleftarrow{\$} \{0,1\}^{\ell+3n}] &= \sum_{u=2}^{w-1} \frac{\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, s \xleftarrow{\$} \{0,1\}^{\ell+3n} \wedge r = u]}{w-1} \\
&\qquad + \frac{\Pr[\mathcal{D}(\boldsymbol{s}) = 1 \,|\, \boldsymbol{s} \xleftarrow{\$} \{0,1\}^{(w-1)(\ell+n)+2n}]}{w-1} .
\end{aligned}$$

There may exist a better distinguisher for G_{02} than $\mathcal{D}'$ with the same running time. Thus, we have

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_{02}}(\mathcal{D}') \geq \frac{1}{w-1}\mathrm{Adv}^{\mathrm{prbg}}_{G_{02}^{(w-1)}}(\mathcal{D}).$$

The running time of $\mathcal{D}'$ is at most $t_{\mathcal{D}} + (w-2)\, t_{G_{02}} + O(w\ell)$. □

Lemma 4. *Let $\mathcal{D}$ be a distinguisher for G_0 which runs in $t_{\mathcal{D}}$. Then, there exist distinguishers $\mathcal{D}'$ for G_{01} and $\mathcal{D}''$ for $G_{02}^{(w-1)}$ such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_0}(\mathcal{D}) \leq \mathrm{Adv}^{\mathrm{prbg}}_{G_{01}}(\mathcal{D}') + \mathrm{Adv}^{\mathrm{prbg}}_{G_{02}^{(w-1)}}(\mathcal{D}'').$$

$\mathcal{D}'$ runs in $t_{\mathcal{D}} + t_{G_{02}^{(w-1)}} + O(w\ell)$, and $\mathcal{D}''$ runs in $t_{\mathcal{D}} + O(w\ell)$.

Proof. Let

$$\begin{aligned}
P_0 &= \Pr[\mathcal{D}(s) = 1 \,|\, (K,V) \xleftarrow{\$} \{0,1\}^{2n} \wedge s \leftarrow G_0(K,V)], \\
P_1 &= \Pr[\mathcal{D}(s) = 1 \,|\, s_0^1 \cdots s_{q+1}^1 \xleftarrow{\$} \{0,1\}^{\ell+2n} \wedge s \leftarrow s_0^1 \cdots s_{q-1}^1 \| G_{02}^{(w-1)}(s_{q+1}^1, s_q^1)], \\
P_2 &= \Pr[\mathcal{D}(s) = 1 \,|\, s \xleftarrow{\$} \{0,1\}^{w(\ell+n)+n}].
\end{aligned}$$

Then, there exist $\mathcal{D}'$ and $\mathcal{D}''$ such that

$$\begin{aligned}\mathrm{Adv}^{\mathrm{prbg}}_{G_0}(\mathcal{D}) &= |P_0 - P_2| \le |P_0 - P_1| + |P_1 - P_2| \\ &\le \mathrm{Adv}^{\mathrm{prbg}}_{G_{01}}(\mathcal{D}') + \mathrm{Adv}^{\mathrm{prbg}}_{G_{02}^{(w-1)}}(\mathcal{D}'').\end{aligned}$$

The running time of $\mathcal{D}'$ is $t_{\mathcal{D}} + t_{G_{02}^{(w-1)}} + O(w\ell)$. The running time of $\mathcal{D}''$ is $t_{\mathcal{D}} + O(w\ell)$. □

The following theorem directly follows from Lemmas 1, 2, 3 and 4. It implies that G_0 is a PRBG if HMAC is a PRF.

Theorem 1. *Let $\mathcal{D}$ be a distinguisher for G_0 which runs in $t_{\mathcal{D}}$. Then, there exists an adversary $\mathcal{A}$ for HMAC such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_0}(\mathcal{D}) \le w\,\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}) + \frac{wq(q+1)}{2^{n+1}}.$$

$\mathcal{A}$ runs in $t_{\mathcal{D}} + w(q+2)\,t_{\mathrm{HMAC}} + O(w\ell)$ and asks at most $q+2$ queries, where the length of each query is at most $n+8$.

Remark 1. Suppose that SHA-1 is the underlying hash function of HMAC. Then, $n = 160$. Suppose that $w = 2^{48}$ and $\ell = 2^{11} \times 160$ ($\le 2^{19}$). Then,

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_0}(\mathcal{D}) \le 2^{48}\,\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}) + \frac{1}{2^{90}},$$

where $\mathcal{A}$ runs in time $t_{\mathcal{D}} + 2^{60}\,t_{\mathrm{HMAC}} + O(2^{66.3})$ and makes at most 2050 queries. The big-O notation is abused here. $O(2^{66.3})$ is upper bounded by $c \times 2^{66.3}$ for some positive constant c.

Remark 2. Suppose that SHA-256 is the underlying hash function of HMAC. Then, $n = 256$. Suppose that $w = 2^{48}$ and $\ell = 2^{19}$. Then,

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_0}(\mathcal{D}) \le 2^{48}\,\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}) + \frac{1}{2^{186}},$$

where $\mathcal{A}$ runs in time $t_{\mathcal{D}} + 2^{60}t_{\mathrm{HMAC}} + O(2^{67})$ and makes at most 2050 queries.

4.2 If $adin \ne$ null

If $adin \ne$ null, then the analysis is similar but tedious. We first define several generators.

Let $g_{10} : \{0,1\}^{2n} \to \{0,1\}^{2n}$ be a generator such that

$$g_{10}(K, V) = V \| H(K, V \| \texttt{0x00} \| adin).$$

Let $g_{11} : \{0,1\}^{2n} \to \{0,1\}^{3n}$ be a generator such that $g_{11}(K, V) = s$, where s is obtained as follows:

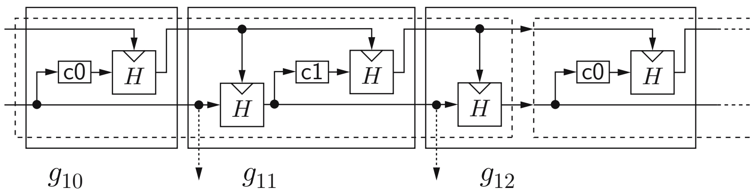


Fig. 4. A diagram of the generators g_{10}, g_{11} and g_{12}. The Update functions are surrounded by dashed rectangles. c0 represents the concatenation with $\texttt{0x00}\|adin$, and c1 represents the concatenation with $\texttt{0x01}\|adin$.

1. $V_1 = H(K, V)$
2. $V_2 = H(K, V_1\|\texttt{0x01}\|adin)$
3. $s = V\|V_1\|V_2$

Let $g_{12} : \{0,1\}^{2n} \to \{0,1\}^{3n}$ be a generator such that $g_{12}(K, V) = s$, where s is obtained as follows:

1. $V_1 = H(K, V)$
2. $V_2 = H(K, V_1\|\texttt{0x00}\|adin)$
3. $s = V\|V_1\|V_2$

The generators g_{10}, g_{11} and g_{12} are depicted in Fig. 4.

Let $G_{10} : \{0,1\}^{2n} \to \{0,1\}^{\ell+3n}$ be a generator equivalent to G_{02} defined in the previous subsection.

Using the generators defined above, we further define two generators G_{11} and G_{12}. $G_{11} : \{0,1\}^{2n} \to \{0,1\}^{\ell+4n}$ is described as follows: $G_{11}(K, V) = s_{-2}s_{-1}\cdots s_{q+1}$, where

1. $V_1K_1 = g_{10}(K, V)$
2. $s_{-2}V_2K_2 = g_{11}(K_1, V_1)$
3. $s_{-1}s_0\cdots s_qs_{q+1} = G_{10}(K_2, V_2)$

$G_{12} : \{0,1\}^{2n} \to \{0,1\}^{\ell+6n}$ is described as follows: $G_{12}(K, V) = s_{-4}s_{-3}\cdots s_{q+1}$, where

1. $s_{-4}V_1K_1 = g_{11}(K, V)$
2. $s_{-3}V_2K_2 = g_{12}(K_1, V_1)$
3. $s_{-2}V_3K_3 = g_{11}(K_2, V_2)$
4. $s_{-1}s_0\cdots s_qs_{q+1} = G_{10}(K_3, V_3)$

Now, we are ready to discuss the pseudorandomness of $G(K, V)$. Let $G_1 : \{0,1\}^{2n} \to \{0,1\}^{w(\ell+4n)}$ be a generator which, for a given (K, V), produces a sequence

$$\begin{cases} s^1_{-2}s^1_{-1}\cdots s^1_{q+1} & \text{if } w = 1 \\ s^1_{-2}s^1_{-1}\cdots s^1_{q-1}\|s^2_{-4}\cdots s^2_{q-1}\|\cdots\|s^{w-1}_{-4}\cdots s^{w-1}_{q-1}\|s^w_{-4}\cdots s^w_{q+1} & \text{if } w \geq 2, \end{cases}$$

where

1. $s^i_j \in \{0,1\}^n$,
2. $s^1_{-2}s^1_{-1}\cdots s^1_{q+1} = G_{11}(K,V)$, and
3. $s^i_{-4}\cdots s^i_{q+1} = G_{12}(s^{i-1}_{q+1}, s^{i-1}_q)$ for $2 \le i \le w$.

Notice that $G(K,V)$ is a part of $G_1(K,V)$. It is easy to see if $w=1$. For $w \ge 2$,

$$\begin{aligned}
&G(K,V)\\
&= s^1_1 s^1_2 \cdots s^1_{q-1} \| s^2_{-4} s^2_1 s^2_2 \cdots s^2_{q-1} \| \cdots \| s^{w-1}_{-4} s^{w-1}_1 s^{w-1}_2 \cdots s^{w-1}_{q-1} \| s^w_{-4} s^w_1 s^w_2 \cdots s^w_{q+1}\\
&= s^1_1 s^1_2 \cdots s^1_q \| s^2_1 s^2_2 \cdots s^2_q \| \cdots \| s^{w-1}_1 s^{w-1}_2 \cdots s^{w-1}_q \| s^w_1 s^w_2 \cdots s^w_q,
\end{aligned}$$

where $s^{i+1}_{-4} = s^i_q$ for $1 \le i \le w-1$. Thus, we discuss the pseudorandomness of G_1 in the remaining part. We only present the results since the proofs are similar.

Lemma 5. *Let $\mathcal{D}$ be a distinguisher for g_{10} which runs in $t_{\mathcal{D}}$. Then, there exists an adversary $\mathcal{A}$ for HMAC such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{g_{10}}(\mathcal{D}) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}).$$

$\mathcal{A}$ runs in $t_{\mathcal{D}} + O(n)$ and asks 1 query.

Lemma 6. *Let g be g_{11} or g_{12}. Let $\mathcal{D}$ be a distinguisher for g which runs in $t_{\mathcal{D}}$. Then, there exists an adversary $\mathcal{A}$ for HMAC such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{g}(\mathcal{D}) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}).$$

$\mathcal{A}$ runs in $t_{\mathcal{D}} + O(n)$ and asks at most 2 queries.

Lemma 7. *Let $\mathcal{D}$ be a distinguisher for G_{11} which runs in $t_{\mathcal{D}}$. Then, there exist $\mathcal{D}_0$, $\mathcal{D}_1$ and $\mathcal{D}'$ such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_{11}}(\mathcal{D}) \le \mathrm{Adv}^{\mathrm{prbg}}_{g_{10}}(\mathcal{D}_0) + \mathrm{Adv}^{\mathrm{prbg}}_{g_{11}}(\mathcal{D}_1) + \mathrm{Adv}^{\mathrm{prbg}}_{G_{10}}(\mathcal{D}').$$

$\mathcal{D}_0$ runs in $t_{\mathcal{D}} + t_{g_{11}} + t_{G_{10}} + O(n)$. $\mathcal{D}_1$ runs in $t_{\mathcal{D}} + t_{G_{10}} + O(n)$. $\mathcal{D}'$ runs in $t_{\mathcal{D}} + O(n)$.

Lemma 8. *Let $\mathcal{D}$ be a distinguisher for G_{12} which runs in $t_{\mathcal{D}}$. Then, there exist $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}'$ such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_{12}}(\mathcal{D}) \le 2\,\mathrm{Adv}^{\mathrm{prbg}}_{g_{11}}(\mathcal{D}_1) + \mathrm{Adv}^{\mathrm{prbg}}_{g_{12}}(\mathcal{D}_2) + \mathrm{Adv}^{\mathrm{prbg}}_{G_{10}}(\mathcal{D}').$$

$\mathcal{D}_1$ runs in $t_{\mathcal{D}} + t_{g_{11}} + t_{g_{12}} + t_{G_{10}} + O(n)$. $\mathcal{D}_2$ runs in $t_{\mathcal{D}} + t_{g_{11}} + t_{G_{10}} + O(n)$. $\mathcal{D}'$ runs in $t_{\mathcal{D}} + O(n)$.

The following theorem directly follows from Lemmas 5, 6, 7 and 8. It implies that G_1 is a PRBG if HMAC is a pseudorandom function.

Theorem 2. *Let $\mathcal{D}$ be a distinguisher for G_1 which runs in $t_{\mathcal{D}}$. Then, there exist adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ such that*

$$\mathrm{Adv}^{\mathrm{prbg}}_{G_1}(\mathcal{D}) \le w\,\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}_1) + 3\,w\,\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}_2) + \frac{wq(q-1)}{2^{n+1}}.$$

$\mathcal{A}_1$ runs in $t_{\mathcal{D}} + w(q+8)\,t_{\mathrm{HMAC}} + O(w\ell)$ and asks at most $q+1$ queries, and $\mathcal{A}_2$ runs in $t_{\mathcal{D}} + (w+1)(q+8)\,t_{\mathrm{HMAC}} + O(w\ell)$ and asks at most 2 queries.

5 Conclusion

We have shown that the binary sequence generation algorithm of HMAC_DRBG is a PRBG if HMAC is a PRF. Future work includes analysis of the instantiate and reseed algorithms of HMAC_DRBG.

