Robert H. Deng
Tao Feng (Eds.)

LNCS 7863

Information Security Practice and Experience

9th International Conference, ISPEC 2013
Lanzhou, China, May 2013
Proceedings

Springer

Lecture Notes in Computer Science 7863

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

Robert H. Deng Tao Feng (Eds.)

Information Security Practice and Experience

9th International Conference, ISPEC 2013
Lanzhou, China, May 12-14, 2013
Proceedings

Springer

Volume Editors

Robert H. Deng
Singapore Management University
School of Information Systems
Singapore 178902, Singapore
E-mail: robertdeng@smu.edu.sg

Tao Feng
Lanzhou University of Technology
School of Computer and Communication
Lanzhou 730050, China
E-mail: fengt@lut.cn

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-38032-7 e-ISBN 978-3-642-38033-4
DOI 10.1007/978-3-642-38033-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013936104

CR Subject Classification (1998): E.3, D.4.6, C.2.0, K.6.5, K.4.4, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The 9th International Conference on Information Security Practice and Experience (ISPEC 2013) was held during May 12–14, 2013, Lanzhou, China.

The ISPEC conference series is an established forum that brings together researchers and practitioners to provide a confluence of new information security technologies, including their applications and their integration with IT systems in various vertical sectors. In previous years, ISPEC took place in Singapore (2005), Hangzhou, China (2006), Hong Kong, China (2007), Sydney, Australia (2008), Xi'an, China (2009), Seoul, Korea (2010), Guangzhou, China (2011) and Hangzhou, China (2012). For all the conferences in the series, as this one, the proceedings were published by Springer in the *Lecture Notes in Computer Science*.

Acceptance into the conference proceedings was very competitive. The Call for Papers attracted 71 submissions, out of which 27 were selected for inclusion in the proceedings. The accepted papers cover multiple topics in information security, from technologies to systems and applications. Each submission was anonymously reviewed by at least three reviewers.

This conference was made possible through the contributions from many individuals and organizations. We would like to thank all the authors who submitted papers. We are grateful to the Program Committee as well as all external reviewers for their time and valuable contribution to the tough and time-consuming reviewing process. We heartily thank Jianying Zhou and Hui Li for their extremely interesting and informative keynote speeches. We sincerely thank the Honorary Chairs Xiaoming Wang and Zhanting Yuan for their generous and strong support. Special thanks are due to Ying Qiu for managing the conference website and to the Organizing Committee for handling all the logistics for the conference.

We are grateful to Lanzhou University of Technology, Lanzhou, China, for organizing and hosting ISPEC 2013. Finally, we would like to thank all the participants for their contribution toward making ISPEC 2013 a success.

May 2013

Feng Bao
Robert H. Deng
Tao Feng
Xiaohong Hao

ISPEC 2013

9th Information Security Practice and Experience Conference

Lanzhou, China May 12–14, 2013

Organized by Lanzhou University of Technology (LUT), China

Supported by Institute for Infocomm Research, Singapore

Honorary Chairs

Xiaoming Wang	Lanzhou University of Technology, China
Zhanting Yuan	Lanzhou University of Technology, China

General Chairs

Feng Bao	Huawei Technologies Pte. Ltd., Singapore
Xiaohong Hao	Lanzhou University of Technology, China

Program Chairs

Robert Deng	Singapore Management University, Singapore
Tao Feng	Lanzhou University of Technology, China

Program Committee

Joonsang Baek	KUSTAR, UAE
Rohit Chadha	ENS Cachan, France
Kostas Chatzikokolakis	École Poly., France
Sherman Chow	Chinese University of Hong Kong, SAR China
Jason Crampton	Royal Holloway, UK
Xuhua Ding	Singapore Management University, Singapore
Dingyi Fang	Northwest University, China
Debin Gao	Singapore Management University, Singapore
Dieter Gollmann	Hamburg University of Technology, Germany
Dawu Gu	Shanghai Jiaotong University, China
Aiqun Hu	Southeast University, China

Xinyi Huang	Fujian Normal University, China
Lucas Hui	Hong Kong University, SAR China
Chunfu Jia	Nankai University, China
Boris Köpf	IMDEA Software Institute, Spain
Steve Kremer	INRIA Nancy, France
Heejo Lee	Korea University, Korea
Chao Li	National University of Defense Tech., China
Fenghua Li	Chinese Academy of Sciences, China
Tieyan Li	Huawei Technologies Pte. Ltd., Singapore
Weihua Li	Northwestern Polytechnical University, China
Yingjiu Li	Singapore Management University, Singapore
Zhenkai Liang	National University of Singapore, Singapore
Jianwei Liu	BeiHang University, China
Mixia Liu	Lanzhou University of Technology, China
Javier Lopez	University of Malaga, Spain
Changshe Ma	South China Normal Univ., China
Di Ma	University of Michigan-Dearborn, USA
Jianfeng Ma	Xidian University, China
Subhamoy Maitra	Indian Statistical Inst., India
Kanta Matsuura	University of Tokyo, Japan
Atsuko Miyaji	JAIST, Japan
Yi Mu	University of Wollongong, Australia
Qingqi Pei	Xidian University, China
Yong Qi	Xi'an Jiaotong University, China
Douglas Reeves	North Carolina State University, USA
Kui Ren	University at Buffalo, USA
Kouichi Sakurai	Kyushu University, Japan
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Kyushu University, Japan
Shaohua Tang	South China University of Technology, China
Guilin Wang	University of Wollongong, Australia
Huaxing Wang	Nanyang Technological University, Singapore
Jingsong Wang	Tianjin University of Technology, China
Lina Wang	Wuhan University, China
Jian Weng	Jinan University, China
Duncan Wong	City University of Hong Kong, SAR China
Hongjun Wu	Nanyang Technological University, Singapore
Yondong Wu	Institute for Infocomm Research, Singapore
Yang Xiang	Deakin University, Australia
Zheng Yan	Aalto University, Finland
Yanjiang Yang	Institute for Infocomm Research, Singapore
Sung-Ming Yen	National Central University, Taiwan
Hongbo Yu	Tsinghua University, China
Fangguo Zhang	Sun Yat-Sen University, China
Yuqing Zhang	Chinese Academy of Sciences, China
Yunlei Zhao	Fudan University, China

Dong Zheng	Shanghai Jiao Tong University, China
Jianying Zhou	Institute for Infocomm Research, Singapore
Jiliu Zhou	Sichuan University, China
Qingsheng Zhu	Chongqing University, China

Organizing Chairs

Baicheng Wang	Lanzhou University of Technology, China
Yong Zhang	Lanzhou University of Technology, China

Organizing Committee

Laicheng Cao	Lanzhou University of Technology, China
Yixin Liang	Lanzhou University of Technology, China
Mixia Liu	Lanzhou University of Technology, China
Peng Liu	Lanzhou University of Technology, China
Jianbing Xue	Lanzhou University of Technology, China
Yan Yan	Lanzhou University of Technology, China
Yong Yu	Lanzhou University of Technology, China
Fuqing Zhao	Lanzhou University of Technology, China

External Reviewers

Ahmed, Mahbub
Au, Man Ho
Cai, Shaoying
Chen, Jiageng
Chu, Cheng-Kang
Dong, Xinshu
Gao, Wei
Gong, Boru
Gorantla, Choudary
Guo, Fuchun
Haghighi, M. Sayad
Hamadou, Sardaouna
Han, Jinguang
Hu, Hong
Huang, Tao
Konidala, Divyan
Lai, Junzuo
Li, Nan
Liang, Kaitai
Liu, Ya
Liu, Zhen
Ma, Jianfeng
Martin, Keith
Nishide, Takashi
Omote, Kazumasa
Sarkar, Santanu
Shi, Jie
Smyth, Ben
Tan, Xiao
Tsang, Ww
Wan, Zhiguo
Wang, Meiqin
Wang, Mingjun
Xu, Jia
Xu, Lingling
Yasunaga, Kenji
Yoneyama, Kazuki
Yuen, Tsz Hon
Zhang, Bingsheng
Zhu, Youwen

Table of Contents

Network Security

Identity-Based Cryptography

Cryptographic Primitives

Security Protocols

System Security

Software Security DRM

Cryptanalysis and Side Channel Attacks

Enhancing False Alarm Reduction Using Pool-Based Active Learning in Network Intrusion Detection

Yuxin Meng and Lam-For Kwok

Department of Computer Science,
City University of Hong Kong, Hong Kong, China
ymeng8@student.cityu.edu.hk

Abstract. Network intrusion detection systems (NIDSs) are an important and essential defense mechanism against network attacks. However, during their detection, a large number of NIDS false alarms could be generated, which is a major challenging problem for these systems. To mitigate this issue, machine-learning based false alarm filters have been developed to refine false alarms, but it is very laborious and difficult for security experts to provide many labeled examples to train a classifier. In this paper, we therefore attempt to investigate the performance of active learning, which can make the optimal use of the given datasets, in this particular field of NIDS false alarm reduction. After analyzing the relationship between the process of false alarm reduction and the process of intrusion detection, we design a simple but efficient pool-based active learning algorithm in a false alarm filter and evaluate its performance by comparing it with several traditional supervised machine learning algorithms. The experimental results show that the designed pool-based active learner can generally achieve a better outcome than a traditional machine learning algorithm, and that the designed scheme can approximatively reduce the required number of labeled alarms by half.

Keywords: Network Security, Active Learning and Its Applications, False Alarm Reduction, Intrusion Detection.

1 Introduction

With the rapid development of computer networks, network intrusion detection systems (NIDSs) are being widely implemented in current network environments (e.g., an insurance company) to defend against various network attacks (e.g., Trojan, malware) [23]. Traditionally, these network intrusion detection systems can be generally classified into two types: *signature-based NIDS* and *anomaly-based NIDS*. For the type of signature-based NIDS [18,25], it mainly detects an attack by examining packets and comparing them to known signatures[1]. On the other hand, an anomaly-based NIDS [5,24] identifies a potential attack by comparing network events with pre-established normal profile. The normal profile is used to represent a normal behavior or network events.

Problem. A big suffering issue for current detection systems is that a large number of NIDS alarms, especially false alarms, non-critical alarms[2] may be generated during

[1] The signature (or called *rule*) is a kind of descriptions for a known attack and is generated based on expert knowledge for this attack or exploit.

[2] A *non-critical alarm* is either a false alarm or a non-critical true alarm [14].

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 1–15, 2013.

the detection process [3], which could significantly decrease the efficiency of detection and heavily increase the burden of analyzing NIDS alarms [10,12]. Thus, this problem is a key limiting factor to impede the development of these systems [3]. Take anomaly-based NIDSs for an example, some traffic accidents can easily cause such detection systems to generate a lot of NIDS alarms, most of which are false alarms [21].

To mitigate this problem, designing and constructing an appropriate false alarm filter is a promising solution (i.e., flexible to deploy). In [13], we have proposed an *adaptive false alarm filter* by applying different machine learning algorithms to filtering out NIDS false alarms. By intelligently selecting the most appropriate machine learning algorithm, in the experiments, we found that the alarm filter could achieve a good and stable filtration accuracy (i.e., above 80 percent). The adaptive selection can avoid the fluctuant performance of a single algorithm. However, for the filter, it requires an expert to label alarms periodically to guarantee its filtration accuracy. In real settings, it is very laborious and difficult for security experts to label a large number of alarms (e.g., true or false) for training a machine learning classifier. In addition, labeling good quality data is very expensive since the cost of manual annotation is very high.

To further resolve this issue, in this work, we mainly attempt to apply *active learning* to the specific field of NIDS false alarm reduction. In particular, active learning usually assumes that the learner has some control over the input space, and it is able to interactively query a user for useful information. Note that active learning has been investigated in improving the performance of detecting potential network attacks, while it has not been *extensively* studied in the aspect of constructing a false alarm filter.

Contributions. As active learning can achieve desirable performance using a small training dataset, in this paper, we therefore attempt to explore its performance in our designed false alarm filter (named *active-learning based false alarm filter*) by comparing it with some traditional machine learning algorithms. The contributions of our work can be summarized as follows:

- In this work, we analyze the real scenarios of false alarm generation in intrusion detection, illustrate the relationship between the false alarm reduction and the process of intrusion detection, and identify the practicality of our work.
- To evaluate the performance of active learning in false alarm reduction, we design a *pool-based active learning algorithm* and use this algorithm to construct an *active-learning based false alarm filter*. The designed algorithm can choose the desirable unlabeled data and query a security expert to label.
- In the experiment, we compared the performance of the active learner with several traditional machine learning classifiers, and evaluated the designed false alarm filter in a network environment. The experimental results show that the active learner can achieve a better outcome with only a small number of labeled NIDS alarms (i.e., reducing the required number of labeled alarms by half).

The remaining parts of this paper are organized as follows. In Section 2, we introduce the background of active learning and present some related work about using active learning in network intrusion detection; Section 3 analyzes the real scenarios of false alarm problem and describes the pool-based active learning algorithm in detail; Section 4 describes the experimental settings and analyzes the experimental results; finally, we conclude our work with future directions in Section 5.

2 Background and Related Work

In this section, we briefly introduce the notion of active learning and describe its applications in network intrusion detection.

2.1 Active Learning

In general, active learning is a form of supervised machine learning in which a learning algorithm has the capability of interactively querying a user for some useful information and thus to obtain the desired outputs. It usually consists of two components [2]: a classifier and a query function. The classifier can be any type of schemes such as Bayesian networks and support vector machines. For the query function, it mainly decides the next examples that should be labeled. The query function is the most significant part of active learning and is also the major difference from a traditional machine learning algorithm. By labeling the most relevant examples (or *instances*), the active learner can minimize the number of queries required. That is, active learning can achieve good performance by only using as few labeled examples as possible.

In the field of intrusion detection, Lee *et al.* [8] first proposed a framework using data mining and machine learning algorithms to identify misuse features if given good quality labeled data. However, obtaining good quality labeled examples is very expensive and it is very hard to collect so many attack examples in network intrusion detection (i.e., the known malicious instances are limited). Therefore, it is a desirable property of active learning that only a small number of labeled instances are required to achieve good performance (e.g., a better classification accuracy).

2.2 Related Work

In order to reduce false alarms in network intrusion detection, a lot of machine learning algorithms have been studied. Pietraszek [16] proposed an adaptive alert classifier system that used both of the analysts' feedback and machine learning techniques to reduce false positives. Then, Law and Kwok [7] proposed and designed a false alarm filter by means of a KNN (k-nearest-neighbor) classifier and their filter achieved a good filtration rate in the experiment. Later, Alharbt *et al.* [1] proposed a method of using continuous and discontinuous sequential patterns to detect and filter out abnormal alarms.

To train these classifiers for achieving a better filtration accuracy, a sufficient number of labeled alarms are required. But in real deployment, a large number of labeled instances are usually unavailable in network intrusion detection. To mitigate this issue, active learning has been investigated in improving the performance of an intrusion detection system. For example, Almgren and Jonsson [2] conducted some experiments to investigate whether an active learning algorithm could perform better than a traditional self-learning algorithm by using less labeled data. Their experimental results showed that the active learning algorithm could indeed outperform other traditional algorithms given the same amount of data. Then, Li and Guo [9] designed an algorithm of active learning based TCM-KNN that could effectively detect anomalies with high detection rate, low false positives by using much fewer selected data. Stokes *et al.* [22] proposed an approach of using active learning combined with rare class discovery and uncertainty

identification to train a classifier. Once trained, their system could be run as a fixed classifier with no further learning. Their experimental results showed that their system could identify more rare classes with a fewer number of labels required. Later, Görnitz *et al.* [6] rephrased anomaly-based detection as an active learning task and proposed an algorithm of ActiveSVDDs to achieve a perfect separation of normal and attack data. Seliya and Khoshgoftaar [19] then conducted a case study of combining active learning with neural networks on DARPA KDD99 datasets and their results showed positive effect of active learning on instance selection.