Acknowledgements

The author would like to thank anonymous reviewers for their valuable comments. This research was supported in part by the National Institute of Information and Communications Technology, Japan.

References

1. American National Standards Institute. Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ECDSA). ANSI X9.62-1998 (1998)
2. American National Standards Institute. Digital signatures using reversible public key cryptography for the financial services industry (rDSA). ANSI X9.31-1998 (1998)
3. Barker, E., Kelsey, J.: Recommendation for random number generation using deterministic random bit generators (revised). NIST Special Publication 800-90 (2007)
4. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006), `http://eprint.iacr.org/`
5. 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)
6. Brown, D.R., Gjøsteen, K.: A security analysis of the NIST SP 800-90 elliptic curve random number generator. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 466–481. Springer, Heidelberg (2007)
7. Campagna, M.J.: Security bounds for the NIST codebook-based deterministic random bit generator. Cryptology ePrint Archive: Report 2006/379, `http://eprint.iacr.org/`
8. Desai, A., Hevia, A., Yin, Y.L.: A practice-oriented treatment of pseudorandom number generators. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 368–383. Springer, Heidelberg (2002)
9. Kan, W.: Analysis of underlying assumptions in NIST DRBGs. Cryptology ePrint Archive: Report 2007/345, `http://eprint.iacr.org/`
10. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Cryptanalytic attacks on pseudorandom number generators. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 168–188. Springer, Heidelberg (1998)
11. U.S. Department of Commerce/National Institute of Standards and Technology. Digital signature standard (DSS). Federal Information Processing Standards Publication 186-2 (+Change Notice) (2000)

A The Instantiate and Reseed Algorithms of HMAC_DRBG

The internal state of HMAC_DRBG includes $K \in \{0,1\}^n$, $V \in \{0,1\}^n$, and a reseed counter d. $data$ is the entropy source. $adin$ is an optional additional input.

The Instantiate Algorithm. The instantiate algorithm Instantiate is described as follows:

```
Instantiate($data, nonce, adin$):
  1. $seed = data \| nonce \| adin$
  2. $K = \mathsf{0x0000 \cdots 00}$
  3. $V = \mathsf{0x0101 \cdots 01}$
  4. $(K, V) = \mathsf{Update}(seed, K, V)$
  5. $d = 1$
  6. Return $(K, V)$ and $d$.
```

If $adin$ is not supported, then the first step of the procedure is replaced by

$$seed = data \| nonce.$$

The Reseed Algorithm. The reseed algorithm Reseed is described as follows:

```
Reseed($K, V, d, data, adin$):
  1. $seed = data \| adin$
  2. $(K, V) = \mathsf{Update}(seed, K, V)$
  3. $d = 1$
  4. Return $(K, V)$ and $d$.
```

The input (K, V) to Reseed is given by the latest Generate. If $adin$ is not supported, then the first step of the procedure is replaced by

$$seed = data.$$

Compact Implementation of SHA-1 Hash Function for Mobile Trusted Module

Mooseop Kim[1], Jaecheol Ryou[2], and Sungik Jun[1]

[1] Electronics and Telecommunications Research Institute (ETRI)
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-700, South Korea
gomskim@etri.re.kr

[2] Division of Electrical and Computer Engineering, Chungnam National University
220 Gung-dong, Yuseong-gu, Daejeon, 305-764, South Korea
jcryou@home.cnu.ac.kr

Abstract. We present a compact SHA-1 hardware architecture suitable for the Mobile Trusted Module (MTM) that requires low-area and low-power characteristics. The built-in SHA-1 engine in MTM is one of the most important circuit blocks and contributes the performance of the whole platform because it is used as key primitives supporting platform integrity and command authentication. Unlike personal computers, mobile platform have very stringent limitations with respect to available power, physical circuit area, and cost. Therefore special architecture and design methods for a compact SHA-1 hardware module are required. Our SHA-1 hardware can compute 512-bit data block using 6,812 gates on a $0.25\mu m$ CMOS process. The highest operation frequency and throughput of the proposed architecture are 114MHz and 164Mbps, which satisfies processing requirement for the mobile application.

1 Introduction

Mobile Trusted Module (MTM) [1] guarantees the integrity of the mobile platform and is a new requirement in the process where the mobile device changes into the open platform and value-based application technology. TMP improves the reliability and security of a device using Trusted Platform Module (TPM), which ensures that the device is running authorized software and hardware. The TPM is a microcontroller-based on an industry standard specification issued by the Trusted Computing Group (TCG). It provides cryptographic functions and is able to store environmental hash values securely in so called Platform Configuration Registers (PCRs).

The built-in SHA-1 engine in MTM is used as a key primitive supporting integrity verification and used in the most of commands for authentication in the MTM. Most mobile devices do not require a very high data processing speed. For example, when cellular wireless network technology migrates from 2.5G to 3G, only the data rate is increased from 144kbps to 2Mbps. Also, data rate of Bluetooth, which is one of the wireless Personal Area Network(PAN), is only

10Mbps maximum. However, when security function is added, considerable computing power is demanded to the microprocessor of a handheld device. For example, the processing requirements for SHA-1 at 10Mbps are 115.4 MIPS [2]. In comparison, a state-of-the art handset processors, such as MPC860 and ARM7 are capable of delivering up to 106MIPS and 130MIPS, respectively [3,4]. The above data indicates a clear disparity between security processing requirements and available processor capabilities, even when assuming that the handset processor is fully dedicated to security processing. In reality, the handset processor also needs to execute the operating system, and application software, which by themselves represent a significant processing workload.

Unlike personal computers, mobile devices have strict environment in power consumption, in battery life and in available circuit area. Among these limitations, the power consumption is the major issue in the design of cryptographic circuits for mobile platforms. For battery-powered systems, the energy drawn from the battery directly influences the systems battery life, and, consequently, the duration and extent of its mobility, and its overall utility. In general, battery-driven systems operate under stringent constraint especially in limited power. The power limitation is more worsen when the mobile device is subject to the demand of security operations. By the estimates of [5], running security applications on a battery-powered device can decrease battery life by as much as half or more.

Therefore, design methodologies at different abstraction levels, such as systems, architectures, logic design, basic cells, as well as layout, must take into account to design of a low-cost SHA-1 module for trusted mobile platform.

In this paper, we introduce an efficient hardware architecture of low-cost SHA-1 algorithm for trusted mobile platforms. As a result, a compact SHA-1 hardware implementation capable of supporting the integrity check and command authentication of trusted mobile platforms was developed and evaluated.

The rest of this paper is constructed as follows. Section 2 describes a brief overview of SHA-1 algorithm and a short summary of some previous works. Section 3 describes the architecture of our SHA-1 circuits and corresponding implementation options. The implementation result is summarized and compared with other SHA-1 implementation in section 4. Finally, in section 5, we conclude this work.

2 SHA-1 Algorithm and Previous Works

2.1 SHA-1 Algorithm

SHA-1 is a technical revision of the Secure Hash Algorithm (SHA), and issued by NIST as FIPS PUB 180-1 in 1995. The SHA-1 algorithm [6] takes a message of length less than 2^{64} bits and produces a final message digest of 160 bits. The message digest is dependent of the input message, composed by multiple blocks of 512 bits each. The data block is fed to the 80 rounds of the SHA-1 Hash function in words of 32 bits, denoted by W_t.

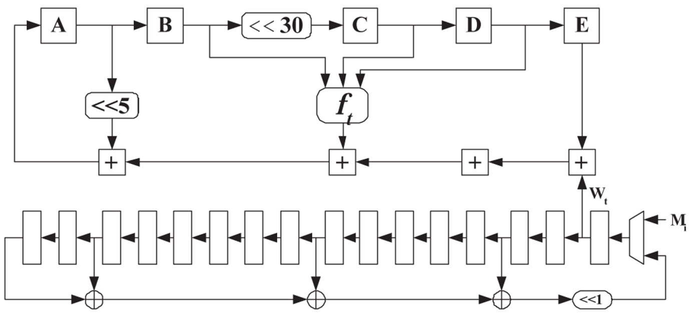


Fig. 1. Block Diagram of the SHA-1 algorithm

The first operation of SHA-1 computation is a message padding. The purpose of the message padding is to make the total length of a padded message congruent to 448 modulo 512 (length = 448 mod 512). Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits should be in the range of 1 to 512. Padding consists of a single 1-bit followed by the necessary number of 0-bits. A 64 bits binary representation of the length of the original message is appended to the end of the message.

A 160-bit buffer is used to store intermediate and final results of the message digest for SHA-1 function. The buffer can be represented as five 32-bit registers (A, B, C, D and E).

The heart of the SHA-1 algorithm is a module that consists of four rounds of processing of 20 steps each. As shown in Fig. 1, four rounds have a similar structure, but each uses a different primitive logical function, which we refer to as f_1, f_2, f_3, and f_4.

Each round takes as input the current 512-bit message block being processed and the 160-bit buffer value (A, B, C, D and E), and then updates the contents of the buffer. Each round also makes use of an additive constant K_t represents the 32-bit constant that also depends on the round. The output of the fourth round (eightieth step) is added to the input of the first round to produce a new 160-bit buffer value for the next 512-bit block calculation. After all 512-bit blocks have been processed, the output of the last block is the 160-bit message digest.

2.2 Previous Works

After the ratification of SHA-1 in 1995, numerous ASIC [7,8,9] and FPGA [10,11, 12,13,14,15,16] implementations of SHA-1 algorithm were previously proposed and evaluated. Major design differences among them lie in the trade-off between area and speed. Most of these implementations feature high speeds and high costs suitable for high-performance usages such as WTLS, IPSec and so on.

Early SHA-1 design were mostly straightforward implementations of various loop rolling architectures with limited number of architectural optimization. S.Dominikus [8] used loop rolling technique in order to reduce area requirement.

He proposed an architecture uses only 4 operation blocks, one for each round. Using a temporal register and a counter, each operation block is reused for 20 iterations. G.Selimis [12] applied the reuse technique of [8] to the non-linear function of SHA-1 algorithm. He modified the operation block to include the four non-linear functions. These architecture use a feedback structure where the data are iteratively transformed by the round functions.

Another architecture of the SHA-1 implementation is based on the use of four pipeline stages [16]. This method exploits the characteristics of the SHA-1 algorithm that requires a different non-linear function for each round. The main advantage of this architecture is that it increases the parallelism of SHA-1 algorithm, which should have a positive effect on throughput. But this approach requires large area, since this method duplicates hardware for implementing each round.

Unfortunately, most of these implementations have been designed aiming only at large message and high speed operation, with no power consumption taken into considerations.

3 Architecture Design of SHA-1 Algorithm

For our SHA-1 implementation, we assume that one 512-bit data block of preprocessed by microprocessor is stored in memory and available to our SHA-1 circuit for reading and writing. The first step for our low power circuit design was to find a minimal architecture. Fig. 2 shows main components and their interactions in our SHA-1 design: interface block, message schedule, message compression and controller.

The IO Interface block in Fig. 2 is responsible for converting 8-bit data applied to an input into 32-bit ones and vice versa when it outputs the result. It also

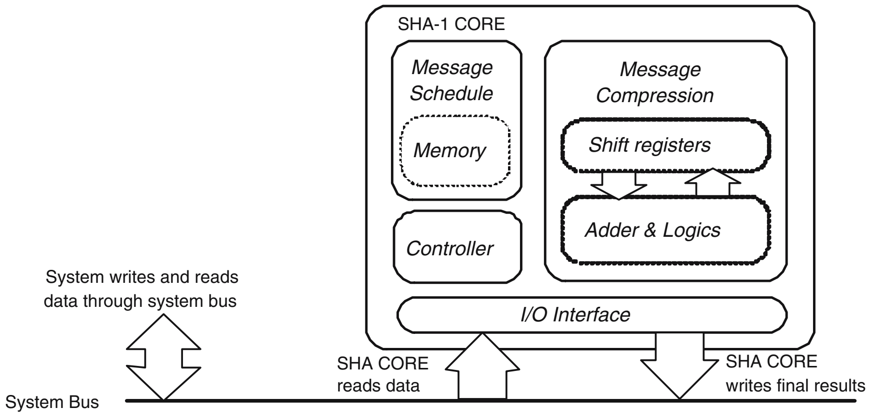


Fig. 2. Outline of SHA-1 circuit block

performs padding operation about the transformed data to generate the padded 512-bit block required by the algorithm. We use 32-bit data bus for efficient design of our SHA-1 circuit. It is not a good idea to make the bus width smaller than 32-bits, because all operation of SHA-1 algorithm and variables need 32 bits of data at one time. A smaller bus may requires less registers, but it uses more data selectors and resource sharing is hindered, resulting in an inefficient implementation.

The controller is used to generate signal sequences to check an input signal sequence or to control datapath parts. The basic structure of controller is state register and two logic blocks. The input logic block computes the next state as a function of current state and of the new sets of input signals. The output logic block generates the control signals for datapath using the function signals of the current states. The power can be consumed in the logic blocks or in the clock distribution to the flip-flops of the state register.

The efficiency of the SHA-1 hardware in terms of circuit area, power consumption and throughput is mainly determined by the structure of message schedule and message compression blocks. For a compact architecture, we use a folded architecture to design the message schedule and message compression blocks. The folded architecture, based on a rolling architecture, executes one round over several clocks to reduce hardware resource. The most important advantage of folded architecture over previous works is that it can dramatically reduce the number of adder in message compression block and the number of register in message schedule block.

Our architecture needs just one adder to compute the round operation of the SHA-1 algorithm. Adders are the most spacious part of a message compression block. We use only one 32-bit register for computation of message schedule, while previous woks require 16 32-bit registers. The number of adders and registers determines the overall size of a SHA-1 hardware circuit. Thus, using an efficient approach for implementing both message compression and message schedule are crucial for SHA-1 hardware.