Note that in the above work, active learning is mainly applied to improving the capability of detecting potential network attacks whereas not much work has directly studied the effect of active learning on the particular field of false alarm reduction. In real-world applications, we find that the false alarm problem has its own characteristics compared to the process of intrusion detection. In this work, our motivation is therefore to explore the effect of a pool-based active learner on constructing a false alarm filter. The experimental results show that our designed active learner is efficient by requiring only half labeled alarms compared to several other traditional supervised classifiers.

3 Our Proposed Method

In this section, we analyze the real scenarios of false alarm generation in network intrusion detection, describe the design of pool-based active learning algorithm and present the architecture of the *active-learning based false alarm filter* in detail.

3.1 False Alarm Problem

False alarms are a big challenge for network intrusion detection systems in which a lot of false alarms could be produced during the detection. These false alarms can greatly lower the efficiency and effectiveness of detection, and make a negative impact on the analysis result. In real deployment, we identify that this issue stems primarily from the inherent limitations of detection approaches.

- *Signature-based NIDS.* The detection capability of these systems is mainly depending on the stored signatures. That is, the detection accuracy is limited to the number and content of their available signatures. But in real settings, the number of signatures is limited and these available signatures are difficult to cover all known attacks and exploits. For example, through simply modifying attack forms (i.e., modifying flag values in a packet format), a signature-based NIDS may generate a lot of false alarms. In addition, massive non-critical alarms could be generated when detecting multi-step attacks.
- *Anomaly-based NIDS.* The detection accuracy of these detection systems depends heavily on the pre-established normal profile. As described above, the normal profile is used to present a normal event. However, it is very hard to establish a good quality normal profile in most cases since network traffic is too flexible and is very hard to predict [21]. Therefore, a lot of generated alarms are false alarms in real network environment. For example, some traffic mutations can easily violate the normal profile and cause an anomaly-based NIDS to produce many false alarms.

In terms of the above analysis, we find that the false alarm generation and the process of intrusion detection have a close relationship. From the view of a machine learning classifier, the process of intrusion detection and the process of false alarm reduction can be both regarded as a learning task. That is, it is very similar to classify incoming network events or incoming NIDS alarms as positive instances or negative instances. Here an alarm is positive if it is a true alarm while an alarm is negative if it is a false alarm. Referred to the features in information retrieval [27], we identify and summarize some common characteristics for both the process of false alarm reduction and the process of intrusion detection.

- *Limited samples*. This characteristic is due to the fact that few security experts can provide a large number of attack examples or false alarm examples. In actual, the number of attack examples, especially for some novel attacks, is very small in real scenarios. This situation is the same for the false alarm reduction: massive unlabeled alarms are available while only a few labeled alarms can be obtained. Therefore, with an extremely small number of training examples, it is a very difficult task for many supervised machine learning algorithms to achieve a high accuracy.
- *Asymmetrical training samples*. Typical machine learning schemes usually assume that both positive and negative examples are distributed approximately equally. However, in the aspect of network intrusion detection, positive (or *normal*) examples are easily obtained and widely available. But the number of available negative (or *malicious*) examples is very small, and these examples can be further divided into many different sub-classes. This situation is also similar in false alarm reduction. With only a few negative examples, it is very hard for training an algorithm to achieve a high accuracy of detecting negative instances.

Although the process of intrusion detection and the process of false alarm reduction are similar, the latter has its distinctive characteristics in real deployment such as *less complexity*, *relatively more samples* and *less reduction flexibility*.

- *Less complexity*. Compared with detecting network attacks, the process of false alarm reduction is relatively less complex since the types of false alarms are usually smaller than the types of attacks. For example, there are 4 general attack types with 39 sub-classes in the DARPA KDD99 datasets [4], in which the types of attacks are far more than the types of false alarms. Due to fewer types (or *sub-classes*), a machine learning classifier is more likely to achieve a higher accuracy of detecting false alarms than network attacks.
- *Relatively more samples.* As illustrated above, the false alarm problem is less complicated than the process of detecting intrusions where the types of false alarms are relatively lower. In this case, it is feasible to identify and obtain more samples of false alarms. For example, it is applicable to retrieve more false alarm samples from the historical data in an organization. With more samples, it is easier to build an accurate classifier in identifying false alarms.
- *Less reduction flexibility*. In order to response to an attack timely, both a signature-based NIDS and an anomaly-based NIDS should examine incoming network packets on-line so that a very powerful self-learning algorithm is desirable (i.e., sometimes this algorithm is complicated due to high requirements). But for the

process of false alarm reduction, it can be conducted either on-line or *off-line* in which a relatively simple algorithm may be more efficient.

On the whole, based on the analysis of the false alarm problem, we identify that a compact active learning algorithm is feasible and can be applied to the field of reducing false alarms rather than a complicated scheme, with the purpose of reducing the computational burden (i.e., deploying on a resource-limited platform).

3.2 Pool-Based Active Learning Algorithm

As discussed above, we find that the process of false alarm reduction has its own characteristics, and that applying a relatively simple machine learning scheme may be more efficient and effective in this particular field. In this section, we therefore design a simple pool-based active learning algorithm to help reduce NIDS false alarms. Note that the term of *pool-based active learning* was used in text classification which aims to choose the best input points to gather output values from a pool of input samples [11]. In this work, a *pool* is just a collection of labeled data during the process of expert classification and its size limits the number of instances that are required to label. We use the term of *pool* since it is a key parameter for our designed algorithm.

Formally, for the active learner AL, let pl denotes a pool, f denotes a classifier, q denotes a query function, U denotes the unlabeled dataset and L denotes the labeled dataset. In this case, we can have $L = P \cup N$ where P and N denote the sets of labeled positive examples and negative examples respectively. Originally, U is the whole unlabeled database. In the area of false alarm reduction, security experts should label several alarm examples including both positive and negative examples. If let P' and N' denote the new labeled positive and negative examples respectively, then the positive set will be $P \cup P'$ and the negative set will be $N \cup N'$. The labeled dataset is obviously $L = P \cup P' \cup N \cup N'$ while the remaining unlabeled data is $U - (P' \cup N')$.

For the classifier f, in this work, we use the KNN (k-nearest neighbor) algorithm to cluster and classify incoming NIDS alarms. The KNN algorithm can classify objects based on closest training examples in the feature space. The reasons for selecting this classifier are described as below:

- The KNN algorithm is easily to implement and causes relatively lower computational burden (i.e., comparing to neural network classifiers). In addition, this algorithm can achieve a fast speed in the process of training and classification.
- In [15], we find that the KNN classifier is good at clustering and classifying incoming NIDS alarms, and in [13], we find that it could achieve a high filtration accuracy of false alarms (i.e., to be competitive with *LibSVM* and *J48*).

For the query function q, we use a method of uncertainty sampling in which the query function is desirable to identify the most uncertain unlabeled examples with regard to the classifier f. Based on this method, we should measure the uncertainty among different unlabeled examples. In this work, the KNN algorithm is selected as the classifier f so that it is applicable to measure the uncertainty by calculating the Euclidean distance. The calculation of the Euclidean distance is described as below:

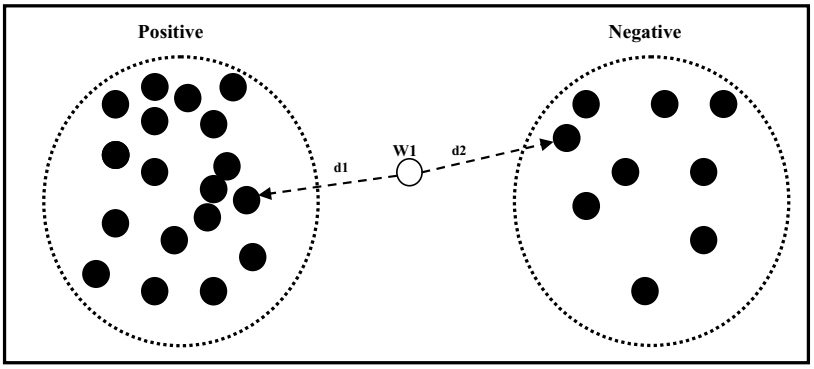


Fig. 1. An example of an unlabeled data point (white point *W1*) and its Euclidean distance (*d1*, *d2*) from both the positive cluster and the negative cluster respectively

$$[Distance\ (P1,\ P2)]^2 = \sum_{0}^{N}(P1_i - P2_i)^2 \tag{1}$$

$P1_i$ and $P2_i$ are the values of the ith attribute of points $P1$ and $P2$ respectively. Intuitively, from the view of the classifier f, the points which have the same or approximately the same Euclidean distance from both the closest positive point and the closest negative point, could be regarded as the most uncertain unlabeled instances.

To better illustrate the measurement of uncertainty, we give an example of calculating the Euclidean distance in Fig. 1. We assume that there are two clusters: positive cluster and negative cluster. Note that in real-world applications, the number of examples in the negative cluster is far less than those in the positive cluster. For the unlabeled white point *W1*, we assume that its Euclidean distance from the closest point in the positive cluster is *d1* while its Euclidean distance from the closest negative point is *d2*. There are three possible situations for *d1* and *d2*:

- $d1 = d2$, the white point *W1* has the same Euclidean distance from both the positive and the negative cluster.
- $d1 < d2$, the white point *W1* is closer to the positive cluster.
- $d1 > d2$, the white point *W1* is closer to the negative cluster.

Obviously, if we have $d1 = d2$, then the white point *W1* will be regarded as the most uncertain instance. For the other situations (e.g., $d1 \neq d2$), we should compare the absolute value of $|d1 - d2|$ among other unlabeled instances. We therefore define a metric of $E_{Distance}$ to measure the uncertainty as below:

$$E_{Distance} = |P_{Distance} - N_{Distance}| \tag{2}$$

Where $P_{Distance}$ means the Euclidean distance from the closest point in the positive cluster while $N_{Distance}$ means the Euclidean distance from the closest point in the negative cluster. Intuitively, a smaller $E_{Distance}$ means more uncertain the unlabeled

Table 1. Pseudo code: our designed pool-based active learning algorithm

Input: (1) Let U denotes the unlabeled dataset, L denotes the labeled dataset, pl denotes a pool, $|pl|$ denotes the size of the pool, $P_{Distance}$ denotes the Euclidean distance from the closest point in the positive cluster and $N_{Distance}$ denotes the Euclidean distance from the closest point in the negative cluster. (2) Let $A = \{A_1, A_2, ..., A_i, ...\}$ denotes an incoming alarm stream.

Phase1:
Query function: initiate the pool pl.

While ($U \neq \emptyset$).
{ **for** i=1 to $|U|$ do
calculate $P^i_{Distance}$, $N^i_{Distance}$ and $E^i_{Distance} = |P^i_{Distance} - N^i_{Distance}|$
end for }

for i=1 to $|pl|$ **do**
for j=1 to $|U|$ **do**
{find the smallest $E^j_{Distance}$,
query for labeling this instance j,
add j to pl and remove j from U. }
end for
end for
Output: pool pl.

Phase2:
Classifier: classify incoming alarm stream.

for all labeled instances x in L, from i=1 to $|L|$ **do**
calculate $d(x_i, A)$ and order $d(x_i, A)$ from lowest to highest.
select the K nearest instances to A.
assign A the most frequent class in L.
end for
Output: labeled alarm stream A.

instance. By only labeling these most uncertain instances (which are also the least confident instances for the classifier f), the burden for a security expert can be greatly reduced and the process of algorithm training can be further enhanced. The query function, which utilizes the metric of $E_{Distance}$, is described as follows:

1. Setting the pool size $|pl|$ for the function.
2. Calculating the Euclidean distance and $E_{Distance}$ for all the unlabeled instances in the dataset.
3. Finding the unlabeled instance(s) that has (have) the smallest $E_{Distance}$.
4. Providing a security expert with this unlabeled instance for determining the correct label, and then adding this labeled instance to pl.

In Table 1, we describe the pseudo code of the designed pool-based active learning algorithm. We initially have a labeled dataset L so that we can have two clusters of

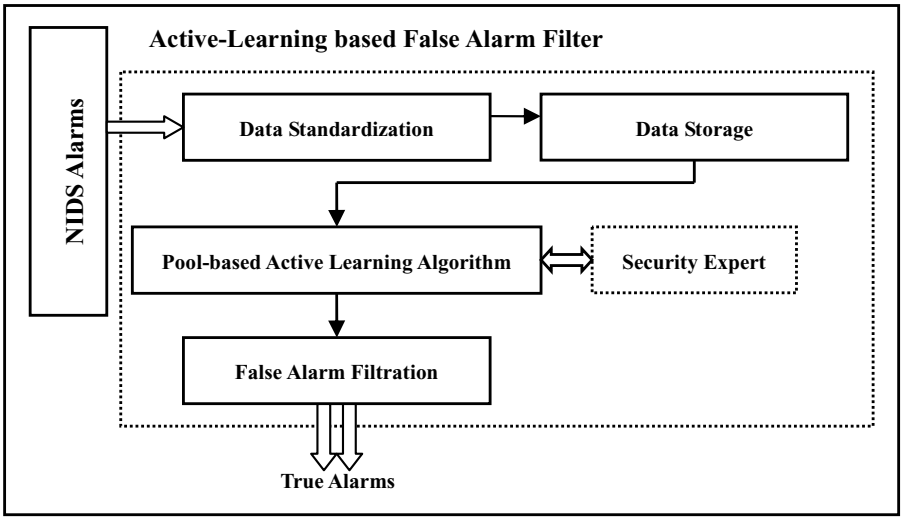


Fig. 2. The high-level architecture of the active-learning based false alarm filter

positive and negative instances. After labeling $|pl|$ uncertain instances, the KNN classifier can be trained and be used to classify the incoming alarm stream A. Note that $d(x_i, A)$ means the distance between labeled instances and the target alarm in the KNN classifier while $E_{Distance}$ means the distance calculated in the query function.

3.3 Active-Learning Based False Alarm Filter

In [13], we proposed an adaptive false alarm filter to reduce false alarms at a high and stable level by intelligently selecting the most appropriate machine learning algorithm from an algorithm pool. The most appropriate machine learning algorithm is defined as the algorithm which conducts the best single-algorithm performance. In this work, we advocate the approach of reducing alarms by constructing a false alarm filter, and design an *active-learning based false alarm filter* by implementing the pool-based active learning classifier. The high-level architecture of the alarm filter is shown in Fig. 2.

In the figure, there are four major components in the alarm filter: *Data Standardization*, *Data Storage*, *Pool-based Active Learning Algorithm*, and *False Alarm Filtration*. The component of *Data Standardization* is responsible for converting incoming NIDS alarms into a common format by representing with some pre-defined features. The conversion is very important to ensure the appropriate training for a classifier. For example, we can use a 8-feature set to represent a Snort alarm [13]. We denote the converted alarms as *standard alarms*. The component of *Data Storage* is used to store all the standard alarms. The *Pool-based Active Learning Algorithm* is responsible for interacting with security experts in labeling necessary alarms for training the active learner. Finally, the component of *False Alarm Filtration* is responsible for filtering out false alarms by means of the trained pool-based active learner and outputting true alarms.

4 Evaluation

In this section, we mainly attempt to evaluate the proposed pool-based active learner by comparing it with several traditional supervised machine learning classifiers (e.g., SVM, KNN, decision tree). In the remaining parts, we briefly describe our experimental methodology, and analyze the experimental results.

4.1 Experimental Methodology

In this evaluation, we mainly perform two experiments (named *Experiment1* and *Experiment2*) to explore the performance of the *pool-based active learning algorithm* and the *active-learning based false alarm filter* respectively.

- *Experiment1:* In this experiment, we deployed the *active-learning based false alarm filter* off-line. We provided three different pool sizes (e.g., 200, 500, 1000) for comparing the active learning algorithm with other three traditional supervised machine learning algorithms. Note that the traditional machine learners can be simulated by means of a random query [2,27].
- *Experiment2:* In this experiment, the *active-learning based false alarm filter* was deployed in a network environment and performed the process of false alarm reduction on-line. In the phase of training, we provided the alarm filter with a labeled alarm dataset (including 456 true alarms and 216 false alarms) to train the classifier. Then in the evaluation, we evaluated the *active-learning based false alarm filter* in the network environment for 3 days.

Overall, the first experiment attempts to investigate the initial performance of the *pool-based active learning algorithm* compared with traditional machine learning algorithms, whereas the second experiment aims to explore the practical performance of the *active-learning based false alarm filter* in a network environment.

4.2 Experiment1

In this experiment, we used the *Snort alarms* to evaluate the *pool-based active learning algorithm*. Snort [17,20] is an open-source signature-based NIDS, and it is very popular and widely adopted in the research of network intrusion detection. The same as [13], a Snort alarm can be represented with a 8-feature set: *description*, *classification*, *priority*, *packet type*, *source IP address*, *source port number*, *destination IP address* and *destination port number*. We used three pool sizes such as 200, 500 and 1000 in the evaluation. For the dataset, the occupancy rate of false alarms and true alarms is nearly 2:1 (note that the number of false alarms is usually bigger than that of true alarms [12] while the occupancy rate may vary with specific network settings). The specific feature extraction of Snort alarms and the classifier training can be referred to [13].

To evaluate the performance, we used two metrics: *classification accuracy* and *hit rate*. These two metrics are defined as below:

$$\textit{Classification accuracy} = \frac{\textit{the number of correctly classified alarms}}{\textit{the number of alarms}} \tag{3}$$

Table 2. The results with different pool sizes of 200, 500 and 1000 in *Experiment1*

Pool Size of 200	Accuracy (%)	Hit Rate (%)	Sample Stable Point
Pool-based Active Learning (K=5)	94.75	93.52	35
KNN (IBK-Random Query)	91.54	90.25	66
SVM (LibSVM-Random Query)	88.74	86.54	70
Decision Tree (J48-Random Query)	91.67	90.55	63
Pool Size of 500	Accuracy (%)	Hit Rate (%)	Sample Stable Point
Pool-based Active Learning (K=5)	95.68	94.12	43
KNN (IBK-Random Query)	92.86	90.88	83
SVM (LibSVM-Random Query)	89.58	87.90	95
Decision Tree (J48-Random Query)	92.87	91.05	80
Pool Size of 1000	Accuracy (%)	Hit Rate (%)	Sample Stable Point
Pool-based Active Learning (K=5)	95.80	94.53	48
KNN (IBK-Random Query)	92.76	90.54	90
SVM (LibSVM-Random Query)	90.28	88.75	100
Decision Tree (J48-Random Query)	92.67	91.85	93

$$\textit{Hit rate} = \frac{\textit{the number of false alarms classified as false alarm}}{\textit{the number of false alarms}} \tag{4}$$

The metric of *classification accuracy* is to measure the capability of identifying both true alarms and false alarms, while the metric of *hit rate* is to measure the capability of detecting false alarms. Intuitively, a better classifier is desirable to have a higher classification accuracy and a higher hit rate.

We used three specific traditional supervised learners such as *IBK-Random Query*, *LibSVM-Random Query* and *J48-Random Query* in the comparison. The experimental results are described in Table 2. We set $K = 5$ for the KNN classifier. It is visible that the *pool-based active learning algorithm* can achieve a better classification accuracy and hit rate. For the pool size of 200, the *pool-based active learning algorithm* achieves a classification accuracy of 94.75% that is greatly better than the other three machine learning algorithms (e.g., KNN 91.54%, SVM 88.74% and decision tree 91.67%). The pool-based active learner also achieves a better hit rate of 93.52% than the others (e.g., KNN 90.25%, SVM 86.54% and decision tree 90.55%).

Through increasing the pool size, that is, increasing the number of labeled examples, we find that both classification accuracy and hit rate of the *pool-based active learning algorithm* can be further improved. For instance, the classification accuracy is increased to 95.68% and 95.80% if the pool size is enlarged to 500 and 1000 respectively, and the hit rate is increased to 94.12% and 94.53% respectively. Note that the other three algorithms are improved as well, but the *pool-based active learning algorithm* can overall outperform them in the aspect of both classification accuracy and hit rate.

Referred to [2], we similarly defined a *sample stable point* as the point where its current accuracy remains greater than a certain limit of 0.1% regarding the final accuracy of the run. This point can be used to measure an algorithm when it achieves its best

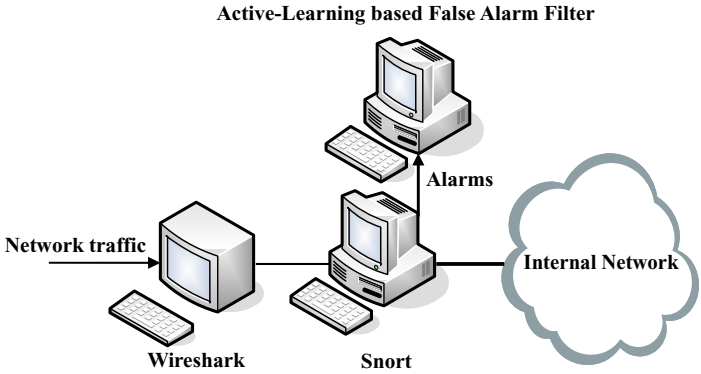


Fig. 3. The deployment of the active-learning based false alarm filter in a network environment

performance, namely *accuracy* versus *the number of labeled examples*. In Table 2, for the pool size of 200, the *pool-based active learning algorithm* can achieve this point by only using 35 labeled examples, where the number is nearly half compared with the required numbers for the other three algorithms (i.e., 66 for *IBK-Random Query*, 70 for *LibSVM-Random Query* and 63 for *J48-Random Query*). For the pool size of 500, the active learner only requires 43 labeled examples to achieve its *sample stable point*, whereas *IBK-Random Query*, *LibSVM-Random Query* and *J48-Random Query* require 83, 95 and 80 respectively. It is the same for the pool size of 1000. These experimental results show that the designed pool-based active learner can achieve better performance, compared with the other three supervised machine learning schemes, in false alarm reduction through decreasing the required number of labeled examples, and that the required number of labeled examples is nearly reduced by half.

4.3 Experiment2

In this experiment, we implemented the *active-learning based false alarm filter* in a network environment and explored the performance of the pool-based active learner on-line. The network deployment is illustrated in Fig. 3. The network environment was deployed in a CSLab and it mainly constructed by means of Snort [20], Wireshark [26] and the *active-learning based false alarm filter*. The Snort was deployed in front of the internal network to examine network packets, and the *active-learning based false alarm filter* was deployed close to the Snort. The Wireshark is responsible for recording network packets and providing statistical data.

During the experiment, all generated Snort alarms were forwarded into the *active-learning based false alarm filter*. In the phase of training, we constructed a labeled Snort alarm dataset (extracted from the historical data), which consisted of 456 true alarms and 216 false alarms, and used this dataset to train the classifier. By training with these labeled examples, the pool-based active classifier could achieve an initial classification accuracy of 90.56% and a hit rate of 90.15%. The construction of this labeled dataset

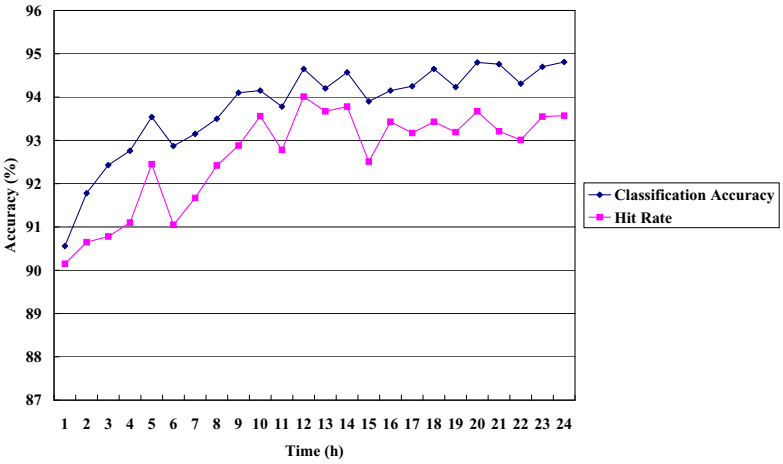


Fig. 4. The 24-hour's results of classification accuracy and hit rate in the network environment

can be referred to [13]. The pool size in this experiment is set to 200 and this experiment was conducted for 3 days. The selection of the pool size 200 is based on the following two major points:

- To use a small pool size can reduce the burden of a security expert (i.e., labeling a smaller number of alarms).
- The active learner could achieve a relatively high accuracy regarding the pool size of 200, based on the results in Table 2.

The initial 24-hour's results regarding classification accuracy and hit rate are described in Fig. 4. In this figure, it is visible that the *active-learning based false alarm filter* can achieve a high classification accuracy and a high hit rate (i.e., both rates are higher than 90%). At the end of the first day, the alarm filter could achieve a classification accuracy of nearly 95% and a hit rate of 93.5%. For the other two days, the alarm filter could achieve an average classification accuracy of 93.3% and 94.6%, and an average hit rate of 92.1% and 93.6% respectively. In these cases, these results indicate that the pool-based active learner and the *active-learning based false alarm filter* can perform well in a network environment.

Note that in the experiment, we only explore the practical performance of the active-learning based false alarm filter in a network environment, and we find that the alarm filter can achieve promising results in real deployment. To evaluate the alarm filter in a large-operational network, we leave it as an open problem in our future experiments.

5 Concluding Remarks

False alarms are a big challenge in intrusion detection. In past few years, many machine learning algorithms have been applied to reducing NIDS false alarms by constructing a

false alarm filter. However, it is a big problem that a large number of labeled alarms are required to train these supervised classifiers.

To mitigate this issue, in this paper, we attempt to apply active learning to the particular field of NIDS false alarm reduction. Specifically, we first analyzed the relationship between the process of false alarm reduction and the process of intrusion detection, and identified distinctive characteristics regarding the false alarm reduction in real deployment. We then designed a pool-based active learning algorithm and implemented it into a false alarm filter (called *active-learning based false alarm filter*). In the evaluation, we conducted two major experiments to investigate the performance of both the pool-based active learner and the designed false alarm filter. The experimental results of the first experiment showed that the active learner could achieve a better classification accuracy and hit rate than several other traditional supervised classifiers, and that the active learner could approximatively reduce the required number of labeled alarms by half. In addition, the second experiment indicated that the designed false alarm filter was encouraging and could perform well in a network environment, achieving both classification accuracy and hit rate above 90%.

In this paper, we designed a simple but efficient pool-based active learning algorithm on reducing NIDS false alarms (note that enhancing the detection capability of NIDSs is beyond the scope of this paper). There are many possible topics in our following work. Future work could include using larger and more alarm datasets to validate our obtained results, and exploring the performance of the alarm filter in a large-scale network environment. In addition, future work could also include investigating other machine learning schemes (i.e., combining semi-supervised learning with active learning) in the field of false alarm reduction and conducting a comparative study.

References

1. Alharby, A., Imai, H.: IDS False Alarm Reduction Using Continuous and Discontinuous Patterns. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 192–205. Springer, Heidelberg (2005)
2. Almgren, M., Jonsson, E.: Using Active Learning in Intrusion Detection. In: Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW), pp. 88–98 (2004)
3. Axelsson, S.: The Base-rate Fallacy and the Difficulty of Intrusion Detection. ACM Transactions on Information and System Security, 186–205 (August 2000)
4. DARPA: KDD Cup 1999 Data, `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`
5. Ghosh, A.K., Wanken, J., Charron, F.: Detecting Anomalous and Unknown Intrusions Against Programs. In: Proceedings of the 1998 Annual Computer Security Applications Conference (ACSAC), pp. 259–267 (1998)
6. Görnitz, N., Kloft, M., Rieck, K., Brefeld, U.: Active Learning for Network Intrusion Detection. In: Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence (AISec), pp. 47–54 (2009)
7. Law, K.H., Kwok, L.F.: IDS False Alarm Filtering Using KNN Classifier. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 114–121. Springer, Heidelberg (2004)
8. Lee, W., Stolfo, S.J., Mok, K.W.: A Data Mining Framework for Building Intrusion Detection Models. In: Proc. of the 1999 IEEE Symposium on Security and Privacy, pp. 120–132 (1999)

9. Li, Y., Guo, L.: An Active Learning based TCM-KNN Algorithm for Supervised Network Intrusion Detection. Computers and Security 26(7-8), 459–467 (2007)
10. Lippmann, R.P., et al.: Evaluating Intrusion Detection Systems: the 1998 DARPA off-line Intrusion Detection Evaluation. In: Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), pp. 12–26 (2000)
11. McCallum, A., Nigam, K.: Employing EM and Pool-Based Active Learning for Text Classification. In: Proceedings of the 15th International Conference on Machine Learning (ICML), pp. 350–358 (1998)
12. McHugh, J.: Testing Intrusion Detection Systems: a Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. ACM Transactions on Information System Security, 262–294 (2000)
13. Meng, Y., Kwok, L.-f.: Adaptive False Alarm Filter Using Machine Learning in Intrusion Detection. In: Wang, Y., Li, T. (eds.) Practical Applications of Intelligent Systems. AISC, vol. 124, pp. 573–584. Springer, Heidelberg (2011)
14. Meng, Y., Li, W.: Constructing Context-based Non-Critical Alarm Filter in Intrusion Detection. In: Proceedings of the 7th International Conference on Internet Monitoring and Protection (ICIMP), pp. 75–81 (2012)
15. Meng, Y., Li, W., Kwok, L.-f.: Intelligent Alarm Filter Using Knowledge-based Alert Verification in Network Intrusion Detection. In: Chen, L., Felfernig, A., Liu, J., Raś, Z.W. (eds.) ISMIS 2012. LNCS, vol. 7661, pp. 115–124. Springer, Heidelberg (2012)
16. Pietraszek, T.: Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 102–124. Springer, Heidelberg (2004)
17. Roesch, M.: Snort: Lightweight Intrusion Detection for Networks. In: Proceedings of the 13th Large Installation System Administration Conference (LISA), pp. 229–238 (1999)
18. Scarfone, K., Mell, P.: Guide to Intrusion Detection and Prevention Systems (IDPS), pp. 800–894. NIST Special Publication (2007), http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf
19. Seliya, N., Khoshgoftaar, T.M.: Active Learning with Neural Networks for Intrusion Detection. In: Proceedings of the 2010 IEEE International Conference on Information Reuse and Integration (IRI), pp. 49–54 (2010)
20. Snort. (May 2012), http://www.snort.org/
21. Sommer, R., Paxson, V.: Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp. 305–316 (2010)
22. Stokes, J.W., Platt, J.C.: ALADIN: Active Learning of Anomalies to Detect Intrusion. Technique Report. Microsoft Network Security Redmond, WA 98052 USA (2008)
23. Symantec Corp., Internet Security Threat Report, vol. 16 (July 2012), http://www.symantec.com/business/threatreport/index.jsp
24. Valdes, A., Anderson, D.: Statistical Methods for Computer Usage Anomaly Detection Using NIDES. Technical Report, SRI International (January 1995)
25. Vigna, G., Kemmerer, R.A.: NetSTAT: a Network-based Intrusion Detection Approach. In: Proceedings of the 1998 Annual Computer Security Applications Conference (ACSAC), pp. 25–34. IEEE Press, New York (1998)
26. Wireshark, (May 2012), http://www.wireshark.org
27. Zhou, Z.-H., Chen, K.-J., Dai, H.-B.: Enhancing Relevance Feedback in Image Retrieval using Unlabeled Data. ACM Transactions on Information Systems 24(2), 219–244 (2006)

Trusted Identity Management for Overlay Networks

Stefan Kraxberger, Ronald Toegl, Martin Pirker, Elisa Pintado Guijarro, and Guillermo Garcia Millan

Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria
stefan.kraxberger@gmail.com, {rtoegl,mpirker}@iaik.tugraz.at, {epintadoguijarro,ggarciamillan}@student.tugraz.at

Abstract. A critical requirement in overlay networks is to have unique, undeniable and verifiable identifiers for each node in the system. Without them, every node in such an overlay network would be able to impersonate other nodes or create an arbitrary amount of bogus nodes. Thus, a node or a group of nodes, could easily gain control over an overlay network by orchestrating such artificial nodes. Most proposed solutions are based on public key cryptography and public key infrastructures. Unfortunately, the process of issuing and distributing certificates is not solved for large scale overlay networks. In this work we provide a solution for creating unique, undeniable and verifiable identifiers for large-scale overlay networks using mechanisms provided by the Trusted Computing Group. We facilitate the use of a unique asymmetric key pair which has been created on a Trusted Platform Module and is vouched for by the manufacturer.