In the following paragraphs, the method presented by this paper to reduce the area of message compression and message schedule is explained in detail.

3.1 Design of Message Compression

The message compression block performs actual hashing. In each step, it processes a 32-bit data W_t, which is generated by the message schedule block.

SHA-1 algorithm uses five 32-bit variables (A, B, C, D, and E) to store new values in each round operation. It can be easily seen from [6] that four out of the five values are shifted by one position down in each round and only determining the new value for A requires computation. As shown in Fig. 1, the computation for A requires two circular right shifting and four operand addition modulo 2^{32} where the operands depend on all input values, the round constant K_t, and current message value W_t.

Fig. 3 shows in detail the architecture of message compression presented by this paper. Our architecture's characteristics are as follows. First, we use a five

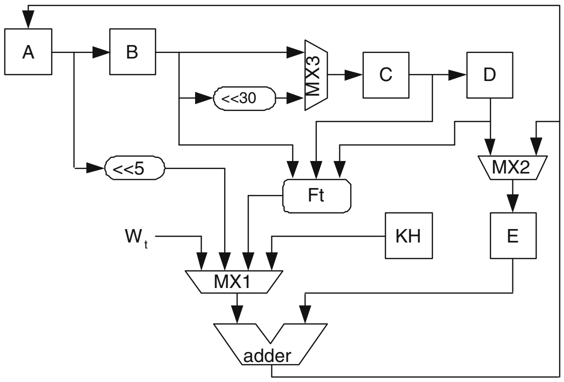


Fig. 3. Proposed architecture of message compression block

stage 32-bit shift registers because four out of the five values(B, C, D, and E) are shifted by one position down in each round.

Second, the number of adder used in the message compression is one for all the round computation. For iterative computation of value A, register E is used for the saving of the temporary addition values. Therefore, four clock cycles are required to compute one round operation as summarized follows:

- Step 1: Execute addition of register E and K_t; write result to the register E
- Step 2: Execute addition of register E and $ROT_{LEFT5}(A)$;
  write result to the register E
- Step 3: Execute addition of register E and W_t; write result to the register E
- Step 4: Execute addition of register E and $F(B, C, D)$;
  write result to the register A;
  shift down by one position of A, B, C and D;

At first, all registers are initialized and multiplexers choose path zero to load initialization constant($H_0 \sim H_4$) stored in KH. Five clock cycles are required to load initial values to each register. For optimized power consumption, we applied gated clock to all registers in data compression.

The F-function(F_t) in Fig. 3 is a sequence of logical functions. For t-th round, F_t operates on three 32-bit data (B, C, and D) and produces a 32-bit output word. The operation of F-function is shown in equation 1.

$$F(B,C,D) = \begin{cases} (B \wedge C) \oplus (\bar{B} \wedge D) & 0 \leq t \leq 19 \\ B \oplus C \oplus D & 20 \leq t \leq 39 \\ (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D) & 40 \leq t \leq 59 \\ B \oplus C \oplus D & 60 \leq t \leq 79 \end{cases} \quad (1)$$

During the final round operation, the values of the working variables have to be added to the digest of the previous message block, or specific initial values for the first message block. This can be done very efficiently with an additional multiplexer and the five stage shift registers for working variables.

KH in Fig. 3 stores initial values H_i and round constant K_t. It also stores updated H_i values, which is used as the initial values for next 512-bit data block computing. Computing the final hash value for one input message block takes five clock cycles.

3.2 Design of Message Schedule

Another important part of SHA-1 data path is a message schedule block. This block generates message dependant words, W_t, for each round of the message compression. The message schedule block is the most expensive part of SHA-1 in terms of hardware resources. Therefore, its implementation is a critical part in the design of a compact design of SHA-1 circuit. As shown in Fig.1, conventional message schedule block is usually implemented using 16 stage 32-bit shift registers for 512-bit data block processing. However, this method is inefficient to use in mobile platforms because it requires a significant amount of circuit area. This method also reduces the energy efficiency significantly.

Our message schedule block performs the function of the equation 2, where $\oplus$ means bitwise XOR and $M_t^{(i)}$ denotes the first sixteen 32-bit data of i-th data block.

$$W_t = \begin{cases} M_t^{(i)} & 0 \leq t \leq 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \leq t \leq 79 \end{cases} \qquad (2)$$

It can be easily seen from equation 2 that the input message stored in the memory is used just the first 16 round computations. In this methods, the memory is not used any more after 16 round calculations. From the viewpoint of resource efficiency, this method is inefficient and wastes too many hardware resource.

For a compact design, we propose an optimized architecture of message schedule, which enhances the resource sharing of the memory. Our approach uses only one 32-bit register(reg_w) and data selector to implement message schedule in the SAH-1 circuit. The reg_w is used to store temporary values during computation of the new W_t. The detailed structure and functional steps of proposed message schedule are shown in Fig. 4 and table 1 respectively.

The iterative functional steps of logic units for the proposed message schedule block are summarized in table 1. Four values of memory data have to be read for the calculation of W_t and the result is written back to memory in each round. This job takes 4 clock cycles. Message schedule could be calculated simultaneously during message compression, which consumes 4 clock cycles. Therefore, no additional clock cycle is required for the computation of W_t in the message schedule. The result of the new W_t, completed on the 4^{th} clock, is used for current round of message compression and is stored in memory for following

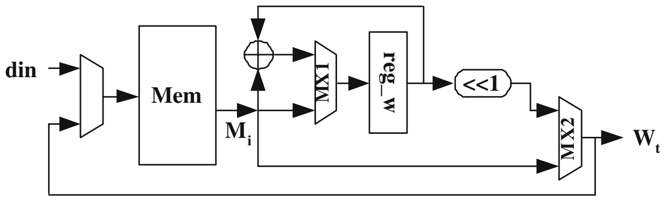


Fig. 4. Proposed architecture of message schedule block

Table 1. Functional steps for message schedule block

		Operation of circuit blocks			
Round(i)	Step	mem.out	mx1.out	reg_w.out	mx2.out
$0 \sim 15$	1	M_i	x	x	M_i
$16 \sim 79$	s1	M_{i-16}	M_{i-16}	M_{i-16}	x
	s2	M_{i-14}	reg_w $\oplus M_{i-14}$	$M_{i-16} \oplus M_{i-14}$	x
	s3	M_{i-8}	reg_w $\oplus M_{i-8}$	$M_{i-16} \oplus M_{i-14} \oplus M_{i-8}$	x
	s4	M_{i-3}	reg_w $\oplus M_{i-3}$	$M_{i-16} \oplus M_{i-14} \oplus M_{i-8} \oplus M_{i-3}$	$\ll 1$(reg_w)

round calculation of W_t. Dedicated hard wired logic is used for computation of necessary address.

The memory in Fig. 4 is register-based and single port 512-bit memory that is installed by using standard logic cells. In order to minimize the power consumption, the internal registers of memory are disabled when they are not being used, thus reducing the amount of unwanted switching activity. Additional multiplexer is used to select input data between initial input message and new word data W_t.

3.3 Design of Software Interface

The software interface of the SHA-1 hardware module is composed of the hardware interface, device driver and application program.

The hardware interface provides the communication between the application and device driver. It delivers the requirement generated in an application to the device driver. The application interface receives the requirements of an application. The device driver receives an allocation about DMA and Memory Mapped I/O and registers the drivers on the kernel for data transmission with applications.

If an interrupt informing the data arrival is generated at hardware, device driver accesses the register of the designated hardware and saves the data in the session buffer. Then, application access the session buffer and reads corresponding data.

4 Implementation Results and Comparison

4.1 Synthesis Results

The described architectures in our design were first described in VHDL, and verified through functional simulation using Active HDL, from Aldec Inc. In order to evaluate the proposed SHA-1 design, we used the Synopsys synthesize flows on a Sun Solaris platform. The target technology library is the Samsung Electronics STD110, featuring 0.25μm CMOS standard process and 2.5V core voltage. After synthesis, Synopsys Power Compiler was used to calculate the overall power dissipation of our design. Although the maximum operating frequency obtained using timing analysis is 114 MHz, we use 25 MHz as the operating frequency to evaluate the consuming power of our circuit because the system clock of most mobile phones is about 20 MHz.

The activity of the netlist was estimated for various test messages so that it could be considered as reasonable values. We would like to emphasize that our design is on the algorithmic and architectural level. Implementing our designs using an low power ASIC library or a full custom design will enable higher power savings.

Table 2 shows the circuit area and power estimation quantities of our design based on the logic blocks. The proposed SHA-1 design(including interface and memory) consumes an area of 6,812 gates and needs 355 clock cycles(includes data input and output cycles) to compute the hash of 512 bits of data. The total power consumption at 25 MHz is about 2.96 mW.

We present the comparison of the proposed design with some previous works for SHA-1 implementation in table 3. It can easily be seen from table 3 that our implementation uses minimum hardware resources than the design of [7, 8, 9, 13]. Also, our architecture satisfies the processing speed requirement for general handheld devices.

Table 2. Logic blocks, complexity, and power consumptions from Samsung 0.25 μm CMOS process

Logic block	Circuit area		Power consumption	
	gates	percentage	mW@25MHz	percentage
Interface	468	6.8	0.054	1.8
memory	3,434	50.4	0.895	30.2
message schedule	314	4.6	0.17	5.7
controller	350	5.1	0.228	7.7
reg_a∼e	876	12.9	0.336	11.3
adder	200	3	0.514	17.4
message compression	1,170	17.2	0.764	25.9
Total	6,812	100%	2.961	100%

Table 3. Comparison with previous works of SHA-1 implementation

Reference	Platform	Hardware size	Frequency (MHz)	clock cycles	Throughput (Mbps)
This work	0.25μm ASIC	6,812 gates	114	355	164
	xc2v3000	735 slices	65.3		94.6
[7]	0.25μm ASIC	20,536 gates	143		893
[8]	0.6μm ASIC	10,900 + RAM	59	255	119
	xcv300E	2,008 slices	42.9		86
[9]	0.18μm ASIC	20,000 gates	166	81	1,000
[13]	xc2v3000	1,550 slices	38.6	22	899.8

4.2 System Level Test

There exist several commercial TPM chips implementing SHA-1 algorithm [18, 19]. It is difficult to directly compare these chips with our architecture because those commercial chips dose not open the system level computing time.

To evaluate actual performance of our design, we designed evaluation system shown in Fig. 5. We used two Xilinx's Virtex2-pro xc2vp20 FPGA chips(FPGA1 & FPGA2) for fast development and easy test. FPGA1 contains microprocessor, dedicated program memory, and dedicated RAM. As a microprocessor, we used EISC3208H core module from ADchip Corp. The user program and test

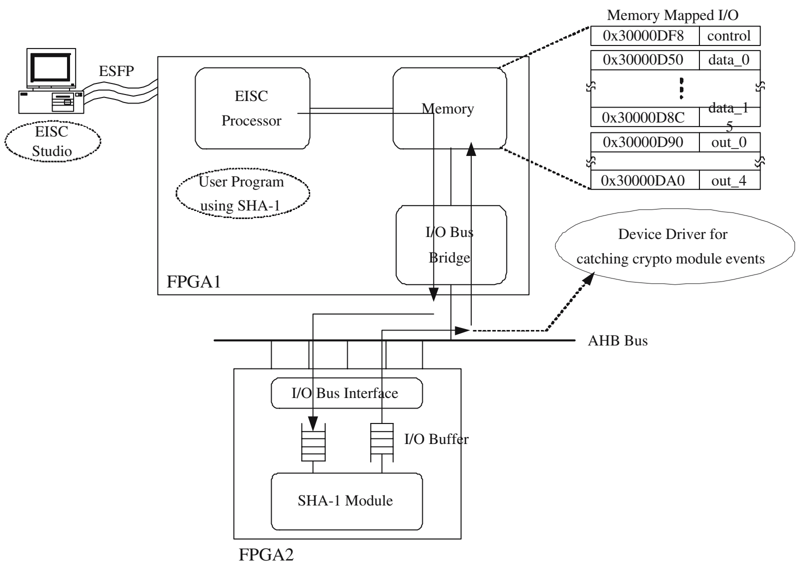


Fig. 5. Evaluating system block diagram

Table 4. Comparison with commercial TPM chip based on system level SHA-1 computations

Reference	Operating Freq.	data length	Computing time
software only	20MHz	1M-bits	<3.02 sec
SSX35A [18]	33MHz	1M-bits	<258 ms
This work	20MHz	1M-bits	<106 ms

code are developed in the EISC Studio at the desktop and downloaded to the microprocessor through EISC Serial Flash Programmer (ESFP).

FPGA2 is used for dedicated cryptographic modules. The developed SHA-1 module is downloaded into FPGA2 and connected to an AHB slave bus system. Polling was used to ensure reliable communications between microprocessor and SHA-1 core. The microprocessor signals the SHA-1 core to start after it fills the input message and polls the FPGA2 to find out that the corresponding operation is finished. Then, the microprocessor fills the message into the memory of SHA-1 core again.

Table 4 presents the comparison result of our design with the most representative TPM chip [18] having the same functionality. Although the operating frequency of the proposed implementation is lower than that of [18], the computing time of 1M-bit message by the proposed SHA-1 circuit on the system level is faster than that of commercial TPM chip [18] designed for desktop computers.

5 Conclusions

This paper presents a compact architecture for low-cost SHA-1 hardware. The presented architecture is a highly effective architecture that can implement the message compression and message schedule in SHA-1 algorithm with a minimum resource usage. The presented architecture has a chip area of 6,812 gates and has a current consumption of 2.96mW at a frequency of 25MHz. The proposed design requires less than 355 clock cycles to compute the message digest of 512 bits of data.

Compared to the other academic implementations and some commercial TPM chips supporting SHA-1 hardware module, the proposed design demonstrated the smallest area in terms of logic gates. Furthermore, according to the implementation result of system level test, the computation speed of the proposed design is at least 270% faster than that of commercial TPM chip supporting SHA-1 circuit, while using lower operating frequency.