1 Introduction

Overlay networks commonly provide an abstraction from the underlying physical and logical network resources. Consequently, nodes in an overlay are addressed using an artificial identifier rather than a physical address. Using artificial identifiers offers advantages as they can be of arbitrary structure and length, and easily enable multi-homing. Overlay networks are often found in virtual private networks, cloud computing structures and peer-to-peer networks.

A large amount of research in overlay networks already exists. However, the issue of security has not been addressed adequately. The results of [5, 22, 28] have outlined security problems in overlay networks and provided some initial solutions especially for structured overlay networks. In [3], Bellovin has shown that additional security challenges exist in unstructured overlay networks. As one of the most fundamental problems in overlays, the issue of providing unique and verifiable node identifiers has been identified. This is due to the fact that all proposed mechanisms for designing and building secure overlay networks are based on the assumption that unique, trustworthy and unforgeable identifiers do exist. A critical weakness of this assumption is that arbitrary identifiers allow an

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 16–30, 2013.

attack called *Sybil attack* [7]. A Sybil attack is one in which an attacker subverts a overlay network by creating a large number of pseudonymous entities, using them to gain a disproportionately large influence, for instance in reputation systems. In essence, the Sybil attack refers to the capability of creating an arbitrary amount of new nodes without *physical* foundation.

The common solution for providing unique and verifiable identities is to use public key cryptography and to have a *Certificate Authority* (CA) issue certificates. The certificate authority issues a public key certificate for each node identifier by signing the node's public key. Thus, one can be assured that the node identifier is unique and through cryptographic protocols it is possible to verify the identity and authenticity of every node. To achieve a strong binding between users and their respective digital identifier, elaborate and expensive enrollment processes, such as presenting oneself with a passport to a CA's representative, are needed. Yet, such a manual process of issuing and distributing certificates is not trivial for large scale open overlay networks found for instance in content distribution systems and cloud computing. A technical solution without human interaction is therefore needed.

To ensure non-repudiable identities for platforms, not users, in overlay networks, a hardware identity token individually certified by the manufacture can be considered a robust solution. With the *Trusted Platform Module* (TPM), the *Trusted Computing Group* (TCG) has specified such a hardware identity token. The TPM has extensive support for identity and anonymity services, such as certified key pairs, and PrivacyCA and Direct Anonymous Attestation (DAA) protocols. Hundreds of millions of desktop and server PCs ship with TPMs every year [21]. However, the readily deployed TPM devices are rarely put to use and the identity mechanisms offered have not been designed to comply with the specifics of overlay networks.

In this paper we propose a solution for deriving and providing unique and verifiable identifiers for large-scale overlay networks using concepts, mechanisms and resources stemming from the Trusted Computing paradigm. We propose a modification of the PrivacyCA concept to provide unique identifiers protected by the tamper resilient TPM to denominate the peers in our network. We describe and implement two variants of a *Trusted Authentication Protocol* (TAP) based on these mechanisms, which provide unique and verifiable identities and mutual authentication for nodes in overlay networks. We report on our prototype implementation and discuss performance and security results.

2 Background and Related Work

A multitude of different overlay networks have been developed and studied over the last decade. In recent research, issues such as selfishness, reliability and security have been of high interest. This interest is related to the observation that in open overlay networks not all users or nodes are committing equal amounts of resources to the system or even actively disturb the system's execution. The problems in overlay networks in terms of security and trust can be separated into two domains: *application specific* and *system intrinsic*.

In the application domain of overlay networks, several proposals address the problems of selfishness, reliability and security. The majority of the work has been concerned with establishing trust between nodes on the basis of reputation systems [1, 10, 11]. Using reputation allows one to tackle selfish behavior of individual nodes by applying metrics on their resource-sharing habits. Other proposed mechanisms are based on Byzantine Fault Tolerance (BFT) to provide reliability and mitigate selfishness and malicious behavior [4,5] or the forming of pseudonymous groups [27]. An alternative way of dealing with selfish behaviour is for instance to use bandwidth as a shadow currency [8], thus creating a fairer behavior. However, a solution which not only focuses on *application specific* problems is needed as explained subsequently.

Castro et al. [5] identified three major *system intrinsic* requirements, which are essential to ensure security and reliability of the overlay and to provide a solid basis to build application specific solutions. These three requirements are:

1. a secure identifier assignment mechanism,
2. secure routing table maintenance, and
3. secure message forwarding.

If these requirements are not addressed, any overlay network is vulnerable to insider attacks such as route fabrication and disruption, message interception and corruption, Sybil [7] and Eclipse attacks [5] performed by Byzantine or rational nodes. This is because all the application specific solutions assume that all nodes in the system have unique identifiers which can be verified remotely from all other nodes. Thus, the first point in the requirements list is the most important one, since all application specific as well as the other system intrinsic requirements are based on secure identifiers.

Balfe et al. [2] show how Trusted Computing can be used to establish a pseudonymous authentication scheme for peers and extend this scheme to build secure channels between peers for future communications. They propose the use of the Direct Anonymous Attestation mechanism of the TPM [26], to create pseudonyms for each peer and use them for authentication. Their solution is more concerned with privacy and anonymity than with preventing Sybil and Eclipse attacks. Also, Pirker et al. [19] use the unique endorsement key of a TPM to establish a secure connection between two attested Cloud devices.

Dinger et al. [6] propose a system to tackle the issue of Sybil attacks which is based on two premises. First, they classify the overlay identifier assignment process and separate network participants from network nodes. Two challenges in the context of Sybil attacks are identified. The first is stability over time, and the second is identity differentiation. Thereafter, they propose an identity registration procedure called self-registration that makes use of the inherent distribution mechanisms of an overlay network. Using this self-registration mechanism they assume that it might be possible to effectively limit the number of identities per participant. But they also clearly state that their approach is not Sybil-proof, but that offers increased resistance, albeit with unknown effectivenss. They also concluded that still several questions remain unsolved within their solution such

as how long the time period lasts in which one can safely assume the network to be dominance-free.

Jyothi and Janakiram [12] propose a solution that enables all honest peers to protect themselves from Sybils with high probability in large structured overlay networks. In their proposed Sybil defense system they associate every peer with another non-Sybil peer known as SyMon. A given peer's SyMon is chosen dynamically such that the chances of both of them being Sybils are very low. The chosen SyMon is entrusted with the responsibility of moderating the transactions involving the given peer and hence makes it almost impossible for Sybils to compromise the system. In contrast to our system, where each node has only one unique identity and thus Sybils attacks are not possible, their system allows Sybils in the system but it is built on the probabilistic premises that no Sybil nodes are chosen as SyMon.

Martucci et al. [16] introduce self-certified pseudonyms which are computed by the nodes themselves from a cryptographic longterm identity. Based on a CA-issued membership certificate, users can create pseudonyms, each valid for one single identity domain. In the scheme, each overlay node performs a computationally expensive verification each other's pseudonymous identity, while preserving privacy.

Ryu et al. propose to use identity-based cryptography to assign node identifiers [20]. In different light-weight protocols, a connection between a public parameter derived from unique node features, such as network adresses, and a private key is established. However, the authors consider only weak authentication through IP-callbacks.

Srivatsa and Liu [23] study the probabilistic properties of attacks on the node identity and suggest to add additional security checks into a central, trusted bootstrap server that joins nodes into a network. They consider the computational costs of creating IDs and conclude that an overlay network may use weak secure IDs only as long as spoofing a large number of identifiers also requires a relatively long computational time.

Of these secure identifier assignment mechanism, none provides a cryptographically robust way, i.e. based on strong hardware authentication mechanisms to ensure unique, platform bound identities in overlay networks. In this paper, we now move on to describe how existing and already rolled-out infrastructures, based on Trusted Computing, can be modified to achieve this goal.

3 Trusted Computing Background

Trusted Computing as envisioned by the TCG promises a solution to the problem of establishing a trust relationship between otherwise unrelated platforms and enables the authentication of a computing platform's software configuration. The software configuration is measured and mapped to a set of so-called Platform Configuration Registers (PCRs). The authenticity of the values of these registers is documented by signing them with a unique private key. This protocol is called remote attestation. Attestation allows a verifying entity to query the software

configuration of a platform and correlate it with a configuration that enforces a set of required policies.

To provide the necessary hardware anchor for trust, the TCG has specified the *Trusted Platform Module* (TPM). Similar to a Smart Card, the TPM features cryptographic primitives, but is physically bound to a specific platform. A tamper-resilient casing contains hardware implementations of cryptographic operations for public-key cryptography, key generation, hashing, and random-number generation. With these components the TPM is able to enforce security policies on cryptographic keys, such as the *Endorsement Key* (EK) which provides it with a unique identity. Note that the EK RSA-key pair is not designed to be user accessible and that it cannot be used to directly sign or encrypt data. The motivation behind this is to prevent the correlation of user activities through the public key across different services.

Therefore the TPM alows the creation of anonymous keys which are referred to as *Attestation Identity Keys* (AIKs) and used for signing attestation reports. The TCG specification guarantees that these private keys never leave the protection of a standard-compliant TPM. To ensure that the signature on an attestation report is actually made by a TPM-protected AIK, the used key must be vouched for within the framework of a *Public Key Infrastructure* (PKI). AIK certificates thus guarantee that the AIK is a key controlled by *some* TPM, but they do not unveil in which specific chip. Another use for AIKs is to certify other keys and their attributes.

The TCG specifications [25] define the PrivacyCA service to protect the privacy of the users. As a trusted third party, it issues AIK certificates, which guarantee that a given AIK is owned and secured by a TPM that enforces the TCG-specified policies for AIKs. It is crucial that the issued AIK certificates do not contain any kind of link to the identity of the specific TPM. By employing more than one AIK per TPM it becomes possible to mask the identity of the users. According to the TCG's key policies, AIKs are not able to perform encryption or decryption operations. They may only sign TPM-created data structures, such as machine configuration reports or selected other keys.

According to the TCG specifications, at first, a RSA key pair is generated locally at a node, but within the protection of the node's TPM. In order to obtain an AIK certificate for it, the node and the PrivacyCA execute a cryptographic protocol. The AIK's public key and certificates that describe the EK is transmitted. The PrivacyCA checks the information. If the request conforms to the CA's policy, an AIK certificate is returned. It is however encrypted with the EK so that only the TPM indicated in the request can extract it using the so-called `TPM_ActivateIdentity` command. `TPM_ActivateIdentity`, is therefore originally intended to "activate" an AIK by decrypting a certificate that is issued by a third party. The node subsequently requests this operation on the response package. Now the AIK key can be used, together with its certificate whenever an attester requests the attestation of the node. A practical implementation of a PrivacyCA is described by Pirker et al. [18].

4 Trusted Identity Management for Overlay Networks

In the following sections we outline the TCG specified components and procedures that facilitate the realization of a trusted identity management service for overlay networks. We will discuss specific keys, certificates and protocols and their intended role. For brevity, we only consider the elements relevant to the identity creation and management. We propose two different approaches to create unique node identities. The first solution is simple and elegant, and directly incorporates the EK. Still, this approach might raise privacy concerns in some scenarios and so we also present a privacy-preserving alternative which is based on a PrivacyCA concept.

4.1 TPM Identity Mechanisms

Many of-the-shelf PCs come with a built-in TPM on the mainboard. As Trusted Computing is an opt-in technology, the TPMs are not activated per default. In order to use a TPM it must be initialized and the user must explicitly take ownership of the TPM. In this simple process, assisted by mainstream operating systems, a fresh key hierarchy is rooted. Following this, AIKs may be created.

Remember that arbitrarily many AIKs may be derived from an EK, so we cannot use an AIK to identify a platform *uniquely*. Instead, we need to create a cryptographic binding to the EK, a feature not intended by the TPM designers. To this end, we use the `TPM_ActivateIdentity` command in a slightly abusive way. This enables us to send arbitrary data, encrypted with the public key of the TPM's EK, to the TPM which then decrypts the data and returns it to the user. This function only works when the public key used to encrypt the data corresponds to the TPM's private EK and thus implicitly provides a proof of possession of the EK.

In order to use `TPM_ActivateIdentity` for our purpose, we first need to create a normal RSA AIK pair on the TPM with the `TPM_MakeIdentity` command. Since we do not actually use the AIK but rather require it only to exist later for the `TPM_ActivateIdentity` command, we omit further details about the AIK and the AIK certification process for now. Once an AIK has been created the `TPM_ActivateIdentity` command can be used an unlimited amount of times to receive data under encryption of the EK.

In theory, every TPM should be equipped with a corresponding TPM EK certificate. This certificate contains the public part of the EK pair, which serves as TPM identity. The private part is stored permanently inside the TPM and can not be retrieved once created or inserted on the production line. The certificate is signed by the TPM manufacturer and vows that the specific TPM conforms with the required specifications and the private EK is kept safe by a real, physical TPM. The appropriate X.509 certificate chain to verify the authenticity of the EK certificate needs to be publicly available.

A remaining issue is the case of a TPM shipped without a manufacturer-issued EK-certificate. The creation of such a certificate is expensive, as it requires on-chip key-pair generation, certificate issuing and injection on the time-critical chip

production line. To the best of our knowledge, at the time of writing, only a few major manufacturers such as Infineon and STMicroelectronics ship TPMs with accompanying certificates. For other selected scenarios, late construction of an EK certificate may be applicable, e.g. a limited deployment in a department-wide setup within an enterprise. Yet, this seem not feasibly for Internet-scale, open overlays, where we need to assume all nodes participating to come with a proper manufacturer provided EK certificate. All other nodes cannot take part in our system because self-created EK certificates cannot be verified without establishing another globally trusted registration authority.

4.2 TAP1: EK-Based Authentication Protocol

We now propose a first protocol (TAP1), that performs an the authentication process using the EK. The basic requirements are that both nodes have each other's EK certificate in order to verify the identity of each other. Since the TPM is involved each time the EK is used to decrypt data and each node can only have one EK we can guarantee that each node can only have one representation in the overlay network and that it is the one he has provided. An additional AIK is also used in this process due to the requirements of the `TPM_ActivateIdentity` function.