The results of power consumption, throughput, and functionality make our SHA-1 cryptographic hardware suitable for trusted mobile computing and other low-end embedded systems that urge for high-performance and small-sized solutions.

Acknowledgements

This work was supported by the IT R&D program of the Ministry of Knowledge Economy, Korea. [2006-S-041-02, "Development of a common security core module for supporting secure and trusted service in the next generation mobile terminals"]

The second author of this research was supported by the Ministry of Knowledge Economy, Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2008-C1090-0801-0016) .

References

1. Trusted Mobile Platform NTT DoCoMo, IBM, Intel, Trusted Mobile Platform: Hardware Architecture Description Rev1.0, Trusted Computing Group (2004)
2. Ravi, S., Raghunathan, A., Porlapally, N.: Securing wireless data: system architecture challenges. In: Proccedings of the 15th International Symposium on System Synthesis, pp. 195–200 (2002)
3. MPC860 Product Summary, http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC860
4. ARM7 Product Summary, http://www.arm.com/products/CPUs/families/ARM7family.html
5. Raghunathan, A., Ravi, S., Hattangady, S., Quisquater, J.: Securing Mobile Appliances: New Challenges for the System Designer. In: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, DATE 2003 (2003)
6. NIST: Secure Hash Standard FIPS-Pub 180-1, National Institute of Standard and Technology (1995)
7. Ming-yan, Y., Tong, Z., Jin-xiang, W., Yi-zheng, Y.: An Efficient ASIC Implementation of SHA-1 Engine for TPM. In: IEEE Asian-Pacific Conference on Circuits and Systems, pp. 873–876 (2004)
8. Dominikus, S.: A Hardware Implementation of MD4-Family Hash Algorithms. In: IEEE International Conference on Electronic Circuits and Systems, vol. 3, pp. 1143–1146 (2002)
9. Helion IP Core Products, Helion Technology, http://www.heliontech.com/core.htm/
10. Zibin, D., Ning, Z.: FPGA Implementation of SHA-1 Algorithm. In: 5th IEEE International conference on ASIC, pp. 1321–1324 (2003)
11. Michail, M.K., Kakarountas, A.P., Milidonis, A., Goutis, C.E.: Efficient Implementation of the Keyed-Hash Message Authentication Code (HMAC) using the SHA-1 Hash Function. In: 11th IEEE International Conference on Electronics, Circuits and Systems, pp. 567–570 (2004)
12. Selimis, G., Sklavos, N., Koufopavlou, O.: VLSI Implementation of the Keyed-HASH Message Authentication Code for the Wireless Application Protocol. In: 10th IEEE International Conference on Electronics, Circuits and Systems, pp. 24–27 (2003)
13. Diez, J.M., et al.: HASH Algorithms for Cryptographic Protocols: FPGA Implementations. In: 10th Telecommunication Forum, TELEFOR 2002 (2002)
14. Kang, Y.-k., et al.: An Efficient Implementation of Hash Function processor for IPSec. In: IEEE Asia-Pacific Conference on ASIC, pp. 93–96 (2002)

15. Michail, H.E., Kakarountas, A.P., Selimis, G.N., Goutis, C.E.: Optimiizing SHA-1 Hash Function for High Throughput with a Partial Unrolling Study. In: Paliouras, V., Vounckx, J., Verkest, D. (eds.) PATMOS 2005. LNCS, vol. 3728, pp. 591–600. Springer, Heidelberg (2005)
16. Sklavos, N., Dimitroulakos, G., Koufopavlou, O.: An Ultra High Speed Architecture for VLSI Implementation of Hash Functions. In: Proc. Of ICECS, pp. 990–993 (2003)
17. Huang, A.L., Penzhorn, W.T.: Cryptographic Hash Functions and Low-Power Techniques for Embedded Hardware. In: IEEE ISIE 2005, pp. 1789–1794 (2005)
18. SSX35A, Sinosun (2005), https://www.trustedcomputinggroup.org/ShowcaseApp/sh_catalog_files//SSX35%20Mar.05.pdf#search=%22SSX35A
19. AT97SC3203 Advance Information Summary, Atmel corp. (2005), http://www.atmel.com/dyn/products/product_card.asp?part_id=3736

An Improved Distributed Key Management Scheme in Wireless Sensor Networks

Jun Zhou and Mingxing He

School of Mathematics and Computer Engineering, Xihua University,
Chengdu, China, 610039
xxyyxw@hotmail.com, he_mingxing64@yahoo.com.cn

Abstract. A distributed key management scheme and the robust continuity of group key establishment are realized in this paper. This scheme derives the concept of seed members who are responsible for the management of users' joining and revocation operations by using secret sharing techniques and Modified Blom's symmetric key establishment mechanism. Besides, a novel seed member alternation mechanism which can efficiently resist of node capture attacks and revocation attacks leaves the resources much more balanced among the sensor nodes to prolong the lifetime of WSNs. Finally, The key continuity enables all the sensors to establish and update both of the pair-wise key and group key with the same key pre-distribution information, which significantly decreases the resource consuming of key management in distributed sensor networks. By comparison, this scheme outperforms most of the existing schemes in terms of network security, key connectivity and complexity of storage as well as communication.

Keywords: Key management, Distributed wireless sensor networks, Secret sharing, Continuity.

1 Introduction

Wireless sensor networks is widely used in fields of education, military, medicine, transportation and environment monitoring. In order to provide the secure communication in wireless sensor networks, key management is significantly essential. However, several conventional key management technologies cannot be applied directly because of the limitation of resources possessed by the sensor nodes such as storage, communication and computation. Consequently it has attracted a series of attention and research all over the world[1][2]. Key management in WSNs can be classified into pair-wise key management and group key management. In pair-wise key management schemes[4 − 8], each pair of communication nodes is able to establish a secure link by encrypting and decrypting the messages using the shared pair-wise key. While group key management schemes[14][15] can also be further classified into central topology adaptive schemes and the distributed topology adaptive ones. In the former, each legitimate node can use the pre-distributed information combined with what they acquired in the broadcast messages transmitted by the group manager to calculate the same group key. While in the latter, peer nodes can take advantage of the contributory information transmitted among themselves to negotiate a shared group key.

Based on the characteristics of WSNs, a reliable key management scheme is also responsible for discovering and revoking the compromised nodes timely. Recently, the key updating scheme with the assistance of TTP proposed by Eschenauer in 2002[6] and the distributed key updating scheme proposed by Chan et al.[11] in 2003 are both able to revoke the compromised nodes from the network, but leaving the key connectivity significantly decreased. The reason is that both of them adopted the random key pre-distributed method, causing that a certain number of pair-wise keys exposed to the adversary are the same as the ones used by the innocent nodes. In 2007, a reliable key updating protocol based on central-topology was proposed by Mi Wen and Kefei Chen[3]. It also realizes the key continuity between pair-wise and group key management. However, due to randomness, an large area of deployment and the unique capacity of revoking compromised nodes belonging to the group manager, it is not likely for the group manager to manage all the revoking tasks timely. In other words, some wiser adversaries would have captured even more nodes and other private information during the time period when the group manager forwards the revoking messages through such a large area. Last but not least, the group manager will also become the Achilles' heel, attracting the most powerful attacks initiated by the adversaries.

The main contribution of this paper is proposing a novel pair-wise key management scheme in the distributed WSNs and realizing the continuity of the group key establishment.

Firstly, it takes advantage of the efficient monitoring mechanism realized by the cooperation of the neighbor nodes of the compromised nodes. These neighbor nodes are named legitimate appealing nodes in this paper to fulfill the task of distributed revocation. This scheme is based on the new concept of seed members responsible for the pair-wise key establishment and updating. A certain number of legitimate appealing nodes can use the Modified Blom's pair-wise key establishment scheme and secret sharing scheme to revoke the compromised nodes efficiently and timely. The previous seed members can be replaced by other nodes to balance the energy consuming of the network, preventing the sub-networks from isolation.

Secondly, in order to prevent the private key information of legitimate appealing nodes and the left innocent nodes from exposure, a novel concept of private shadow matrix is also proposed. They can be used to efficiently mask the original key information of the sensor nodes, making the distributed revocation mechanism come true without any exposure.

Finally, the continuity of pair-wise key and group key is also applied to this scheme to realize the establishment of both of them without decreasing the key connectivity and consuming additional resources simultaneously, prolonging the lifetime of the WSNs.

The rest of this paper is organized as follows. The network model and the attack model of this scheme are described in section 2. A novel distributed pair-wise key management scheme is proposed in section 3 and the characteristic of continuity applied to the group key management is presented in section 4. Security and performance are analyzed in section 5 and conclusions are presented in section 6.

2 Network and Attack Models

It is assumed that a distributed topology with dynamic nodes is applied to WSNs in this paper, which is different from the central topology consisting of base stations, cluster heads and sensor nodes. Each pair of nodes establishes pair-wise keys by using modified Blom's key establishment scheme[5]. All of the sensor nodes possess of nearly the same capacity of calculation, storage and communication.

The attack model can be described as follows.

(1) The adversary has whole-network communication capacity, namely, the adversary can transmit and receive all the messages whenever and wherever she/he wants.

(2) λ-attack limitations. It is assumed that each adversary individually or the collaboration of the adversaries can compromise at most λ sensor nodes. Once a sensor node is compromised, all the key establishment information and routing information stored in the node can be acquired by the adversaries. They can be used to initiate further attacks even revocation attacks[4] in which an adversary can use the distributed node revocation protocol to selectively revoke uncompromised nodes from the network.

(3) It is assumed that the compromised nodes can selectively drop packets which they have received, but the adversary cannot jam or delay local communications in the network whose sources and destinations are both uncompromised sensor nodes. The adversary can not block or delay multi-hop broadcast messages, either neighborhood wide or network wide. The adversary is also unable to partition the network via node capture attacks. Therefore, all the transmissions of key establishment and updating information can not be hampered by this kind of attacks.

3 Pair-Wise Key Management Scheme

In this section, a novel and reliable pair-wise key management scheme is proposed based on the Modified Blom's pair-wise key establishment mechanism[5] and PRKU pair-wise key updating scheme[3]. In a network situation without the existence of TTP, this scheme realizes the dynamically joining and leaving of sensor nodes efficiently. The alternation of controlling authorities by the co-operation of $\lambda+1$ member nodes can update the seed members reliably. λ is the trapdoor of Modified Blom's mechanism[5]. The seed members are defined as $\lambda+1$ members who are responsible for the generation of $\lambda+1$ different seeds of $\lambda+1$ hash functions included in the initial private symmetric matrix respectively.

Table 1. Notation for Our Scheme

Notation	Description
D_0	$(\lambda+1, \lambda+1)$ initial private symmetric matrix, used for pre-distribution for initial N_0 sensor nodes
N_0	The initial number of sensor nodes
L_0	$(\lambda+1\,\lambda+1)$ private symmetric shadow matrix, generated by the system in the key pre-distribution phrase
G_0	$(\lambda+1, N_0)$ public matrix, each column of which represents the identity information of initial members
U_0	equals to $(D_0 \times G_0)^T$, a $(N_0, \lambda+1)$ private matrix, each row of which is separately stored in initial members
$U_0^{'}$	equals to $(L_0 \times G_0)^T$, a $(N_0, \lambda+1)$ private matrix, used for masking original key information of legitimate appealing nodes in distributed revocation and stored in each node in initialization
R_0	equals to $U_0 + U_0^{'}$, revocation information transmitted among legitimate appealing nodes
$H(\cdot)$	An hash function
$MAC(\cdot)$	An data integration verifying function without a secret key

By doing this, this scheme can adapt itself to the distributed character and resist to node capture attacks better, outperforming the previous schemes[6][8][11] in whatever respective of storage, calculation or communication.

All the notations that will be used in this scheme are illustrated in Table 1.

Notations such as $D_t, N_t, L_t, G_t, U_t, U_t^{'}, R_t$ denote the same meanings as explained in Table 1 but during the time period $T = t$. The introduction of Modified Blom's symmetric key establishment scheme is illustrated as follows. Each element in matrixes D_0, L_0, G_0 is selected over a finite field $GF(q)$ where q is a prime number and $q < N_0$. $\lambda+1$ prime seeds $s_i^0 (i = 1, 2, \cdots, \lambda+1)$ over $GF(q)$ are generated by the system and only known to the system itself. Then the system creates a random $(\lambda+1) \times (\lambda+1)$ matrix D_0 over $GF(q)$. Each row of D_0 is composed of hash values of the prime seeds. The elements of the matrix D_0 are generated as follows.

$$\begin{aligned} d_{0_{ij}} &= H^i(s_j)\ (i, j \in \{1, 2, \cdots, \lambda+1\}\ and\ i > j) \\ d_{0_{ij}} &= H^j(s_i)\ (i, j \in \{1, 2, \cdots, \lambda+1\}\ and\ i \leq j) \end{aligned} \tag{1}$$

Where $d_{0_{ij}}$ represents the element in D_0. As an example of the initial private symmetric matrix, D_0 with size 3×3 can be demonstrated as follows:

$$D_{0_{(3\times3)}} = \begin{pmatrix} H^1(s_1) & H^2(s_1) & H^3(s_1) \\ H^2(s_1) & H^2(s_2) & H^3(s_2) \\ H^3(s_1) & H^3(s_2) & H^3(s_3) \end{pmatrix}$$

Finally, the pair-wise key establishment based on Blom symmetric key construction mechanism [5] can be described as follows. It is assumed that K_0 is the pair-wise key matrix generated ultimately.

$$K_0 = (D_0G_0)^TG_0 = G_0^TD_0^TG_0 = G_0^TD_0G_0 = (U_0G_0)^T = K_0^T \tag{2}$$

Thus K_0 is also a symmetric matrix and $k_{ij} = k_{ji}$, where k_{ij} is the element of K_0 at i-th row and j-th column. We take k_{ij} or k_{ji} as the pair-wise key established between node i and node j.

The pair-wise key management scheme includes the following steps:

(1) Off-line Initialization: Each node SN_i is pre-distributed with a row $U_{0_{l_i}}$ $(1 \leq l_i \leq N_0)$ from the initial original matrix U_0. In order to achieve stronger security, it is not necessary for the nodes to store the rows corresponding to their own identities, eg. l_i is equal to the identity of the node SN_i , only presenting the row of private key information stored in the node SN_i by the notation $U_{0_{l_i}}$. The corresponding row $U'_{0_{l_i}}$ $(1 \leq l'_i \leq N_0$ and $l'_i = l_i)$ from the private shadow matrix U'_0 is also stored in the node SN_i.

(2) Pair-wise Key Establishment: The initial N_0 nodes establish pair-wise keys by using modified Blom's pair-wise key establishment mechanism illustrated in section 3.

(3) Pair-wise Key Updating: It is necessary to update the established pair-wise keys every time period T, taking $t = m$ for example.

(3.a)$\lambda + 1$ seed members choose $\lambda + 1$ different seeds of $\lambda + 1$ hash functions $s_1^m, s_2^m, \cdots, s_{\lambda+1}^m$, calculate the private hash values $H^j(s_j^m)(j = 1, 2, \cdots, \lambda + 1)$ and broadcast the hash values $H^{j+1}(s_j^m)(j = 1, 2, \cdots, \lambda + 1)$.

(3.b)After receiving other λ broadcast messages, each seed member SN_j firstly checks whether some seed members have selected the same seed of hash functions as its own by comparing the received broadcasted messages with $H^{j+1}(s_j^m)$. If they are not equal to each other and all the seeds of $\lambda + 1$ hash functions are assured to be different, SN_j calculates the hash values she/he needs according to the initial private symmetric matrix D_0 generating algorithm described in section 3 combined with its own private value $H^j(s_j^m)$. Then she/he can recover the j-th row in the new private symmetric matrix D_m. Finally, SN_j generates all the elements in each row of U_m for other nodes.

$$U_m(i, j) = \sum_{k=1}^{\lambda+1} D_m(j, k) \times G_m(k, i)(i = 1, 2, \cdots, j-1, j+1, \cdots, N_m) \tag{3}$$

(3.c)Each seed member transmits partial key information encrypted by the established pair-wise keys at time $t = m - 1$ to other nodes respectively.

$$\begin{gathered} SN_j \rightarrow SN_i : E_{K_{j,i}^m}\{U_m(i, j), j, MAC(U_m(i, j), j)\} \\ (i = 1, 2, \cdots, j-1, j+1, \cdots, N_m) \end{gathered} \tag{4}$$

(3.d)After receiving $\lambda + 1$ pieces of encrypted partial key information, each node decrypts them and recovers its own key information by rearranging all the $\lambda + 1$ pieces of partial key information according to the order j which is included in the message it has received from SN_j. By doing this, the new private matrix U_m has been updated and all the N_m nodes can reestablish the new pair-wise keys accordingly. From then on, two pieces of private key information have been stored in each sensor node SN_i. One is the current key information $U_{m_{l_i}}(1 \leq l_i \leq N_m)$ varying from time to time and the other is the initial key information $U_{0_{l_i}}(1 \leq l_i \leq N_0)$ used to generate the pair-wise keys for encrypting the partial key information transmitted to the joining nodes at $t = n$ because the newly joining nodes can only be pre-distributed with the initial off-line key information in U_0 before deployment.

(4) Member Join: It is assumed that SN_n wants to join the network at time $t = n$. The new member node has been pre-distributed with a row $U^{'}_{n_{l_i}}(l_i = N_n + 1)$ from the private shadow matrix $U^{'}_n$ before its deployment and the identity of the joining member is arranged as the $(N_n + 1)$-th column in the public matrix G_n. Other steps are the same as the ones in the stage of off-line initialization. After deployment, the steps are described as follows.

(4.a)SN_n accomplishes the initial pair-wise key establishment with the existed nodes according to the modified Blom's pair-wise key establishment mechanism illustrated in section 3 by using the initial key information in U_0 pre-stored in it. The initial pair-wise keys are notated as $K^0_{j,n}(j = 1, 2, \cdots, \lambda + 1)$ and used to encrypt and decrypt the current pair-wise key establishment materials in steps 4.c and 4.d.

(4.b) All the $\lambda + 1$ current seed nodes generate the additional $\lambda + 1$ elements $U_n(N_n+1, j)(i = 1, 2, \cdots, \lambda+1)$ for the joining node SN_n respectively. The j-th seed node is responsible for generating the j-th element in the $(N_n + 1)$-th row in U_n. The algorithm is as follows.

$$U_n(N_n + 1, j) = \sum_{k=1}^{\lambda+1} D_n(j, k) \times G_n(k, N_n + 1)(i = 1, 2, \cdots, \lambda + 1) \qquad (5)$$

(4.c) Each seed node transmits the partial key information to the joining node in the form of encryption respectively.

$$SN_j \rightarrow SN_n : E_{K^0_{j,n}}\{U_n(N_n + 1, j), j, MAC(U_n(N_n + 1, j), j)\} \qquad (6)$$

(4.d) After receiving the $\lambda + 1$ pieces of partial key information, the joining node SN_n decrypts them and recovers its own private key information in U_n according to the element order j included in the transmitted message in step (4.c). Therefore, the joining node SN_n can establish the current pair-wise keys shared with the existed members.

(5) Member Revocation: It is assumed that no less than $\lambda + 1$ legitimate neighbor nodes of SN_i vote to appeal SN_i cooperatively at time $t = t\prime$. As a result, they will successfully revoke the compromised node SN_i from the network. We named the no less than $\lambda + 1$ legitimate neighbor nodes of SN_i as a group of legitimate appealing nodes $SN_s(eg.\ s \in \{i, i + 1, \cdots, i + \lambda \cdots\})$. In order to prevent the original key information of other appealing nodes and left innocent nodes from being exposed in the process of distributed revocation, a private shadow matrix $U^{'}_{t\prime}$ is introduced, the meaning of which has been illustrated in Table 1. The steps of distributed revocation are described as follows.

(5.a) Once a legitimate appealing node SN_s discovers a compromised node SN_i, she/he will ask SN_i to broadcast its check value

$$V_i^{t'} = H(R_{t'_{i0}} \parallel R_{t'_{i1}} \parallel \cdots \parallel R_{t'_{i\lambda}}) \tag{7}$$

all over the network. If SN_i refuses to submit its check value during a certain period of time, all the nodes in the network will consider it as an authentic compromised node and delete all the established pair-wise keys between SN_i and themselves. On the other hand, if the check value has been broadcasted by SN_i, each legitimate appealing node SN_s calculates its own masked key information

$$R_{t'_s} = U_{t'_s} + U'_{t'_s} \tag{8}$$

respectively and transmits the following encrypted message to the other appealing nodes $SN_k (k \in S$ and $k \neq s)$.

$$SN_s \rightarrow SN_k : E_{K^{t'}_{s,k}} \{ID_s, ID_i, R_{t'}(s), MAC(ID_s, ID_i, R_{t'}(s))\} \tag{9}$$

(5.b) After receiving at least λ messages from other legitimate appealing nodes, each member of SN_s firstly decrypts them. If there are no less than λ messages appealing the same node SN_i, she/he calculates $R_{t'_{s'}} (s' \in I = \{1, 2, \cdots, N_{t'}\} \setminus S)$ with at least $\lambda + 1$ pieces of information $R_{t'_s} (s \in S = \{i, i+1, \cdots, i+\lambda \cdots\})$ according to modified Blom's symmetric key construction mechanism illustrated in section 3. Then, each member of SN_s calculates

$$k'^{t'}_{ss'} = R_{t'_{s'}} \times ID_s \tag{10}$$

and

$$k'^{t'}_{si} = U'_{t'_s} \times ID_i \tag{11}$$

Besides, each member of SN_s makes such subtraction as blindness removing

$$k^{ct'}_{ss'} = k'^{t'}_{ss'} - k'^{t'}_{si} \tag{12}$$

and compares all the values $k^{ct'}_{ss'}$ with the pair-wise key established between SN_i and itself until a pair of equal values is found. The corresponding row $R_{t'_i}$ in $R_{t'_{s'}} (s' \in I = \{1, 2, \cdots, N_{t'}\} \setminus S)$ used to calculate the pair of equal values is just the masked key information of the compromised node SN_i. Finally, each member of SN_s calculates

$$V_i^{st'} = H(R^s_{t'}) = H(R^s_{t'_{i0}} \parallel R^s_{t'_{i1}} \parallel \cdots \parallel R^s_{t'_{i\lambda}})(s \in S) \tag{13}$$

and broadcasts $(ID_s, ID_i, H(R^s_{t'_i}))(s \in S)$.

(5.c) All the sensor nodes in the network compare the received check value $V_i^{st'} (s \in S)$ in step (5.b) with the check value $V_i^{t'}$ submitted by the compromised node itself in step (5.a). They are able to judge that SN_i has been legitimately revoked from the network and delete all the established pair-wise keys between SN_i and themselves if and only if at least one of the following two cases exists. (5.c.1) Each sensor node has received at least $\lambda + 1$ check values $V_i^{st^{'}}$ $(s \in S)$ from $\lambda + 1$ different legitimate appealing nodes, each of which is the same as the submitted check value $V_i^{t^{'}}$. (5.c.2) Although the $\lambda + 1$ received check values $V_i^{st^{'}}$ $(s \in S)$ are not equal to $V_i^{t^{'}}$, they are equal to each other in-between them.

(5.d)If the revoked node SN_i belongs to the current group of $\lambda + 1$ seed nodes $SN_p(p \in P = \{u, u+1, \cdots, u+\lambda\})$, all the left seed nodes $SN_{p'}(p^{'} \in P \backslash \{i\})$ will negotiate to determine a successive node $SN_{i'}$ to replace SN_i by the technique of secret sharing. Meanwhile, reserved resources, locations and other security-relevant characteristics of the sensor nodes will also be taken into consideration together in the cooperative selection. The process resembles the one of member revocation. After a new seed node is selected and verified correctly, all the key information and pair-wise keys will be updated as follows.

(5.d.1) The new seed node $SN_{i'}$ generates a new seed $s_{i'}^{t^{'}}$ for the hash functions, calculates a private hash value $H^i(s_{i'}^{t^{'}})$ and broadcasts $H^{i+1}(s_{i'}^{t^{'}})$.

(5.d.2)$SN_l(l \in N_{t'}^{new})$ updates key information. A new group of $\lambda + 1$ seed nodes calculates the hash values she/he needs according to the initial private symmetric matrix D_0 generating algorithm described in RPKU scheme[3] combined with the remained hash values to recover the j-th row of $D_{t'}^{new}$ and updates the key information for $SN_l(l \in N_{t'}^{new})$ as follows.

$$U_{t'}^{new} = U_{t'}(l, j)(j < i) \tag{14}$$

$$U_{t'}^{new} = U_{t'}(l, j) + \sum_{k=i}^{\lambda+1}(H^k(s_{i'}) - H^k(s_i)) \times G_{t'}^{new}(k, l)(j = i) \tag{15}$$

$$U_{t'}^{new} = U_{t'}(l, j) + (H^j(s_{i'}) - H^j(s_i)) \times G_{t'}^{new}(i, l)(j > i) \tag{16}$$

After that, all the nodes $SN_l(l \in N_{t'}^{new})$ can take advantage of the updated key information to reestablish pair-wise keys.

4 Group Key Management Scheme

(1) Group Key Establishment: It is assumed that $SN = \{SN_1, SN_2, \cdots, SN_n\}$ are n peer-to-peer nodes intended to negotiate a group session key in the distributed sensor networks. SID_j is denoted as the session number of the j-th session. Group key is established as follows.

Each member node $SN_i(1 \leq i \leq n)$ is pre-distributed with a row in U_0 at the stage of off-line initialization. After deploying, SN_i calculates the pair-wise keys $K_{i,i-1}^0$ and $K_{i,i+1}^0$ between two neighbor nodes and itself. Then she/he also calculates the value

$$K_{i,i}^{0} = \sum_{\varepsilon=1}^{\lambda+1} u_{0_{i\varepsilon}} g_{0_{\varepsilon i}} \tag{17}$$

where u_0 and g_0 represent the elements in the initial private matrix U_0 and initial public member matrix G_0 respectively. Finally, each member node SN_i transmits $K_{i,i}^{0}$ encrypted by established pair-wise keys to its neighbor nodes respectively and uses BD protocol[13] to accomplish the following steps of group key establishment.

(2) Group Key Updating: Group key management in the distributed WSNs is not only required to provide an efficient group key establishment mechanism, but also should be responsible for considering the case of dynamic management of the group key. The basic security requirements are as follows. (a)secrecy of group key (b)forward security and(c)backward security. Consequently, it is necessary to update the group key in case that there are nodes revoked from the network for energy-exhausting, being compromised by adversaries and joining the group. After successful join or revocation, other innocent nodes can update their key information respectively using the updating algorithm illustrated in Section 3, transmit the updated contributory key information $K_{i,i}^{'}$ to its neighbor nodes in the encrypted form and have the session number SID_j updated. The steps of group key updating resemble the process of group key establishment.

By the above analysis, we have seen that our protocol has robust continuity between the pair-wise key and group key management. Even if there are a variety of node addition and eviction in the network, our scheme can also maintain the robust continuity over time.

5 Analysis

We analyze the proposed distributed pair-wise key management scheme to verify that our scheme is a reliable one against dynamic topology changes through such three evaluation metrics of the experiments as network security, key connectivity and scalability. Then, we illustrate that the proposed scheme is resilient to revocation attack and the continuity enables the nodes to establish and update both of the pair-wise key and group key with the same pre-distributed secrets. Finally, we analyze the communication and computation overhead for our scheme.