This protocol assures that the initiator possesses the private EK which matches the specified identity. In consequence the verifier can also be sure that no identity attacks such as Sybil has been performed since the used identity has only a one to one relation to the keys used for authentication. Thus, it is possible to assure that only legitimate nodes with only one identity exist in an overlay network if this solution is used for authentication and joining.

This first proposal therefore replaces the TCG's anonymity mechanism, as it unveils the public part of the EK to the overlay network. This is only acceptable if the EK is not further used in other services.

4.3 TAP2: PrivacyCA-Based Approach

We now present an alternative protocol, TAP2, that preserves the pseudonymity of users, but at the cost of higher complexity.

As an alternative to the unique and privacy sensitive EK, the TCG introduced AIKs and the associated AIK certificates. AIK certificates do not contain any information that links the certificates to the specific platform hosting the AIKs. The AIK certificates assure that the identity keys are indeed TPM hosted. Thus, using AIK certificates would enable us to provide trusted identities without compromising privacy. Unfortunately, the original intention of the PrivacyCA was to provide an unrestricted amount of AIKs for a particular endorsement certificate or TPM. Thus, it would again be possible to mount identity attacks such as the Sybil or Eclipse attack.

Therefore, in our PrivacyCA solution we must ensure that for one particular TPM or EK certificate only one AIK key-pair is issued at a time. This can easily be solved at the organizational level by modifying the PrivacyCA configuration.

Since in our case we want to protect the system from identity attacks we limit the PrivacyCA to only issue one AIK per TPM for a defined period of time. Thus, every user can obtain one AIK from the PrivacyCA at a time. The user can revoke the AIK by leaving the overlay network and can thereafter obtain a new AIK by joining the overlay network again.

Currently we achieve revocation by issuing short-lived certificates only. When the user does not leave the network properly (e.g.: the users computer crashes) she must wait until the AIK is revoked automatically at the specified expiration date. Thus, each host can only obtain and possess one identity in the overlay network and is not able to perform a Sybil attack.

In contrast to the EK solution, the joining node must first obtain a AIK certificate from the PrivacyCA. The PrivacyCA ensures that only one AIK certificate is issued per EK at a given time period. The nodes inside the network must only verify that the AIK certificate is still valid and if the issuer is the trusted PrivacyCA. The AIK and the AIK related TPM commands can then be used for identity verification inside the network. We use the Needham-Schroeder-Lowe [15] protocol as underlying authentication protocol.

This proposed use of a PrivacyCA is a slight but distinct deviation from the original intend behind this mechanism, as in theory, the issued AIK certificate could be reused for different applications than identification to one overlay network only. However, in our experience, which includes the creation and maintenance of a TCG-compliant PrivacyCA service [18], AIKs are scarcely used in practice. Especially, there currently appears to be no Internet-wide use of AIKs. Enterprises rather tend to prefer proprietary internal certification mechanisms, while private users, arguably the the main consumers of peer-to-peer technology today, hardly ever activate their TPMs at all. This approach therefore uses a system resource, the EK-provided identity, which would in many cases have been wasted, i.e. not used at all.

An advantage of this protocol is, that the EK is only known to one node, the overlay network's PrivacyCA. Given that this is a trusted third party, the same EK, and therefore TPM, can be reused in interaction with other PrivacyCAs, thus ensuring continued privacy protection.

5 Protocol Specification and Implementation

In this section we outline the details of the two *Trusted Authentication Protocols* (TAP1, TAP2) and report on our implementation. We describe the approach using the EKs, AIKs and the PrivacyCA, and thereafter the one using only EKs. We have implemented a Java software component that handles communication with the TPM hardware. To this end, we modified several libraries from the IAIK jTSS and jTT framework[1].

[1] `http://trustedjava.sourceforge.net/`

5.1 TAP1: Direct Trusted Authentication without PrivacyCA

In this authentication mode we begin with an initial exchange of keys, the AIK public key and the EK public key with the EK certificate. For each peer with its respective TPM, the authentication challenge will be encrypted. Only the target machine is able to decrypt this information. Thus, if the TPM hosts the actual AIK key that was specified then it will be able to use the EK for decryption.

The Trusted Authentication Protocol without PrivacyCA is illustrated in Figure 1 and outlined below.

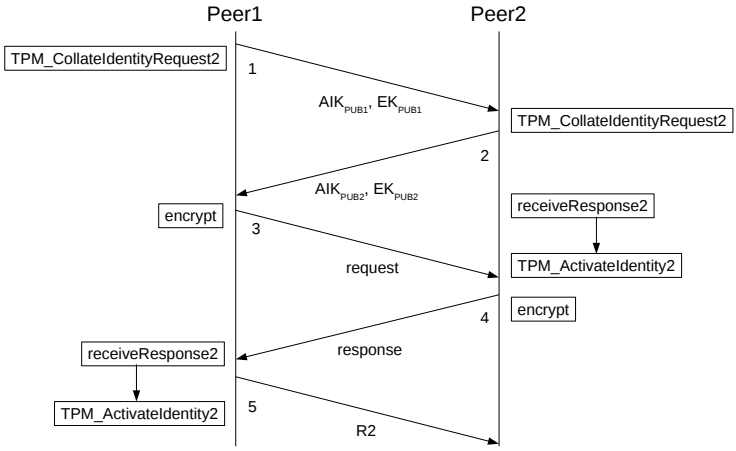


Fig. 1. TAP1: Trusted Authentication Protocol without PrivacyCA

1. Peer 1 invokes `TMP_CollateIdentityRequest2` to generate an AIK key pair called AIK_1. It sends its Endorsement Public Key (EK_{PUB1}) and the Attestation Public Identity Key (AIK_{PUB1}) together with the EK_1 certificate.
2. Peer 2 invokes `TMP_CollateIdentityRequest2` to generate a key pair called AIK_2. It sends its Endorsement Public Key (EK_{PUB2}) and the Attestation Public Identity Key (AIK_{PUB2}) together with the EK_2 certificate.
3. Peer 1 generates a random number (R_1) and a symmetric key. The symmetric key, along with a hash of AIK_{PUB2} is encrypted with EK_{PUB2}, which must be certified through the EK_2 certificate, obtained in the initial phase (2). The resulting structure is called asymmetric blob. R_1, among other data, is encrypted with the symmetric key creating the symmetric blob. The result package formed by the symmetric and asymmetric blobs assure that the request can only be decrypted by the intended recipient TPM. The hybrid symmetric/asymmetric scheme is needed to fit the required information into the TPM data structures, which just allow enough space for a short (typically 128-bit) symmetric key.

4. Peer 2 invokes `receiveResponse2` to decrypt the data calling `TMP_ActivateIdentity2`. This way we obtain R_1. Afterwards, peer 2 generates a random number R_2 and encrypts R_1 and R_2 following the same process as peer 1, using a different symmetric key to require mutual authentication on the EK_1.
5. Peer 1 receives R_1 and R_2 decrypting in the same way as peer 2 did. Then it checks that the received R_1 corresponds with the R_1 that was sent before. Finally, peer 1 sends R_2 without any encryption to peer 2, who will check if R_2 matches. This exchange of nonces R_1, R_2 assures that this is done in a fresh session and that there is no attempt of replaying as part of an attack.

Thus, authentication of peer 1 is achieved over R_1 and R_2 for joining the overlay network. This fully automatic process ensures that only nodes with a strong hardware-identity, i.e. their unique, TPM-manufacturer-certified EKs can join the network.

5.2 TAP2: Trusted Authentication with PrivacyCA

The process starts when a user, called attestant, wants to connect to a network. This attestant has an associated AIK certificate that does not contain any information that could link to the specific platform hosting the AIK. To avoid the creation of possible identity attacks, our PrivacyCA has to make sure that a certain TPM or EK certificate can only issue one single AIK certificate.

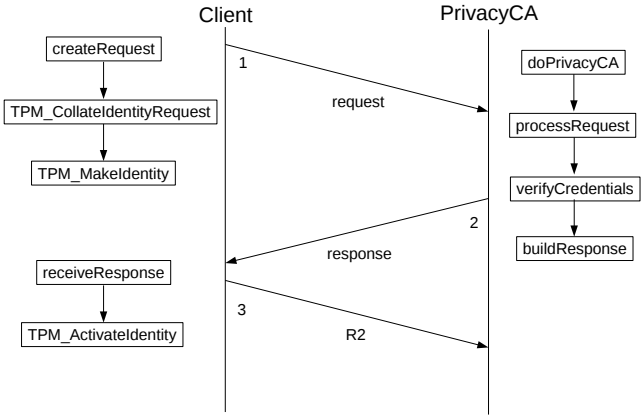


Fig. 2. TAP2: Trusted Authentication Protocol with PrivacyCA

1. The client prepares a request to send to the PrivacyCA:
   – In the first step the client calls the method `createRequest` that collates identity request data such as EK and platform certificates to initiate a new AIK creation.

- The client invokes the `TPM_MakeIdentity` TPM function to create the new AIK.
- The TPM returns a structure containing the public AIK, signed with the private AIK key. Then a request is created where the data is stored in an `identityProof` structure containing the signed blob returned by the TPM, the first random number $R1$ and the EK certificate for the TPM of the platform.
- This data is encrypted with the public key of the PrivacyCA.
- The client sends this request to the PrivacyCA.

2. The PrivacyCA prepares a response to send to the client:
   - The PrivacyCA calls the method `processRequest` where it decrypts the data, and then it validates its content and the certificates contained in the request. The PrivacyCA ensures that it has not yet issued an AIK for the given EK certificate in the current time frame.
   - On successful validation the PrivacyCA issues an AIK certificate, encrypted with a symmetric key.
   - This symmetric blob also contains the random number $R1$ that was received before and a new random number generated by the PrivacyCA (R_2).
   - The symmetric key, along with a hash of the public AIK, is encrypted with the public key of the EK of the TPM, obtained from the EK certificate of the request. This structure is called asymmetric blob.
   - The resulting package formed by the symmetric and asymmetric blobs assures that the response can only be decrypted by the intended recipient TPM.
   - After building the response, it is sent back to the client.
3. The client checks the received data from the PrivacyCA:
   - Then, the client calls the `receiveResponse` method that invokes the `TPM_ActivateIdentity` function where it decrypts the response.
   - The TPM subsequently decrypts the response package with the private part of the EK. If the referenced AIK is available on the TPM, the symmetric key is returned and it is used to decrypt the symmetric blob contained in the response. The AIK certificate is now available on the Client.
   - The client checks if the random number received from the PrivacyCA is the same number that was generated. Finally, the client sends R_2 to the PrivacyCA, in this case without encryption.

Through this protocol, an AIK and the corresponding AIK certificate are uniquely established at the client. The AIK certificate can then be used in any asymmetric authentication protocol to join an overlay network, without revealing the actual hardware identity. Due to the time-frame mechanism and the already predistributed EK certificates, no manual intervention is needed and this process can be performed automatically.

6 Evaluation and Performance

In this section we analyze the two approaches and provide time measurements on the authentication process. As the two protocols fit two different application scenarios, we outline the possible benefits and trade-offs. For strong privacy, only the PrivacyCA concept is workable. On the other hand, if simple deployment and performance are more of a concern, for instance in a closed system within a company, the better choice may be performing the authentication process utilizing the EK directly.

6.1 TAP1: Trusted EK Authentication without PrivacyCA

The following section shows the test results for the Trusted Authentication Process without PrivacyCA. We have done a series of consecutive tests to measure times between the following different parts of execution.

The initial `createRequest2` method, which creates the AIK credential using the TPM. In this phase we also obtain the Endorsement public key of the TPM. When this is done, the AIK public key and the Endorsement public key is exchanged between machines. `encrypt` generates the random number R_1, and prepares the encryption with AIK public key and the Endorsement public key of the other TPM. This way the intended TPM alone will be able to decrypt the data. `receiveResponse2` decrypts with its TPM the encryption blob received from the other machine. This encryption blob consists in a symmetric and asymmetric blob structure as seen in the authentication with the PrivacyCA.

For this process, a short series of measurements was taken on an HP dc7900CMT Core 2 duo E8400 3.0GHz with an Infineon TPM version 1.2, Firmware version 3.16 with Java 1.6 and the IAIK JCE cryptography library. The average time for `createRequest2` was 3.40[s], for `encrypt` 0.20[s] and 2.43[s] for `receiveResponse2`. The overall average total processing time was 6.02[s].

This process adds little overhead when joining an overlay network. These results indicate that this authentication type for peer-to-peer networks is faster than using authentication with PrivacyCA, as we will see in the next section.

6.2 TAP2: Trusted Authentication with PrivacyCA

The following section shows again test results for the Trusted Authentication process, this time with the PrivacyCA. We have done a series of consecutive tests to measure times between the following different parts of execution.

`createRequest` implements the creation of AIK credentials and builds an encrypted request using the PrivacyCA public key. `doPrivacyCA` decrypts the request received from the client, it extracts the first random number (R_1) and also the certificates of the client. Thereafter, it generates a new random number

(R_2), and the PrivacyCA creates a response composed of two parts. The symmetric part contains a PrivacyCA credential, R_1 and R_2; and the asymmetric part contains among other data the symmetric key, which is used to decrypt the symmetric part. The asymmetric part is encrypted with the endorsement public key of the client and using the AIK public key to assure that only the intended client will be able to decrypt the response. `receiveResponse` receives the response from the PrivayCA and decrypts it using the private key stored inside the TPM.

For this process, a short series of measurements was taken as well. The average time for `createRequest` was 3.50[s], for `doPrivacyCA` 19.91[s] and 2.40[s] for `receiveResponse`. The overall average total processing time was 25.78[s].

The PrivacyCA in our prototype requires a lot of time in preparing the encrypted data structure to allow a correct decryption in the TPM. The process of creating a fresh X.509 credential for the AIK is rather costly as a lot of entropy from a true random number generator is used. A more optimized implementation including entropy bufferingis likely to reduce this overhead.

7 Security and Privacy Analysis

First and foremost, the security of our solution depends on the integrity and tamper-resilience of the TPM. Although the TPM has been designed and manufactured with the protection of key material in mind, some attacks on the integrity of the data stored on the TPM or transmitted during the communication have been reported [9, 13, 14]. Currently only one successful attack on the the private key material within the TPM exists [24]. Still, this attack is very resource (electron microscope) and time (1 year) intensive and for most application scenarios it is still reasonable to assume that the TPM is a tamper-proof device and can be trusted for the TAP proposals.

Secondly, our solution depends on the security of the AIK and the used authentication protocol. The authentication protocol uses well-known, TCG-specified cryptographic procedures for transmitting data in a confidential and integrity ensuring manner. The authentication protocol of the EK solution itself is based on the Needham-Schroeder-Lowe [15, 17] protocol which is a well-understood and verified mutual authentication protocol commonly used. Thus, the process of joining the overlay with either the normal EK solution or the PrivacyCA solution follows best cryptographic practices.

The most important security issue, which was the intention of the whole work, the provision of unique identities which are not subject to Sybil attacks, is ensured through the reliance on the uniqueness of the EKs and the simple registration process in the overlay. Since the manufacturer of the TPM provides the EKs for each platform and signs them their authenticity can be verified by checking the signature in the EK certificate. The possession of the corresponding private key as well as the retention of the keys in the TPM is ensured by the TCG's specifications.