5.1 Security Analysis

Network Security is measured by the resilience to node capture attack and requires that it is not possible for the adversary to acquire information of the innocent sensor nodes and any other links of the rest WSNs even though a certain number of nodes have been captured. The higher resilience exists, the lower number of compromised links is exposed[12].

In this section, we study the resilience by calculating the probability of the exposure of established pair-wise keys between the innocent nodes under node capture attacks. Figure 1 shows that the key connectivity of our scheme at

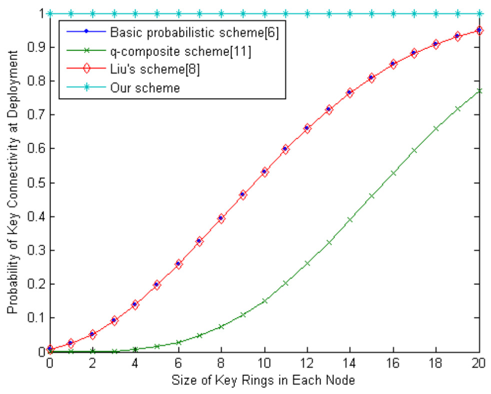


Fig. 1. The probability of key connectivity at deployment v.s. size of key rings in each node (S=s=P=170,q=2)

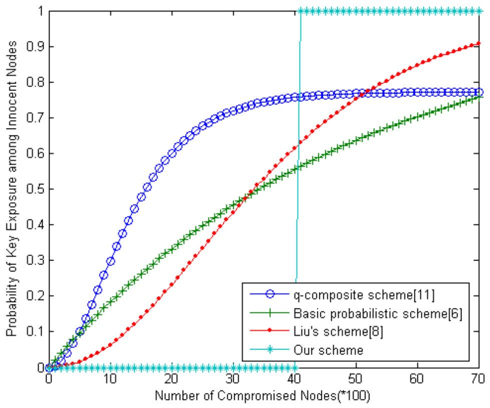


Fig. 2. Comparison of the resilience to compromise attack (q=2,m=20,S=170, k=2000,P=10^5,$s^{'}$=4,s=55,t=12,λ=40)

deployment remains 100% no matter of the size of the key rings, while the key connectivity of other random key pre-distribution schemes[6][8][11] is much lower than ours because it is not destined for each pair intended to establish a pair-wise key to share a common key in their key rings during initialization. Figure 2 shows the resilience against node capture attacks of the basic probabilistic scheme[6], the q-composite scheme[11], the symmetric bi-polynomial based random key pre-distribution scheme[8]and our scheme. The figure apparently shows that as the number of compromised nodes increases until it reaches the trapdoor of the scheme taking λ=4000 for example in Fig.2, the probability of the exposure of the established pair-wise keys between innocent sensor nodes remains zero. The reason is that each established pair-wise key is independent, which means the information the adversary acquired from the private matrix U_t and the private shadow matrix $U_t^{'}$ corresponding to the compromised node is merely relevant to the pair-wise key establishment of the compromised node itself rather than others. This kind of uniqueness can be realized because all the seeds of $\lambda+1$ hash functions selected by $\lambda+1$ seed member nodes in (3.a) are assured to be different from each other and then all the elements generated in the initial private symmetric matrix are different according the the collision resistance property of hash functions. In addition, the assumption of λ-attack ability of the adversary also prevents the adversary from calculating all the private key information of the innocent nodes. Finally, any seed nodes' compromise triggers a key information updating process and consequently, the knowledge of the compromised seed node gives the adversary no priority in deriving the key information of the innocent nodes. By comparison, it is apparent that our scheme outperforms the others. In the probabilistic schemes[6][11], once the number of compromised nodes increases, the fraction of the affected communication links in the rest of the network increases quickly. The reason is that in probabilistic schemes the same key may be pre-distributed in the key rings of a certain number of sensor nodes and be used by several different pairs of sensors. As a result, some sensor nodes' compromise may made the pair-wise keys established between the innocent nodes exposed. For the hybrid schemes[8], there are no key updating approaches. Thus, when the number of compromised nodes is more than their thresholds, all pair-wise keys of the entire network will be compromised.

Revocation attack is defined as an attack where an adversary uses the distributed node revocation protocol to selectively revoke uncompromised nodes from the network. In our scheme, there are two methods designed against this kind of attack. On one hand, it is assumed that a certain number of neighbor nodes of a fixed node are selectively compromised by the adversaries who will ally to pretend to be these compromised nodes as legitimate appealing members. In the revocation attack, it is probable for these pseudo appealing nodes to broadcast the check value $H(R_{t'}(i"))$ of the uncompromised node $SN_{i"}$ to other nodes who will mistake $SN_{i"}$ for the compromised node. However, in our scheme only λ adversaries are assumed to have the ability of initiating this kind of cooperation at most, as a result, it will fail when the left nodes judge whether $SN_{i"}$ is the objected revoked members according to the two metrics (5.c.1) and (5.c.2) described above. Because in either metric, at least $\lambda+1$ same check values received are required. On the other hand, in the last stage of the distributed revocation we are able to adopt the same algorithm described in (5.d.2) to select $\lambda+1$ new seed nodes from the left $N_{t'}^{new} - \lambda - 1$ nodes to replace the current ones, making the key information acquired in the compromised nodes in vain in computing the pair-wise keys established afterwards.

The proposed pair-wise key establishment scheme also retains the property of forward security and backward security. According to the dynamically joining, revoking and updating algorithm, when each of the $\lambda+1$ seed nodes SN_j generates the corresponding j-th row in the private symmetric matrix D_t, it is only required to receive the broadcast messages in the following form $H^k(s_{j'})(j' < j$ and $k = j > j')$. Consequently, for each seed generator, it is computationally impossible for the nodes SN_j to compute the seeds $s_{j'}(j' \neq j)$ and the corresponding values $H^{j'}(s_{j'})(j' \neq j)$ not belonging to themselves according to the non-reversed property of hash functions. In this way, all the updating information can be remained private among seed nodes and can not be acquired by the adversary limited with the λ -attack ability. As a conclusion, both the forward and backward security are reserved.

5.2 Performance Analysis

(1) Key Connectivity

Key connectivity is measured by the probability to establish a shared key during the dynamic topology changes. After the key-updating phase following the compromised node revocation, a pair of innocent nodes in the rest network needs to reestablish a direct or indirect key when the current pair-wise key is compromised. In our scheme, since each pair of two communicating parties has a unique pair-wise key, any sensor node's compromise cannot compromise the secure communication between innocent nodes. Furthermore, the key updating triggered by the compromise detection mechanism disables the compromise of the links composed of innocent nodes. Thus, any two innocent nodes can also establish a secret pair-wise key between them with the probability of 100%.

However, the revocation of a captured node's key ring in the probabilistic schemes[6][11] ensures a certain number of keys pre-distributed on that ring are removed from the whole network. It not only disables all connectivity of the compromised node, but also affects a few innocent nodes and a part of their key rings, thus the probability of shared key among the rest sensor nodes reduces. The symmetric bi-polynomial based random key pre-distribution scheme[8] has no obvious key revocation mechanism, but we can also compute the reestablishing probability by assuming the compromised links are in vain. Assuming that each node has available storage equivalent to C=360 keys, contacts d=40 neighbor nodes and N=26 000, Figure 3 illustrates the relationship between the probability of reestablishing a pair-wise key for innocent nodes and the number of compromised nodes in the network. It shows that our scheme has the highest probability to reestablish a pair-wise key in the rest of WSNs.

(2) Storage, Computation and Communication Overhead

Our distributed pair-wise key management scheme proposed based on Modified Blom's pair-wise key establishment mechanism[5] has the r-revocation ability which is defined as the capability of revocation of r compromised nodes at the same time without much more especially linearly increasing additional overhead in whatever respective of storage, computation and communication. The reason is that in our scheme, once a compromised node is detected by at least $\lambda + 1$ neighbor legitimate appealing nodes, the masked key information $R_{t'}(s)(s \in S = \{i, i+1, \cdots, i+\lambda \cdots\})$ transmitted among the legitimate

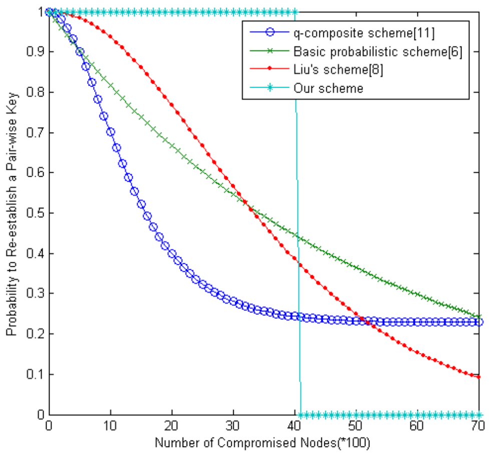


Fig. 3. The probability to reestablish a pair-wise key v.s. number of compromised nodes (q=2,m=20,S=170,k=2000,P=10^5,$s^{'}$=4,s=55,t=12,λ=40)

Table 2. Comparison between Distributed Revocation Schemes in WSNs

Schemes	Storage	Communication	Computation
Chan's scheme[4]	$O(rm\log m)$	$O(rt\log m)$	$O(t)$
Our scheme	$O(\lambda)$	$O(\lambda^2)$	$O(\lambda)$

appealing nodes is qualified for each of them to calculate all the $R_{t'_{s'}}$ $(s^{'} \in I = \{1, 2, \cdots, N_{t'}\} \setminus S)$. Therefore, if the appealing nodes discover at most r compromised nodes at one time, all their masked key information has been included in $R_{t'_{s'}}$ which was attained when the first compromised node appeared. While in the distributed revocation scheme[4] proposed by Haowen Chan, if r compromised nodes are required to be revoked simultaneously, it is necessary for each neighbor legitimate appealing node to repeat the protocol, broadcast the revocation votes along with Merkle authentication values that verify the votes and reconstruct different t-degree polynomials by secret sharing for r times independently. Besides, it will also be required for r times larger storage memory size because r distinct point pairs $\{QB(x_s), x_s\}$ and authentication values are stored in each sensor node in the pre-distribution phrase. Consequently, the storage, computation and communication overhead of Chan's scheme[4] are all much lager than ours. Table 2 illustrates the storage, computation and communication overhead of Chan's scheme[4] and ours respectively when r compromised nodes are required to be revoked at the same time. It shows that our scheme has much less overhead and is much more adaptive to the situation where collaboration of adversaries is sponsored.

6 Conclusions

A novel distributed key management scheme is proposed in this paper. It realizes the pair-wise key establishment and updating scheme based on modified Blom's symmetric key construction mechanism. Further, it not only addresses the robust continuity in the key establishment of the pair-wise key and group key, but also addresses the robust continuity in the key updating of them. In a word, this scheme has the following characteristics: (1)By the introduction of the concept of seed members, it is able to refresh the real-time key information and update the pair-wise key and group key timely when the distributed network topology changes such as dynamically joining and leaving. (2)By the novel seed member alternation mechanism combined with the technique of secret sharing, this scheme can efficiently resist to node capture attacks and revocation attacks, leaving the key connectivity of the innocent links remaining 100% after the revocation of compromised nodes. Further, the adversaries cannot take advantage of the distributed node revocation protocol to selectively revoke uncompromised nodes from the network afterwards. Finally, the resources are much more balanced among the sensor nodes network-wide, which is beneficial to prolonging the lifetime of WSNs. (3)By comparison, the continuity between the pair-wise key and group key simplifies the key management scheme[4] significantly and outperforms other similar protocols in whatever respective of storage, computation and communication overhead. Finally, how to make our schemes resilient to other more attack models and realize more efficient key management mechanism in WSNs is in the scope of our further research.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (Grant no.60773035); the Key Projects Foundation of Ministry of Education of China (Grant no.205136); The Foundation of Science and Technology Bureau of Sichuan Province, China (Grant no. 05JY029-131).

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine 8, 393–422 (2002)
2. Zhong, S., Chuang, L., Fujun, F., Fengyuan, R.: Key Management Schemes and Protocols for Wireless Sensor Networks. Journal of Software 18, 1218–1231 (2007)
3. Mi, W., Kefei, C., Yan-fei, Z., Hui, L.: A Reliable Pairwise Key-Updating Scheme for Sensor Networks. Journal of Software 18, 1232–1245 (2007)
4. Chan, H., Gligor, V.D., Perrig, A., Muralidharan, G.: On the Distribution and Revocation of Cryptographic Keys in Sensor Networks. IEEE Transanctions on Dependable and Secure Computing 2, 233–247 (2005)
5. Blom, R.: An Optimal Class of Symmetric Key Generation Systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985)

6. Eschenauer, L., Gligor, V.: A key management scheme for distributed sensor networks. In: Proc. of the 9th ACM Conf. on Computer and Communications Security, pp. 41–47. ACM Press, New York (2002)
7. Du, W., Deng, J., Han, Y.S., Varshney, P.K.: A pairwise key pre-distribution scheme for wireless sensor networks. In: Proc. of the 10th ACM Conf. on Computer and Communications Security, pp. 42–51. ACM Press, New York (2003)
8. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: Proc. of the 10th ACM Conf. on Computer and Communications Security, pp. 52–61. ACM Press, New York (2003)
9. Liu, D., Ning, P.: Location-Based pairwise key establishments for static sensor networks. In: Proc. of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, pp. 72–82. ACM Press, New York (2003)
10. Du, W., Deng, J., Han, Y.S., Chen, S., Varshney, P.K.: A key management scheme for wireless sensor networks using deployment knowledge. In: Proc. of the IEEE INFOCOM, pp. 586–597. IEEE Press, Piscataway (2004)
11. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy, pp. 197–213 (2003)
12. Camtepe, S.A., Yener, B.: Key distribution mechanisms for wireless sensor networks: A Survey. Technical Report, TR-05-07, Rensselaer Polytechnic Institute (2005)
13. Burmester, M., Desmedt, Y.: A secure and scalable group key exchange system. Information Processing Letters 94, 137–143 (2005)
14. Raazi, S.M.K., Khan, A.M., Khan, F.I., Lee, S.Y., Song, Y.J., Lee, Y.K.: MUQAMI: A locally distributed key management scheme for clustered sensor networks. IFIP International Federation for Information Processing 238, 333–348 (2007)
15. Das, A.K., Sengupta, I.: A key establishment scheme for large-scale mobile wireless sensor networks. In: Janowski, T., Mohanty, H. (eds.) ICDCIT 2007. LNCS, vol. 4882, pp. 79–88. Springer, Heidelberg (2007)

Protection Profile for Connected Interoperable DRM Framework*

Donghyun Choi, Sungkyu Cho, Dongho Won, and Seungjoo Kim**

Information Security Group, School of Information and Communication Engineering,
Sungkyunkwan University,
Suwon-si, Gyeonggi-do, 440-746, Korea
{dhchoi,skcho,dhwon,skim}@security.re.kr

Abstract. Nowadays, interoperability of DRM is an issue in market. It can be achieved by three different approaches: full-format, configuration-driven and connected interoperability. In particular, the connected interoperability refers to devices that contact an online translation service. When the consumers want to transfer content to a different device, they access the online translation server and rely upon online services. The connected interoperability does not need defining one standard and has the ability to monitor. Therefore, the connected interoperability is used more than others in market. However, in the connected interoperability, even though two distinct DRM systems want to share the content, they cannot share the content if the online translation server does not guarantee reliability. In this paper, we propose a protection profile for connected interoperable DRM framework to solve the problem. This PP can be used to establish trust between the DRM provider and the online translation server for interoperability. If that happens, the connected interoperable DRM market will be more active than before.