8 Conclusion

In this paper we show how mechanisms available by Trusted Computing can be used to enable the provision of unique trustworthy platform identities beyond existing PrivacyCA specifications. We presented two distinct variants of a Trusted Authentication Protocol which can be used in overlay networks. One solution provides simple authentication using solely the nodes' unique EKs. Although this is a simple, elegant solution it may raise privacy concerns due to the use of globally unique public keys. Thus, we provide a second solution which refines the concept of a PrivacyCA in order to provide anonymous credentials which have no relation to each other but still provide the same level of assurance in terms of authentication for overlay networks. Both approaches guarantee the creation of unique, undeniable, and verifiable identifiers in large-scale overlay networks. This effectively prevents Sybil attackers from pretending to offer more physical platform than they actually control. Our implementation demonstrates the feasibility of the approach using the readily available but hitherto nearly unused hardware identities of the TPM.

Acknowledgments. The authors thank the anonymous reviewers for their helpful comments. This work was supported by the EC, through projects FP7-ICT-SEPIA no. 257433 and FP7-SEC-SECRICOM, no. 218123.

References

1. Aberer, K., Despotovic, Z.: Managing trust in a peer-2-peer information system. In: Proc. CIKM 2001, pp. 310–317. ACM, New York (2001)
2. Balfe, S., Lakhani, A.D., Paterson, K.G.: Trusted computing: providing security for peer-to-peer networks. In: Proc. Fifth IEEE Int. Conf. Peer-to-Peer Computing, P2P 2005, pp. 117–124 (2005)
3. Bellovin, S.M.: Security aspects of Napster and Gnutella. In: Proc. USENIX (2001)
4. Bickson, D., Reinman, T., Dolev, D., Pinkas, B.: Peer-to-peer secure multi-party numerical computation facing malicious adversaries. Peer-to-Peer Networking and Applications 3(2), 129–144 (2010)
5. Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.: Secure routing for structured peer-to-peer overlay networks. SIGOPS Oper. Syst. Rev. 36(SI), 299–314 (2002)
6. Dinger, J., Hartenstein, H.: Defending the sybil attack in p2p networks: taxonomy, challenges, and a proposal for self-registration. In: Proc. ARES 2006 (2006)
7. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
8. Eger, K., Killat, U.: Bandwidth trading in bittorrent-like p2p networks for content distribution. Comput. Commun. 31(2), 201–211 (2008)
9. Grawrock, D.: Dynamics of a Trusted Platform: A Building Block Approach. Intel Press (February 2009) ISBN 978-1934053171
10. Hoffman, K., Zage, D., Nita-Rotaru, C.: A survey of attack and defense techniques for reputation systems. ACM Comput. Surv. 42(1), 1:1–1:31 (2009)

11. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. 43(2), 618–644 (2007)
12. Jyothi, B.S., Dharanipragada, J.: Symon: Defending large structured p2p systems against sybil attack. In: Proc. IEEE Ninth Int. Conf. Peer-to-Peer Computing, P2P 2009, pp. 21–30 (2009)
13. Kauer, B.: Oslo: improving the security of trusted computing. In: SS 2007: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, pp. 1–9. USENIX Association, Berkeley (2007)
14. Kursawe, K., Schellekens, D., Preneel, B.: Analyzing trusted platform communications. In: Cryptographic Advances in Secure Hardware Workshop (2005)
15. Lowe, G.: Breaking and fixing the needham-schroeder public-key protocol using fdr. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996)
16. Martucci, L.A., Kohlweiss, M., Andersson, C., Panchenko, A.: Self-certified sybil-free pseudonyms. In: Proceedings of the First ACM Conference on Wireless Network Security, pp. 154–159. ACM, Alexandria (2008)
17. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM 21(12), 993–999 (1978)
18. Pirker, M., Toegl, R., Hein, D., Danner, P.: A PrivacyCA for anonymity and trust. In: Chen, L., Mitchell, C.J., Martin, A. (eds.) Trust 2009. LNCS, vol. 5471, pp. 101–119. Springer, Heidelberg (2009)
19. Pirker, M., Winter, J., Toegl, R.: Lightweight Distributed Heterogeneous Attested Android Clouds. In: Katzenbeisser, S., Weippl, E., Camp, L.J., Volkamer, M., Reiter, M., Zhang, X. (eds.) Trust 2012. LNCS, vol. 7344, pp. 122–141. Springer, Heidelberg (2012)
20. Ryu, S., Butler, K., Traynor, P., McDaniel, P.: Leveraging identity-based cryptography for node id assignment in structured p2p systems. In: Proc. AINAW 2007, pp. 519–524 (2007)
21. Shim, R., Mainelli, T., O'Donnell, B., Chute, C., Pulskamp, F., Rau, S.: Worldwide interfaces and technologies embedded in PCs 2010-2014 forecast. Tech. rep., IDC (2010)
22. Sit, E., Morris, R.: Security considerations for peer-to-peer distributed hash tables. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 261–269. Springer, Heidelberg (2002)
23. Srivatsa, M., Liu, L.: Vulnerabilities and security threats in structured overlay networks: a quantitative analysis. In: 20th Annual Computer Security Applications Conference, pp. 252–261 (2004)
24. Tarnovsky, C.: Hacking the smartcard chip. Blackhat Conference (2010)
25. Trusted Computing Group: TCG infrastructure specifications, `https://www.trustedcomputinggroup.org/specs/IWG/`
26. Trusted Computing Group: TCG TPM specification version 1.2 revision 103 (2007)
27. Wakeman, I., Chalmers, D., Fry, M.: Reconciling privacy and security in pervasive computing: the case for pseudonymous group membership. In: Proceedings of the 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing: Held at the ACM/IFIP/USENIX 8th International Middleware Conference, pp. 7–12. ACM (2007)
28. Wallach, D.S.: A survey of peer-to-peer security issues. In: Okada, M., Babu, C. S., Scedrov, A., Tokuda, H. (eds.) ISSS 2002. LNCS, vol. 2609, pp. 42–57. Springer, Heidelberg (2003)

Situational Awareness for Improving Network Resilience Management

Mixia Liu[1], Tao Feng[2], Paul Smith[3], and David Hutchison[4]

[1] E-government Research Center, Chinese Academy of Governance, Beijing, China
liumix@163.com
[2] College of Computer and Communication, Lanzhou University of Technology, Lanzhou, China
Fengt@lut.cn
[3] AIT (Austrian Institute of Technology), Vienna, Austria
paul.smith@ait.ac.at
[4] School of Computing and Communication, Lancaster University, Lancaster, UK
d.hutchison@lancaster.ac.uk

Abstract. Computer networks, widely used by enterprises and individuals nowadays, are still vulnerable when facing traffic injection, human mistakes, malicious attacks and other failures though we spend much more time and cost on security, dependability, performability, survivability, and risk assessment to make the network provide resilient services. This is because these measures are commonly viewed as closely related but a practical means of linking them is often not achieved. Network resilience research brings together all the planning that the network can be managed at a holistic view of resilience management. This paper focuses on network resilience management from "reactive" paradigm to a "proactive" one through Situational Awareness (SA) of internal factors of network and external ones of complex, dynamic and heterogeneous network environment. After surveying the research of network resilience and resilience assessment in the network, we give a model to discuss how to construct awareness of resilience issues which includes four stages. The first step is to get the situational elements about what we are interested in. Second, to understand what happened and what is going on in the networks, pattern learning and pattern matching are exploited to identify challenge. Then, to make proactive resilience management, we need to predict challenges and look for potential ones at this stage. At the fourth stage, resilience management can help take actions of remediation and recovery according to the policy of defender and attacker. After that, the two players' behaviors of defender and attacker are modeled in the same model by using Extended Generalized Stochastic Game Nets (EGSGN) which combines Game theory into Stochastic Petri Nets. Finally, we give a case study to show how to use EGSGN to depict the network resilience situation in the same model.

Keywords: resilience, situation awareness, resilience situation, Petri net.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 31–43, 2013.

1 Introduction

Resilience is a semantically overloaded term in the sense that it means somewhat different things in different fields, such as ecological resilience, economics and business resilience, industrial and organizational resilience, psychological resilience, socio-ecological resilience and networks resilience [1]. Resilience in the network, which is defined as the ability of the networks to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation, must be viewed as an essential design and operational characteristic of future networks in general, and the Global Internet in particular [2]. In case of network failures, restoration and protection switching mechanism can reroute traffic from broken paths to backup ones provided that the networks are still physically connected. Therefore, the majority of the research focuses on the availability of the network in terms of disconnection probability and efficient calculation of network's resilience after nodes or links fail. However, the network is still vulnerable when facing traffic injection, human mistakes, malicious attacks and other failures though we spend much more time and cost on security, dependability, performability, survivability, and risk assessment to make the networks provide resilient services. This is because these measures are commonly viewed as closely related but a practical means of linking them is often not achieved. In [2], we presented strategies, principles and disciplines for network resilience. [3] continued providing a dynamic adaptation architecture for realizing these strategies as a holistic view of resilience problem. Based on the previous research, in this paper, we use Situation Awareness (SA) to construct awareness of network resilience issues to make network resilience management move from a "reactive" paradigm to a "proactive" one. SA exists to inform administrators or operators at all levels of the status of the network in order for them to recognize, understand, decide and act on any incident or event which is used here for improving the network resilience management and assessing it easily.

In light of the foregoing, this paper is organized as follows. Section 2 reviews the related work of network resilience and resilience assessment. Section 3 discusses how to make network resilience management based on SA from four stages. A case study is illustrated in section 4, and then the last part is about the conclusion and future work.

2 Related Work

2.1 Network Resilience

Networks resilience NR(p) was at first as probabilistic measure of potential disconnections in a network [4]. Joseph [5] provides a method for resilient and reliable end-to-end connectivity in heterogeneous networks. Cholda [6] made the first extensive survey of ideas related to differentiation of communication services with respect to resilience, which needs to provide services with different resilience characteristics such as the availability, continuity and duration of downtimes in the

same network. Menth [7] presents a framework for resilience analysis of packet-switched communication networks about ingress-egress unavailability and link congestion due to failures, changes of user behavior and changed interdomain routing. Rerouting in networks is considered as resilience for networks failure [8,9].

Resilience is also defined as a mechanism to assure service robustness, by ensuring that resources are re-established in case of failures [10]. This re-establishment is possible due to protection (actions before failure) and/or restoration schemes (action after failure). Resilience is a function of availability and recovery, which are determined based on the protection scheme supported.

A systematized ResiliNets strategy is presented in [2] which describes a real-time control loop to allow dynamic adaptation of networks in response to challenges, and a nonreal time control loop that aims to improve the design of the network, including the real-time loop operation, reflecting on past operational experience. And then, a systematic approach to building resilient networked systems is presented in [3]. A framework for the design and evaluation of network resilience management was presented in [11].

From what I mentioned above, network resilience was measured passively by lost traffic, disabled nodes, and network state transition which cannot meet the requirements of recovery for resilience management from "reactive" paradigm to a "proactive" one. Resilience is not only the performance to measure the system, but it provides a systemic method to make the system maintain service after failure happening by anticipation, adaptation, detection and recovery. Therefore, we looked network resilience management as a function of SA in a complex, dynamic and heterogeneous network environment.

2.2 Resilience Evaluation

Quality of resilience is decided by recovery switching from different ways, such as layers in which recovery operates, recovery resources setup method, recovery resources sharing level, scope of recovery, and domains crossed by recovery [12]. Multi Protocol Label Switching (MPLS) is as an object of assessing resilience in Resilience-Differentiated Quality of Service (RD-QoS) framework [13] and Quality of Resilience (QoR) [12]. QoR combines QoS metrics (e.g. packet loss, delay) with resilience metrics (e.g. steady-state availability, mean downtime). The Resilience Evaluation Framework (REF) [10] is suitable for general protocol resilience evaluations and defines a more realistic evaluation framework than QoR. REF can be employed to assess the resilience of any given protocol as long as the protection model (1:1 or 1+1) is taken into consideration. These resilience assessment methods aim at protocol resilience and focus on recovery techniques.

Besides assessing protocol's resilience as mentioned above, Bursztein [14] presented a logical-based framework to evaluate the resilience of computer networks in the face of incidents, i.e. attacks from malicious intruders as well as random faults. It is futile to evaluate resilience by modeling incidents only. It also needs to model response, typically by administrators. However, this assessment method has no figure description of the actions of the two plays in the network, cannot give the network resilience situation, and does not include probabilistic transitions.

Sterbenz et al [15] evaluated network resilience based on a two-dimensional state space: the horizontal axis representing the operational state of the network and the vertical axis describing the service delivered in which node and link failures are simulated as challenges. It is clear that systematic, automatic, self-adaptive, self-learning and decision making resilience networks are needed [16,17] to provide resilient service to the applications. However, there is no previous resilience assessment which can look this problem from a holistic view. QoR and REF focus on availability and recovery based on protection model, and Sterbenz's resilience framework [15] quantified network resilience as a measure of service degradation in the presence of challenges that are perturbations to the operational state of the network.

The challenges like malicious attack may destroy or damage critical components in the network services, or disable network infrastructure which have been becoming more and more seriously. It is important to predict these challenges before they happen and impact the networks. So we use SA to get resilience management automatically. Endsley's definition of SA [18], which includes the perception of the elements of the environment, the comprehension of their meaning and the projection of their status in order to enable decision superiority, can meet the dynamical resilience requirements.

3 Network Resilience Management Based on SA

To estimate resilience of networks, many factors will be considered in the process of perception not only including network internal data such as topology, traffic, resource, security, fault and vulnerabilities in the network, but the external factors from the environment of the network.

From all the frameworks or models of SA [19, 20], it is clear that SA can be used for realizing resilience management dynamically. Therefore, we use SA for improving the network resilience management. And then we analyze the factors that affect the resilience and define potential situation that we are looking for in the first stage. Such situation can be divided into attacks and failures or interruptions. Once we have defined a set of situations that we are interested in and the set of data sources available we will store the necessary data and the processes required by each source to translate and cleanse the data. According to the network resilience situation management model, challenges affecting resilience will be analyzed and anticipated. After identifying the challenges by fusing the information we obtained, it is necessary to define the state transition and recovery. The most important here is to estimate the resilient state based on the state transition model.

3.1 Model of Network Resilience Management

To improve resilience management, SA was used here for fusing data from the internal and external factors of the networks to get awareness of resilience issues, which is revised from [21] as shown in Fig. 1. The result of detection will be stored in

DISco (Distributed Store for challenges and their outcome) [3], according to which the networks resilience manager can make decision using remediation and recovery by policy of "Us" and policy of "Them". Policy of "Us" is from our work in [22] which can be changed according to the adversary's policy. Policy of "Them" comes from the awareness issues of resilience. When we take measures by remediation, the state of networks described by S_{00}, S_{ij} to S_{nn} as shown in Fig. 2 will change from one to another. S_{00} denotes that the system is in an initial state of "normal" for which acceptable service is delivered during normal operations.

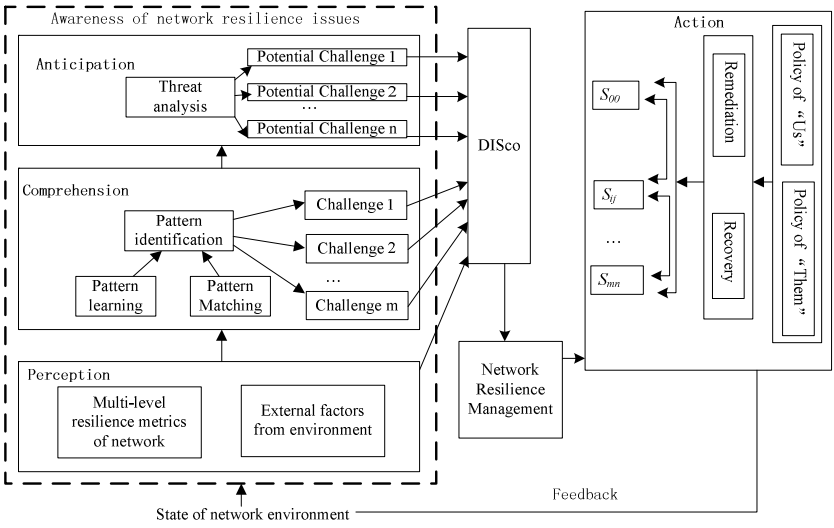


Fig. 1. Situational Awareness (SA) model for resilience management

In the process of perception of the model, it will involve perceiving the status, attributes and dynamics of task-related elements in the surrounding environment. At this stage, the data are merely perceived. It is well known that the future network will be more complex, diverse, and more heterogeneous including fiber, mobile and wireless networks etc. Therefore, to improve resilience management of the networks, it is necessary for the administrators to consider more factors in the first level including internal sensors of the networks, and also external sensors of the environment. This is because wireless signals travel through the atmosphere, which are more susceptible to different types of interference than standard wired network, such as radio frequency interference, electrical interference, weather conditions. The internal elements consist of fault, security, topology, traffic, vulnerabilities, and resource. Fault collects data of mis-configuration or not following correct policy which will lead to accidents [2]. Security collects alerts from different security tools such as IDS, firewall, malware detector, etc, which will result in malicious attack [23]. Changes of topology will be detected in the topology module. The root causes of

higher layer problems may be located in a different protocol layer or on a different network host, which is frequently not visible to the higher layer. Fault localization technology will be used here [24]. The onset of either a DDoS attack or a flash crowd could be initially detected by measuring traffic at the destination networks [25]. In the resource module, the missions or importance of the computers within the networks will be considered. In the vulnerabilities module, weak points in the network will be detected. In this stage, we can get lots of original data of different challenges which are incomplete information. How to understand these challenges and their impact to the network is the second stage's content.

3.2 Challenges Awareness of Resilience Issues

In the process of comprehension, the decision maker will form a holistic picture of the environment, comprehending the significance of objects and events. One way in accomplishing this is to build a graph from the evidence. In the resilient networks, to analyze challenges, we need to know the levels of our service, the states of our networks, state transition, and triggered condition. There are two steps to analyze the challenges: (1) Information extraction: we need to find, filter and categorize relevant data and extract facts and entities to understand what is happening of the networks; (2) Model generation and learning algorithms: it is necessary to generate predefined model and train model to recognize patterns of activity. To identify challenges, data fusion can be used here to get evidences from other sensors. In the wireless network, the environment elements can also give evidence of challenges which will result in good remediation. In this stage, we also construct challenges scenario if there is causal relationship among them. From this causality consequence, we can predict potential challenges in the next step before they lead to serious impact to the network. So it is necessary to define a set of resilience situation classification. Once we have defined a set of situations that we are interested in we next can store them in our DISco which can be matched with the challenge analysis.

In the stage of anticipation, we can anticipate the potential challenges which will have a possibility to lead to the transition of the network state by combing the knowledge from perception and comprehension. Attacker's action can be described in this situation which is not enough in resilient network. Therefore, game theory [20] can be exploited here which can model both the actions each player decides to take and the resulting consequences, which may not be controlled by the players. According to the model of comprehension, an analyst may wish to search for information that is missing from a particular model or use the model to project or anticipate where and what new data might exist.

3.3 Resilience Estimate and Analysis According to SA and Remediation

In the stage of action, the resilience management will perform resource allocation and provide feedback information to the previous stages, and the state of the network will change from one to another because of the behaviors of the attacker and the defender which comes from the result of game in the anticipation stage. Moitra [26] argued that

system states of measuring survivability {S} may be {normal, under attack, compromised, recovered, and nonfunctional}. Evaluating networks resilience in this way effectively quantifies it as a measure of service degradation in the presence of challenges to the operational state of the networks. Network can be viewed (at any layer) as consisting of two orthogonal dimensions as illustrated in Fig.2: one is the operational state of network, which consists of its physical infrastructure and their protocols; the second dimension is the service being provided by the network and its requirements [2]. Sterbenz et al [27] quantified network resilience as a measure of service degradation in the presence of challenges that are perturbations to the operational state of the network. They focus on impact of link failure to the service provided in which the critical nodes or links are shut down for the duration of the challenge period to simulate an attacker with knowledge of the network structure. To describe the different states of the network, serial numbers are used in the set of nine (3×3) regions. If there is a challenge or a serial challenges happening, it will make the state change from the best state S_{00} to the worse state S_{ij} or else. Each region may contain multiple states if the service demands such a fine granularity. If necessary, the states can be added from S_{ij} to S_{nn}. In the limiting case, each region represents just one state. S_{00} denotes that the system is in an initial state of "normal" for which acceptable service is delivered during normal operations. Network resilience can be evaluated in terms of the various network state transitions under the presence of challenges. When the operation is affected by challenged, which in turn may result in degradation of the service. Therefore, S_{22} denotes the state "non-functional" which means challenges make the operation severely degraded leading to unacceptable service provided in the system.

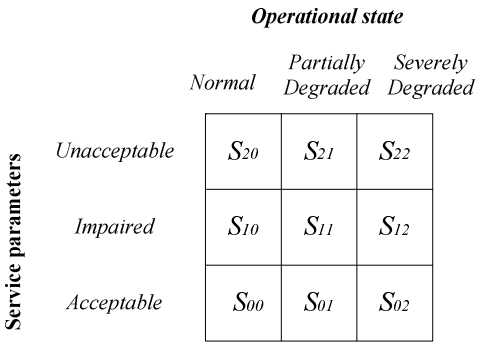


Fig. 2. Resilience state space [2]

3.4 Extended Generalized Stochastic Game Net for Resilience State Transition

Lin [28] presented Stochastic Game Nets (SGN) by extending Stochastic Petri nets with game theory and applied the SGN method to investigate the network attacks, compute the Nash equilibrium to each player. Zakrzewska [29] extended Petri Net to allow real-time cyber conflicts to be modeled. However, they all have not mentioned

how to get the action of the attackers. We add observable place and observable arc in Lin's SGN to describe inflection from security tools' alerts to attack action, add immediate transition to represent a choice between two alternatives which have exactly the same input places (firing conditions), and the probabilities assigned to them will reflect the relative probabilities of the two choices, and use it to describe and recognize resilience situation. An Extended Generalized Stochastic Game Net (EGSGN) is represented as the ten-tuple vector (P, S, O, T, π , F, R, λ , U, M_0), where

(1) P= {$Player_1$, $Player_2$, …, $Player_n$}is the finite nonempty set of players. In the network, $player_1$ denotes attacker, and $player_2$ denotes defender in the network;

(2) S= {S_1, S_2, …, S_n} is a finite set of state of different players;

(3) O = {O_1, O_2, …, O_n} is the finite set of alerts from security tools;

(4) T = $T_1 \cup T_2 \cup \cdots \cup T_k$ is a finite set of transitions, where T_1 is the set of transitions with respect to $Player_1$, T_2 is the set of transitions of $Player_2$, T_k is the transition set of $Player_k$;

(5) π :T→[0, 1] is a rate of transition happening which represents the probability of choosing a particular transition,

(6) $F = ((P \times T) \cup (O \times T)) \cup (T \times P)$ is the set of arcs between transitions and places;

(7) R is a reward function for the player taking each transition;

(8) $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_w\}$ is a set of transition firing rates in the transition set, where w is the number of transitions;

(9) U is the utility function of players;

(10) M_0 is the initial marking.

To represent resilience of networks, the Nash equilibrium can correspond to the optimal strategy of each play. First, it needs to construct a reachability tree with respect to the EGSGN, and then find out the Nash equilibrium. A situation of resilience includes a lot of steps of attacks and defenses. Resilience situation recognition is to find the transition sequence of attacker and defender in reachability tree of the EGSGN model.

4 Case Study

To provide clients with a reliable and responsive service, redundant Web service is used in Amazon, Google and MSN [14]. The two HTTP nodes HTTP[1]:1 and HTTP[1]:2 serve the Web pages index.php[1]:1 and index[1]:2 respectively. The index.php [1]:1 file depends of the FTP[1] service for being updated. The HTTP[1]:1 service depends on index.php[1]:1, as the service will fail if the file index.php to be served does not exist. We use Stochastic Petri net to establish the resilience model as shown in Fig. 3. There are two types of transition: generalized white rectangles represent timed transitions, and black rectangles represent instantaneous transitions. If FTP[1] failed because of attack or other reasons, Failure1 will be activated and take token form FTP[1], and then HTTP[1]:1 cannot provide service. If SSH[1] failed, Failure2 will be fired and take token form SSH[1], so the copy transition will not be fired and lead to the failure of HTTP[1]:2.

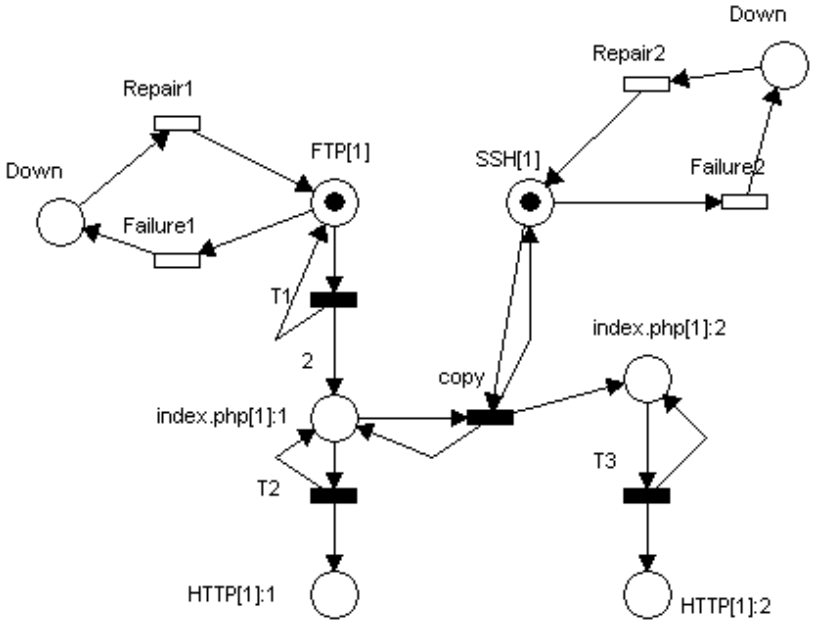


Fig. 3. Petri model of resilient Web service

According to [14], there are attacker rules t_1 and defender rules t_2 which can be extended as follows:

t_1= {Defunct, DefunctProp, Comp1, Comp2, CServProp1, CervProp2, CFileProp1, CFileProp2}. Defunct is a typical accidental fault: any file or service that is subject to faults (e.g., bugs) and is available can become unavailable. DefunctProp states that a vertex may crash when all vertices it depends on crashed, taking into account equivalent vertices. Comp1 models the case where the attack is completely successful, and the target vertex is compromised. Comp2 models a typical case where, e.g., the attack is by code injection, but the attack fails and instead the target service crashes. The remaining rules represent incident propagation. CServProp1 states how locally vulnerable services depending on compromised files. CServProp2 is the case where the service crashed instead. CFileProp1 states that non-encrypted files depending on compromised vertices may get compromised. The CFileProp2 rule here would have a larger delay, and represents compromission of encrypted (Crypt) files with a weak key (VulnLocal).

t_2= {Patch, Deny, Allow, ORest, OReco, PReco, PRest1, PRest2}. Patch is that the defender may update services. Deny is to configure network devices so as to make a service unreachable. Allow is to configure network devices so as to make a service reachable. ORest states that defender can always repair the damage. OReco is an optimistic vertex recovery rule. PReco is a pessimistic recovery rule where a compromised vertex is recovered thanks to an available, uncompromised equivalent vertex. PRest1 states that the vertex is also made available. PRest2 states that vertex is unfortunately recovered from another compromised vertex.

We construct the simulation EGSGN model for the resilient Web service as shown in Fig. 4. Each transition in the model is considered to be a Poisson distribution with a rate of for a unit of time. The time until the transition fires has an exponential probability distribution. At the first step only the state vulnerability has token, so we have the S_0= {1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 }. With the happening of transitions, tokens will be removed from the input state of the transition and be added to the output state of the transition which will lead to the change of marking from S_0 to S_1, S_2, ….,S_n. To simplify the reachability tree, we use the defense policy to replace the steps of the defender because when the defender takes measures to the attacker's action, different policies will lead to different results. PIPE (Platform Independent Petri Net Editor 2) [30] is used to model the interactive process of attacker and defender based on policy as Fig. 5. T_5 and T_6 in Fig.5 represent immediate transitions which are alternative policy of Patch FTP or Deny FTP in defending. An inhibitor arc from a place to a transition indicates that the transition cannot fire if there is a token in the place; it can fire when there is no token in the place if the places connected to its input arcs do contain tokens. Inhibitor arcs in Fig. 5 represent if the response speed is faster than the attacker's, the actions of the attack T_3 and T_4 cannot happen.

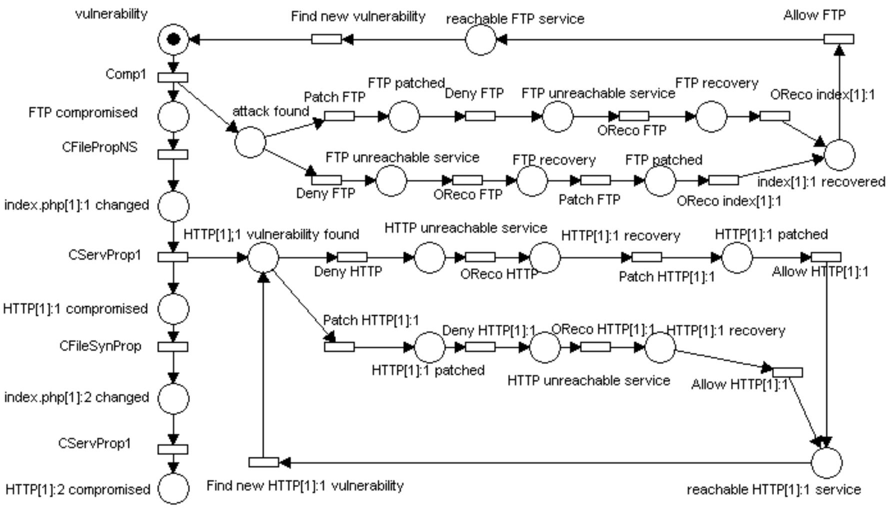


Fig. 4. Attacker and defender's action model based on EGSGN

Fig. 6 shows the reachability tree about that when FTP is compromised the defender takes two different policies to defend no considering the inhibitor arcs. Using algorithm to recognize resilience situation, we got the situation as follows: {T_0, T_6, T_1, T_2, T_3, T_4, T_8}, {T_0, T_6, T_8, T_1, T_2, T_3, T_4}, {T_0, T_6, T_1, T_8, T_2, T_3, T_4}, {T_0, T_6, T_1, T_2, T_8, T_3, T_4}, {T_0, T_6, T_1, T_2, T_3,T_8, T_4}, { T_0, T_5, T_1, T_2, T_3, T_4, T_8}, { T_0, T_5, T_8, T_1, T_2, T_3, T_4}, { T_0, T_5, T_1, T_8, T_2, T_3, T_4}, { T_0, T_5, T_1, T_2, T_8, T_3, T_4}, { T_0, T_5, T_1, T_2, T_3, T_8, T_4}. From these resilience situations, we can find where we should take measure and optimize policy which will improve the resilience of the network.