1 Introduction

Nowadays, we could manufacture high quality digital contents and easily trade the digital contents because of the increasing availability of computer networks and the improvement in computer technology. However, this has also caused illegal reproduction of the digital contents because such content can be easily redistributed and copied. This is threatening the digital content market. So, companies have developed the DRM (Digital Rights Management) technology. The DRM system protects the value of digital content by the license. Some examples of existing DRM systems include Apple iTunes' Fairplay[13], Windows media DRM[15], the OMA(Open Mobile Alliance)'s DRM scheme[14], etc.

* This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement)(IITA-2008-C1090-0801-0016 and IITA-2008-C1090-0801-0028).

** Corresponding author.

However, most DRM systems are neither standardized nor interoperable. From the consumers' point of view, it harms the consumers' rights of use of the content obtained legally, because they cannot render the digital content they have purchased on the device. A recent survey by INDICARE showed that consumers are willing to pay a higher price for more usage rights and device interoperability[12]. In other words, interoperability of DRM is an issue in market.

The interoperability of DRM can be achieved by three different approaches: full-format, configuration-driven and connected interoperability[6]. The full-format interoperability needs a global standard. So, all participants use the same data representation, encoding, protection scheme, etc. The configuration-driven interoperability needs a appropriate tools that can locally translate the content and the license. The tools can be downloaded from the content provider. The connected interoperability needs an online translation service. When the consumers want to transfer content to a different device, they access the online translation server and rely upon online services.

However, the full-format interoperability needs defining one standard for all applications and different business models and the configuration-driven interoperability has some problems that DRM providers do not have the ability to monitor how the user controls the content and the appropriate tools are hard to implement [5]. On the other hand, the connected interoperability does not need defining one standard and has the ability to monitor. Therefore, the connected interoperability is used more than others in market. There are lots of the connected interoperability approaches such as NEMO[6] (Networked Environment for Media Orchestration) which is used as a building block by both the Coral Consortium[17] and the Marlin Initiative[16], Import/Export in DRM[7], and OPERA[11].

However, in the connected interoperability, even though two distinct DRM systems want to share the content, they cannot share the content if the online translation server does not guarantee reliability. To address the problem, we apply the CC (Common Criteria) to the connected interoperability DRM framework, because the CC provides assurance that the process of specification, implementation and evaluation of a IT product with well-defined security evaluation criteria. Moreover, a protection profile that considers connected interoperability DRM framework is not proposed yet. In this paper, we propose a protection profile for connected interoperable DRM framework. This PP can be used to establish trust between the DRM provider and the online translation server for interoperability. If that happens, the connected interoperable DRM market will be more active than before.

The rest of the paper is organized as follows: In Section 2, we review related works. In Section 3, we propose the protection profile for connected Interoperable DRM framework. In Section 4, we present our conclusions.

2 Related Works

2.1 Common Criteria and Protection Profile

Common Criteria. The Common Criteria philosophy is to provide assurance based upon an evaluation of the IT product or system that is to be trusted[2].

The CC does so by providing a common set of requirements for the security functionality of IT products and for assurance measures applied to these IT products during a security evaluation[2]. This security evaluation includes the testing and an analysis of the IT products.

The evaluation process establishes a level of confidence that the security functionality of these IT products and the assurance measures applied to these IT products meet these requirements[2]. The evaluation results may help consumers to determine whether these IT products fulfil their security needs[2].

The CC is presented as a set of distinct but related parts as identified below.

– Part 1: The part 1 is the introduction to the CC. It defines the general concepts and principles of IT security evaluation and presents a general model of evaluation[2].
– Part 2: The part 2 establishes a set of security functional components that serve as standard templates upon which to base functional requirements for TOEs[3]. CC Part 2 catalogues the set of functional components and organizes them in families and classes[3].
– Part 3: The part3 establishes a set of security assurance components that serve as standard templates upon which to base assurance requirements for TOEs[4]. CC Part 3 catalogues the set of assurance components and organizes them into families and classes[4]. Furthermore, CC Part 3 describes seven assurance package and evaluation criteria for PPs and STs.

Protection Profile. The Protection Profile (PP) is a document that expresses an implementation independent set of IT security requirements for IT product or system. Such TOEs (Target Of Evaluation) are intended to meet common consumer needs for IT security. Therefore, consumers can construct or cite a PP to express their IT security needs without reference to any specific TOE[1].

The purpose of a PP is to state a security problem rigorously for given IT products or systems and to specify security requirements to address that problem without dictating how these requirements will be implemented[1]. Whereas an ST always describes a specific TOE, a PP is intended to describe a TOE type[2]. For this reason, a PP is implementation independent. Fig. 1 shows contents of protection profile.

2.2 Interoperable DRM

The goal of interoperable DRM is to offer consumers fully rights for the content downloaded or purchased. In other words, interoperable DRM is a technology that can, on the one hand, let rights holders receive a just remuneration for their efforts and, on the other, let end-users fully render the content on the devices in a domain. It can be achieved by three different approaches: full-format, configuration-driven and connected interoperability[6].

The full-format interoperability refers to a global standard that all parties adhere to[5]. In other words, the full-format interoperability needs defining one standard for all applications and different business models. However, developing industry standards is a long process.

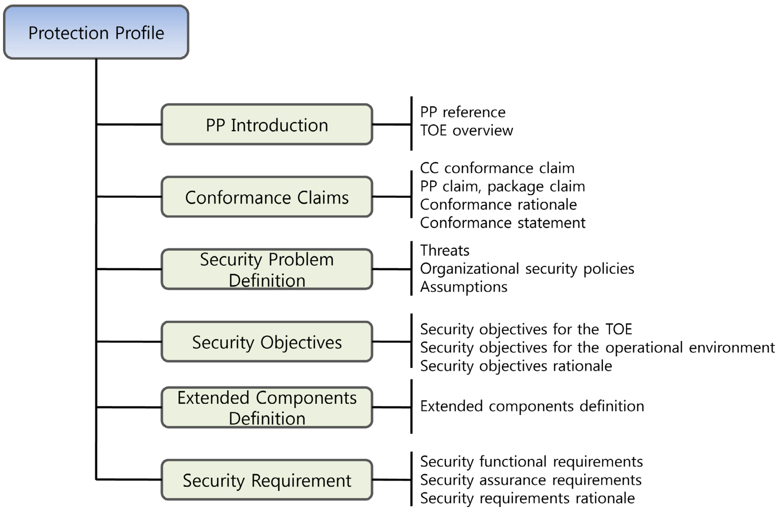

Protection Profile
PP Introduction
PP reference
TOE overview
Conformance Claims
CC conformance claim
PP claim, package claim
Conformance rationale
Conformance statement
Security Problem Definition
Threats
Organizational security policies
Assumptions
Security Objectives
Security objectives for the TOE
Security objectives for the operational environment
Security objectives rationale
Extended Components Definition
Extended components definition
Security Requirement
Security functional requirements
Security assurance requirements
Security requirements rationale

Fig. 1. Contents of Protection Profile

The configuration-driven interoperability needs a appropriate tools that can locally translate the content and the license. The tools can be downloaded from the content provider. This allows consumer systems to effectively "acquire" functionality on demand in order to accommodate new formats, protocols, and so on[6]. However, DRM providers do not have the ability to monitor how the user controls the content and the appropriate tools are hard to implement.

The connected interoperability refers to devices that contact an online translation service[5]. When the consumers want to transfer content to a different device, they access the online translation server and rely upon online services. While different parties may do things in different ways, translations or bridges exist between the ways different parties perform DRM functions, and that mutually trusted parties can perform these translations transparently, as long as device are connected at least some of the time[6]. This approach does not need defining one standard and has the ability to monitor how the user controls the content. Therefore, the connected interoperability is used more than others in market.

3 Protection Profile

In this section, we propose the protection profile for connected interoperable DRM framework.

3.1 Introduction of Protection Profile

Overview of TOE. In this paper, the TOE is the connected interoperable DRM framework. The TOE translates the neutral license and the neutral content into format of importing DRM system online. The TOE can be a framework consists of software, hardware and firmware. Fig. 2 represents the TOE.

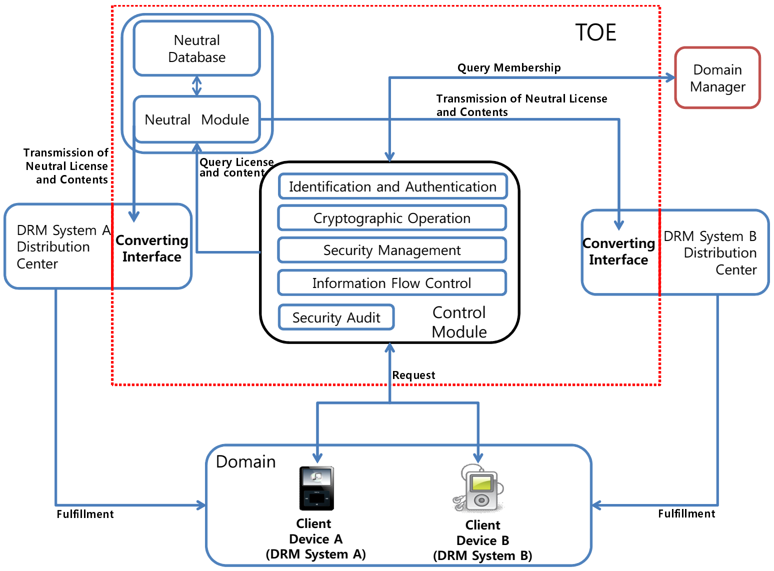

TOE
Neutral Database
Neutral Module
Query Membership
Domain Manager
Transmission of Neutral License and Contents
Transmission of Neutral License and Contents
Query License and content
Identification and Authentication
Cryptographic Operation
Security Management
Information Flow Control
Security Audit
Control Module
DRM System A Distribution Center
Converting Interface
Converting Interface
DRM System B Distribution Center
Request
Domain
Client Device A (DRM System A)
Client Device B (DRM System B)
Fulfillment
Fulfillment

Fig. 2. The TOE

The TOE comprises control module, converting interface and neutral module. A description of each module is as follows:

– Control Module (CM): CM certifies and manages the DRM systems that want to participate the connected interoperable DRM framework. This module controls the other modules and the flow of information on the TOE. When there is a query for content, CM coordinates the required services in order to fulfill the request, search, matching, update, rights exchange, and notification services.
– Neutral Module (NM): This module has the neutral formatted version of the license and the content. If neutral module receives the request from the CM, this module sends the neutral formatted version of the license and the content to importing DRM systems' converting interface.
– Converting Interface: To adopt the neutral formatted version of the license and the content, the procedure of adaptation is needed. Converting interface is located in importing DRM systems' distribution center. This module translates the neutral license and content into format of importing DRM system.

Modules in the TOE interact in a peer-to-peer fashion. Two peers interact by making service invocation requests and receiving response.

The process of TOE is as follows:

1. Device B requests a rights transfer to CM.
2. Through the domain manager, CM verifies that the device B is in fact registered in the domain.

Table 1. The Threats

Name	Threat Definition
T1.ACCESS	Unauthorized access to the TOE
T2.ATTACK_DATA	Threat agent enforces the cryptanalysis attack on the packaged content
T3.DOMAIN	An attacker may impersonate a user who is under certain domain
T4.IMPERSONATE	An unauthorized DRM system may impersonate an authorized DRM system of the TOE and thereby gain access to TOE data, keys, and operations
T5.INFORMATION_LEAK	An attacker may exploit information that is leaked from the TOE during normal usage
T6.MODIFICATION	An attacker may utilize unauthorized device and manipulating of the TOE to modify security-critical data so that the TOE can be used fraudulently
T7.REPLACEMENT	An attacker may physically or logically replace a part of the TOE
T8.TAMPERING	Threat agent may tamper TOE security functional data in order to avoid trail or cause misusage
T9.UPDATE	If importing DRM's client are not up to date versions of their corresponding software and firmware, attackers gets direct access to the digital content with vulnerabilities of certain implementations

3. If device B is a member of the domain, CM contacts the NM to request the neutral license and content.
4. The neutral license and content is delivered to the converting interface of DRM system B by the NM.
5. The device B acquires a native DRM license and content using the native mechanisms from the distribution center.
6. The device B renders the content in its native format, using its native DRM(DRM system B).

3.2 Security Problem Definition

This section defines the security problem. The security problem definition consists of three subsections, threats, organizational security policies and assumptions. The process of deriving the security problem definition falls outside the scope of the CC[2].

Table 2. The Organizational Security Policies

Name	Policy Definition
P1.AUDIT	A policy shall be implemented to ensure that the audit logs are examined regularly and frequently for accountability, and appropriate action taken over any irregularities discovered
P2.CRYPTOGRAPHY	When TOE uses a cryptographic function, approved and validated cryptographic algorithm is required
P3.SECURE_MANAGE	An authorized administrator shall manage the TOE in a secure manner

Table 3. The Assumptions

Name	Assumption Definition
A1.DOMAIN_MANAGER	Devices in a certain domain are authorized by Domain Manager
A2.ADMINISTRATOR	Administrators are non-hostile, appropriately trained and follow all administrator guidance
A3.PHYSICAL	It is assumed that the TOE is maintained in a physically secure location

Table 4. The Security Objectives for the TOE

Name	TOE Security Objectives
O1.AUDIT	The TOE will provide the capability to detect and create records of security-relevant events
O2.BRUTE_FORCE	The TOE security functions shall protect from the brute-force attack which is to find security information
O3.CRYPTOGRAPHY	The TOE shall use approved and validated cryptographic services
O4.DISCLOSURE	The protection of the data against unauthorized access within the IT system, as well as the transfer is enforced by the use of encryption procedures
O5.DATA_PROTECTION1)	Licenses and contents which is in TOE shall be encrypted and stored securely
O6.DATA_PROTECTION(2)	The TOE shall protect the stored data from the unauthorized exposure, modification, and deletion
O7.I&A	The TOE must be able to identify and authenticate users prior to a llowing access to TOE user data and TSF data
O8.INFORMATION_FLOW	The TOE controls information flows according to the specified TOE security policy
O9.INTEGRITY	The TOE shall provide appropriate integrity protection for user dat a and software
O.10.MANAGE	The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.
O11.RESIDUAL_INFORMATION	The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated or system log off
O12.SELF_PROTECTION	The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or un authorized disclosure through its own interfaces
O13.UPDATE(1)	The TOE shall be updated periodically
O14.UPDATE(2)	If TOE makes the malfunction such as system failure, TOE shall be detected, re-installed, and updated

Threats. This subsection of the security problem definition shows the threats that are to be countered by the TOE. A threat consist of a threat agent, an asset and an adverse action of that threat agent on that asset[2].

The specification of threats should include all threats detected up to now, if it is not done the TOE may provide inadequate protection. In other words, if the specification of threats is insufficiency, the assets may be exposed to an unacceptable level of risk. So we reviewed most related papers and reports ([5,6,7,8,9,10,11,12,16,17]). In the result, we derive the threats in table 1, below.

Organizational Security Policies. The organizational security policies define rules and policies of organization, supporting the TOE security, which operates the TOE. The organizational security policies for this paper are described in Table 2.

Assumptions. The assumptions are "givens" regarding secure usage of the TOE and necessary conditions in order to guarantee completeness of the TOE security, because the TOE cannot support all security functions. Furthermore, the assumptions are needed to reinforce physical and personal security requirements those are not managed by common criteria. In other words, the assumptions can be regarded as axiomatic for the TOE evaluation.

Table 5. The Security Objectives for the Operational Environment

Name	TOE Security Objectives
OE1.DOMAIN_MANAGER	The environment provides a reliable domain manager
OE2.ADMINISTRATOR	The authorized administrators shall be carefully selected and trained for proper operation of the system
OE3.PHYSICAL	The environment provides physical security commensurate with the value of the TOE and the data it contains

Table 6. The Security Objective Rationale

	Security objectives														Security objectives for the operational environment		
	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	OE1	OE2	OE3
T1			X	X	X		X										
T2		X	X	X	X	X											
T3							X										
T4							X										
T5								X		X		X					
T6						X				X							
T7									X								
T8	X					X					X	X					
T9							X						X	X			
A1															X		
A2																X	
A3																	X
P1	X																
P2			X														
P3										X						X	

These assumptions should be minimized for security. So we reviewed most related papers and reports([5,6,7,8,9,10,11,12,16,17]). In the result, we derive the minimum assumptions for this protection profile in Table 3.

Table 7. The Security Functional Requirements

Security Functional Class	Security Functional Components	
Security Audit	FAU_GEN.1	Audit data generation
	FAU_GEN.2	User identity association
	FAU_SAR.1	Audit review
	FAU_SAR.2	Restricted audit review
	FAU_STG.2	Guarantees of audit data availability
	FAU_STG.4	Prevention of audit data loss
Cryptographic Support	FCS_CKM.1	Cryptographic key generation
	FCS_CKM.2	Cryptographic key distribution
	FCS_CKM.3	Cryptographic key access
	FCS_CKM.4	Cryptographic key destruction
	FCS_COP.1	Cryptographic operation
User Data Protection	FDP_ACC.2	Complete access control
	FDP_ACF.1	Security attribute based access control
	FDP_ETC.2	Export of user data with security attributes
	FDP_IFC.2	Complete information flow control
	FDP_IFF.1	Simple security attributes
	FDP_ITC.1	Import of user data without security attributes
	FDP_ITT.1	Basic internal transfer protection
	FDP_RIP.2	Full residual information protection
	FDP_UCT.1	Basic data exchange confidentiality
Identification and Authentication	FIA_AFL.1	Authentication failure handling
	FIA_SOS.1	Verification of secrets
	FIA_UAU.1	Timing of authentication
	FIA_UAU.6	Re-authenticating
	FIA_UID.1	Timing of identification

3.3 Security Objectives

The security objectives derived from the security environment in section 3.2 and describe abstract security functions which reinforce all assumptions, threats, and organizational security policies. Each element of the security environment must be mapped to one or more security objectives. If not, it is hard to ensure that entire system includes the TOE is completely secure. The security objectives for this paper are described in Table 4 and 5.

Security Objectives Rationale. The Rationale proves that the requirements are specified completely. Generally, the rationale is described in formal table. According to the rationale, it is possible to determine that security requirements are correct, complete and both protection profile author and potential developer can verify security of the proposed TOE. Table 6 describes the rationale.

3.4 Security Requirement

Security Functional Requirements. The Security functional requirements substantiate the security objectives. Each security functional requirement must be related to one or more security objectives. These requirements are defined in CC part 2, and protection profile author just chooses and uses appropriate requirements. In addition, if the requirements defined in CC part 2 are not sufficient to demonstrate the security objectives, then, the protection profile author can refine and reinforce conditions in detail to established requirements. The security functional requirements for this paper are described in Table 7 and 8.

Table 8. The Security Functional Requirements

Security Functional Class	Security Functional Components	
User Data Protection	FDP_ACC.2	Complete access control
	FDP_ACF.1	Security attribute based access control
	FDP_ETC.2	Export of user data with security attributes
	FDP_IFC.2	Complete information flow control
	FDP_IFF.1	Simple security attributes
	FDP_ITC.1	Import of user data without security attributes
	FDP_ITT.1	Basic internal transfer protection
	FDP_RIP.2	Full residual information protection
	FDP_UCT.1	Basic data exchange confidentiality

Table 9. The Security Assurance Requirements

Security Assurance Class	Security Assurance Components	
Development	ADV_ARC.1	Security architecture description
	ADV_FSP.4	Complete functional specification
	ADV_IMP.1	Implementation of the TSF
	ADV_TDS.3	Basic modular design
Guidance Document	AGD_OPE.1	Operational user guidance
	AGD_PRE.1	Preparative procedures
Life-cycle Support	ALC_CMC.4	Production supports, acceptance procedures and automation
	ALC_CMS.4	Problem tracking CM coverage
	ALC_DEL.1	Delivery procedures
	ALC_DVS.1	Identification of security measures
	ALC_LCD.1	Developer define life-cycle model
	ALC_TAT.1	Well-defined development tools
Security Target Evaluation	ASE_CCL.1	Conformance claims
	ASE_ECD.1	Extended components definition
	ASE_INT.1	ST introduction
	ASE_OBJ.2	Security objectives
	ASE_REQ.2	Derived security requirements
	ASE_SPD.1	Security problem definition
	ASE_TSS.1	TOE summary specification
Tests	ATE_COV.2	Analysis of coverage
	ATE_DPT.3	Testing : modular design
	ATE_FUN.2	Functional testing
	ATE_IND.2	Independent testing – sample
Vulnerability assessment	AVA_VAN.4	Methodical vulnerability analysis

Security Assurance Requirements. Our protection profile adopts EAL4+(methodically designed, tested, and reviewed) level in common criteria, because it is difficult to restore spilled asset to its former condition and result of attack has heavy damage. The security assurance requirements for the TOE are described in table 9. Because the role of TOE is carried out between heterogeneous DRM systems to control entire information flow, we extend security assurance requirements to confirm structurally about entire information control and functional operation. Extended requirements are ATE_FUN.2, ATE_DPT.3, and AVA_VAN.4.

Security Requirements Rationale. Table 10 and 11 describe the rationale between security objectives and security functional requirements. Because each security objective and threat are mapped to one or more security requirements and objective, it is ensured that the TOE can prevent known threats and is kept secure in operation.

Table 10. The Security Requirements Rationale

	O 1	O 2	O 3	O 4	O 5	O 6	O 7	O 8	O 9	O 10	O 11	O 12	O 13	O 14
FAU_GEN.1	X													
FAU_GEN.2	X													
FAU_SAR.1	X													
FAU_SAR.2	X													
FAU_STG.2	X													
FAU_STG.4	X													
FCS_CKM.1			X											
FCS_CKM.2			X											
FCS_CKM.3			X											
FCS_CKM.4			X											
FCS_COP.1		X	X											
FDP_ACC.2				X		X								
FDP_ACF.1				X		X								
FDP_ETC.2					X									
FDP_IFC.2								X						
FDP_IFF.1								X						
FDP_ITC.1				X										
FDP_ITT.1				X										
FDP_RIP.2											X			
FDP_UCT.1			X	X										

Table 11. The Security Requirements Rationale

	O 1	O 2	O 3	O 4	O 5	O 6	O 7	O 8	O 9	O 10	O 11	O 12	O 13	O 14
FIA_AFL.1							X							
FIA_SOS.1							X							
FIA_UAU.1							X							
FIA_UAU.6							X							
FIA_UID.1							X							
FMT_MOF.1										X				
FMT_MSA.1										X				
FMT_MSA.2										X				
FMT_MSA.3										X				
FMT_MTD.2										X				
FMT_SMF.1										X				
FMT_SMR.1										X				
FPT_AMT.1													X	X
FPT_ITC.1					X									
FPT_ITI.1									X					
FPT_ITT.1					X									
FPT_PHP.2														
FPT_PHP.3												X		
FTP_ITC.1				x										

4 Conclusions

Nowadays, the connected interoperability is used more than others in market. However, the connected interoperable DRM market is not buoyant, because many of content providers cannot trust the online translation server. To address the problem, we proposed a protection profile for connected interoperable DRM framework. This PP can be used to establish trust between the DRM provider and the online translation server for interoperability. If that happens, the connected interoperable DRM market will be more active than before.

References

1. Lee, S., Shin, M.: Protection Profile for Software Development Site. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 499–507. Springer, Heidelberg (2005)
2. Common Criteria, Common Criteria for Information Technology Security Evaluation; part 1: Introduction and general model, Version 3.1 R1, CCMB-2006-09-001 (September 2006)
3. Common Criteria, Common Criteria for Inofrmation Technology Security Evaluation; part 2: Security functional components, Version 3.1 R2, CCMB-2007-09-002 (September 2007)
4. Common Criteria, Common Criteria for Information Technology Security Evaluation; part 3: Security assurance components, Version 3.1 R2, CCMB-2007-09-003 (September 2007)
5. Taban, G., Cardenas, A.A., Gligor, c.d.: Towards a Secure and Interoperable DRM Architecture. In: ACM Workshop On Digital Rights Management, pp. 69–78 (October 2006)
6. Koenen, R.H., Lacy, J., Mackey, M., Mitchell, S.: The Long March to Interoperable Digital Rights Management. Proceedings of the IEEE 92(6), 883–897 (2004)
7. Safavi-Naini, R., Sheppard, N.P., Uehara, T.: Import/Export in Digital Rights Management. In: Proceedings of the 4th ACM workshop on Digital rights management, pp. 99–110 (2004)
8. Schmidt, A.U., Tafreschi, O., Wolf, R.: Interoperability Challenges for DRM Systems. In: Second International Workshop on Virtual Goods, Ilmenau, Germany (May 2004)
9. Michiels, S., Verslype, K., Joosen, W., De Decker, B.: Towards a Software Architecture for DRM. In: Proceedings of the ACM Digital Rights Management workshop DRM 2005, pp. 65–74 (2005)
10. Kravitz, D.W., Messerges, T.S.: Achieving Media Portability through Local Content Translation and End-to-End Rights Management. In: Proceedings of the ACM Digital Rights Management workshop DRM 2005, pp. 27–36 (2005)
11. Wegner, S.: Prototype Description of an Open DRM Architecture, OPERA-Interoperability of Digital Rights Management Technologies. EURESCOM project report (December 2003)
12. The Informed Dialogue about Consumer Acceptability of DRM Solutions in Europe. Consumer Survey on Digital Music and DRM (May 2005)
13. iTunes FairPlay, http://www.apple.com/lu/support/itunes/authorization.html
14. Open Mobile Alliance, http://www.openmobilealliance.org/
15. Microsoft Windows Media Rights Manager, http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx
16. Intertrust's Coral and Marlin, http://www.intertrust.com/main/research/initiatives.html
17. Coral Consortium, http://www.coral-interop.org/

Author Index