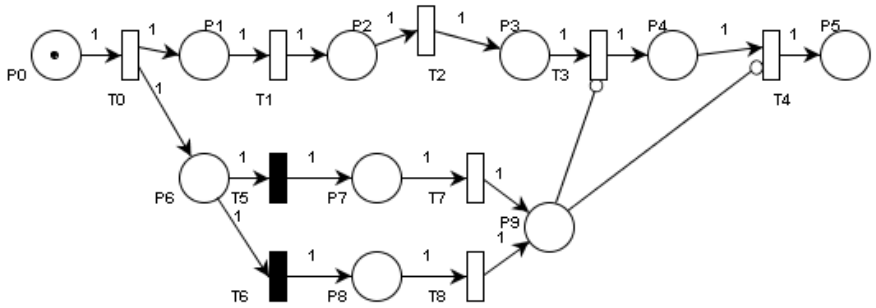


Fig. 5. EGSGN model based on policy

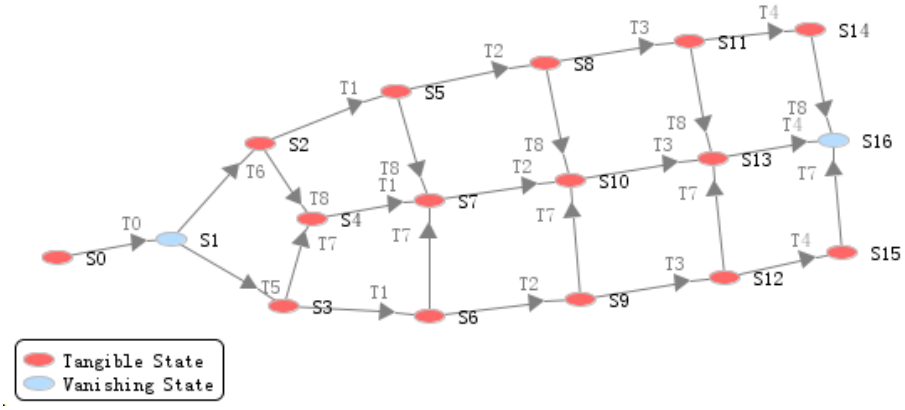


Fig. 6. Reachability tree of the Petri model

5 Conclusion and Future Work

Resilience is a dynamic and systematical property of networks which provides and maintains acceptable service level after failure happening. SA is intended to improve the network resilience management in this paper. After surveying of resilient system and resilience assessment in the network, we have addressed resilience management model through SA, which is a part of our dynamic adaptation architecture realizing resilience control loop. Four stages can obtain the resilience situation. Stage 1 is challenge data collecting from internal factors in the network including fault, topology, security, traffic, vulnerabilities, and resource and external factors from environment. The data is analyzed in stage 2 for challenge identification, and in the stage 3 we can predict the threat by game theory. To decide which action will be done, policies of attacker and defender are considered in the stage 4. We also give a case study of resilience analysis, from which we can see that the state of network will change from one to another because of the actions of the attacker and the defender.

As future work, we will research formalization methods for network resilience situation awareness and resilience situation classification which can be useful for judging whether the networks are resilient. Furthermore, optimizing the strategies of the attacker and defender to help the administrator to find the best way against challenges and improving the resilience of the network is our next step.

Acknowledgements. This research was supported by the European Union Research Framework Programme 7 via the ResumeNet project with contract number FP7-224619 and National Natural Science Foundation of China 60972078.

References

1. Madni, A.M., Jackson, S.: Towards a conceptual framework for resilience engineering. IEEE Systems Journal 3(2) (2009)
2. Sterbenz, J.P.G., Hutchison, D., Cetinkaya, E.K., et al.: Resilience and Survivability in Communication Network: Strategies, Principles, and Survey and Disciplines. Computer Networks 54, 1245–1265 (2010)
3. Smith, P., Scholler, M., Fessi, A., et al.: Network Resilience: A Systematic Approach. Submitted to IEEE Communication (December 2010)
4. Najjar, W., Gaudiot, J.: Network resilience: A measure of fault tolerance. IEEE Trans. Comput. 39(2), 174–181 (1990)
5. Joseph, D., Franks, J.K., Freeman, C.N., et al.: Reliable and Resilient End-to-End Connectivity for Heterogeneous Networks. US 2011/0038256 A1 (2011)
6. Cholda, P., Mykkltveit, A., et al.: A survey of resilience differentiation frameworks in communication network. IEEE Communications Surveys & Tutorials 9(4) (2007)
7. Menth, M., Duelli, M., Martin, R., Milbrandt, J.: Resilience analysis of packet-witched communication networks. IEEE/ACM Transactions on Networking (2009)
8. Keralapura, R., Moerschell, A., Chuah, C.N., et al.: A Case for Using Service Availability to Characterize IP Backbone Topologies. Journal of Communications and Networks 8(2) (2006)
9. Haider, A., Harris, R.: Recovery Techniques in Next Generation Networks. IEEE Communications Surveys & Tutorials 9(3) (2007)
10. Sousa, B., Pentikousis, K., Curado, M.: REF: Resilience Evaluation Framework. In: 2010 International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) (2010)
11. Schaeffer-Filho, A., Smith, P., Mauthe, A., Hutchison, D., Yu, Y., Fry, M.: A Framework for the Design and Evaluation of Network Resilience Management. In: 13th IEEE/IFIP Network Operations and Management Symposium (2012)
12. Cholda, P., Tapolcai, J., Cinkler, T., et al.: Quality of Resilience as a Network Reliability Characterization Tool. IEEE Network (2009)
13. Autenrieth, A.: Differentiated Resilience in IP-Based Multilayer Transport Networks. Ph.D. dissertation. Technische University Munchen, Munchen (2003)
14. Bursztein, E., Goubault-Larrecq, J.: A Logical Framework for Evaluating Network Resilience against Faults and Attacks. In: Cervesato, I. (ed.) ASIAN 2007. LNCS, vol. 4846, pp. 212–227. Springer, Heidelberg (2007)

15. Sterbenz, J.P.G., Cetinkaya, E.K., Hameed, M.A., et al.: Evaluation of, Network Resilience, Survivability and Disruption Tolerance: Analysis, Topology Generation, Simulation and Experimentation. Springer Telecommunication Systems Journal (2011)
16. Dove, R.: Patterns of Self-Organizing Agile Security for Resilient Network Situational Awareness and Sensemaking. In: 8th International Conference on Information Technology: New Generations (ITNG) (2011)
17. Mayron, L.M., Bahr, G.S., et al.: A Hybrid Cognitive- Neurophysiological Approach to Resilient Cyber Security. In: The 2010 Militay Communicatons Conference – Cyber Security and Network Management (2010)
18. Endsley, M.R.: Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors Journal 37(1), 32–64 (1995)
19. Bass, T.: Intrusion systems and multisensor data fusion: Creating cyberspace situational awareness. Communications of the ACM 43(4), 99–105 (2000)
20. Chen, G., Shen, D., et al.: Game Theoretic Approach to Threat Prediction and Situation Awareness. Journal of Advances in Information Fusion 2(1) (2007)
21. Liu, M., Hutchison, D.: Towards Resilient Networks Using Situation Awareness. In: The 12th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (2011)
22. Schaeffer-Filho, A., Smith, P., Mauthe, A.: Policy-driven Network Simulation: a Resilience Case Study. In: SAC 2011, March 21-25 (2011)
23. Zhai, Y., Ning, P., Iyer, P., et al.: Reasoning About Complementary Intrusion Evidence. In: Proceedings of 20th Annual Computer Security Applications Conference (2004)
24. Steinder, M., Sethi, A.S.: Probabilistic Fault Localization in Communication Systems Using Belief Networks. IEEE/ACM Transactions on Networking 12(5) (2004)
25. Michael, F., Mathias, F., Paul, S., David, H.: Challenge Identification for Network Resilience. In: 2010 6th EURO-NF Conference on Next Generation Internet (NGI) (2010)
26. Moitra, S.D., Konda, S.L.: The Survivability of Network Systems: An Empirical Analysis. CMU/SEI-2000-TR-021 (2000)
27. Sterbenz, J.P.G., Cetinkaya, E.K., Hameed, M.A., et al.: Evaluation of Network Resilience, Survivability and Disruption Tolerance: Analysis, Topology Generation, Simulation and Experimentation. Springer Telecommunication Systems Journal (2011)
28. Lin, C., Wang, Y., Wang, Y.: A Stochastic Game Nets Based Approach for Network Security Analysis. In: Proc. of the 29th International Conference on Application and Theory of Petri Nets and other Models of Concurrency, Concurrency Methods: Issues and Applications 2008 Workshop (2008) (invited paper)
29. Zakrzewska, A.N., Ferragut, E.M.: Modeling Cyber Conflicts Using an Extended Petri Net Formalism. In: 2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS) (2011)
30. Imperial College DoC MSc Group And MSc Individual Project, `http://pipe2.sourceforge.net/`

Optimal Defense Strategies for DDoS Defender Using Bayesian Game Model

Yuling Liu[1,2], Dengguo Feng[1], Yifeng Lian[1,2], Kai Chen[2], and Yingjun Zhang[1,2]

[1] Laboratory of Trusted Computing and Information Assurance, Institute of Software, Chinese Academy of Science, Beijing 100190, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
{ylliu,feng,lianyf,chenk,yjzhang}@is.iscas.ac.cn

Abstract. In a typical DDoS attack and defense scenario, both the attacker and the defender will take actions to maximize their utilities. However, each player does not know his opponent's investment and cannot adopt the optimal strategies. We formalize a Bayesian game model to handle these uncertainties and specify two problems usually faced by the defender when choosing defense measures. A nonlinear programming method is proposed to handle policies' permutation in order to maximize the defender's utility. Followed the Nash equilibrium, security administrators can take optimal strategies. Finally, the practicality and effectiveness of the model and method are illustrated by an example.

Keywords: DDoS defense, Bayesian game, Nonlinear Programming.

1 Introduction

Distributed denial-of-service (DDoS) attacks have become a widely-spread threat to network security, especially in the industry of high real time requirement, e.g. online game, VOD (video-on-demand), video conferencing. When DDoS attacks happened, a critical resource usually network bandwidths were exhausted. More seriously, through disrupted the normal service, DDoS attacks lead to corporations lost revenue, public reputation embarrassment and law penalty [1, 2]. Arora et al. have illustrated the impact of recent DDoS attacks [3]. Based on the type of victim, DDoS attacks can be divided into the following five categories: application attacks, host attacks, resource attack, network attacks and infrastructure attacks [7]. In this paper, we mainly study network attacks which consumed the victim's bandwidth with flooding DDoS attack packages. This is also the principal form of DDoS attacks.

Due to DDoS attacks' enormous destructive, many defense mechanisms such as statistical filtering [4, 5], traceback [6] have been proposed. Mirkovic et al. [7] and Douligeris et al. [8] summarized various protective mechanisms and analyzed their classification. Their work could guide security administrators to choose and deploy suitable mechanisms to defeat DDoS attacks. However, defense strategies selection is a systematic, economic related work. Security administrators should compare and

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 44–59, 2013.

evaluation every mechanism's performance, coverage and other factors. There were already many models and methods for assessing DDoS defense strategies from technical level which can be used as a reference for security administrators [4, 5, 6, 12, 13, 14, 15].

However, security budget constraints are also very important and the chosen strategy or strategy profile should maximize the protective effect. Many methods have been proposed to select policies with economic aspects. Bulter [2] presented a cost-benefit approach SAEM to compare and select the suitable security designs. Based on this work, Dewri et al.[1] formalized a multi-objective optimization on attack tree model to solve the problem how to select security hardening measures within the security budget. Böhme [9] discussed the problem of security investment planning from a high-level. He described the mapping relations between security cost, security level and security benefit and provided a model to measure security investment. At some level the above works could enlighten security administrators on planning DDoS protection investment and choosing applicable defense policies, but they have some drawbacks. First of all, existing works assessed the effectiveness of defense mechanisms or security investment from the perspective of the defender and lacked the attacker side's considerations. Actually, in order to choose protection measures well, we should comprehensive consideration of both the cost and benefits of the attacker and the defender. Because when expenses surpass gains, a rational attacker will not carry out an attack. Secondly, the works above did not discussed how to deal with residual risks. Cyber-insurance has been introduced into the field of information security and provided a way to transfer the residual risk for security administrators in practice [10, 11]. So, we treat cyber-insurance as an important choice when defeat DDoS attacks in this paper. Last but not least, the problem that how to select DDoS defense strategies with limit information security investment budget has not been resolved. How to schedule the budget to maximize the defender's rewards is remain a challenge. Security administrators on the one hand can only deals with similar problems from personal experiences, on the other hand cannot report to their superiors the effect of investment quantitatively.

In view of the above problems, we propose a Bayesian game model to characterize the relationship between DDoS attacker and defender, and quantitative evaluate of rational attacker and defender's utility by calculating the equilibrium under different investments. When computing the utilities of defender, cyber-insurance is considered and a nonlinear programming method is used to allocate security investment budgets.

In summary, this paper makes the following contributions:

-it formulizes a Bayesian game model for DDoS attack and defense (BGM-DAD), and the Nash equilibrium results of this game can guide defense mechanisms selection;

-it proposes a nonlinear programming method, which is useful when scheduling investment budgets in order to maximize defense mechanisms' performance;

-it specifies two problems usually faced by security administrators when selecting defense policies and modeled these problems into BGM-DAD;

-and it demonstrates experimental results under different offensive and defensive scenarios.

The rest of paper is structured as fellows. In section 2, we discuss the related work of ours. Section 3 formalizes the Bayesian game model for DDoS attack and defense and we present calculation methods of the attacker and the defender's utility functions in section 4. Two problems usually faced by security administrators are described in Section 5. Section 6 shows our experiments results and the final section concludes our paper.

2 Related Work

Because the number of DDoS defense mechanisms is multitudinous, researchers have proposed various methods to compare and evaluate the performance of strategies, with the aim of providing a quantitatively method to guide security administrators choosing appropriate policies. Li et al. [5] formulated DDoS attacks on statistical-based filtering, and evaluated the effectiveness of static filter and adaptive filter against static attackers and dynamic attackers respectively. Kuznetsov et al.[6] first put forward four evaluation metrics: number of packets required for reconstruction, computation overhead, robustness and deployment overhead and cost, and then used these metrics to analyze and assess primary traceback approaches quantitatively. Direct measurements of flow-level information have been regarded as a major element in above methods, but indirect measurements was neglected. Due to this, a method using multiple data sources was proposed, which could improve the measurement accuracy [12]. Mirkovic et al. [13] presented a universal evaluation method for DDoS, and developed a benchmarks suite which can depict the basic elements of assessment scenarios: the attack, the legitimate traffic and the network topology. Besides attack flows elimination, prevention and resistance were also effective solutions to defeat DDoS/DoS attacks. A client-puzzles-based prevention protocol was presented and a game-based formal method was used to verify the network's availability [14]. Based on Meadows's cost-based framework, Ramachandran [15] analyzed the performance of JFK protocol which is a DoS-resistant protocol. The methods above assessed the performance from defender's perspective only [5, 6, 12, 14, 15], and lacked cost considerations [5, 12, 13, 14].

DDoS defense is an economic-related problem. To withstand the DDoS attacks, security administrators should schedule their security budgets. Information security investment decision-making was a promising research field and has been previously addressed in a variety of ways. The difficult problem of this decision-making process is how to deal with the dynamic uncertainty. In a botnet attack, virtual bots could produce uncertainty, so Li et al. [16] modeled botnet masters and renters' relation as a profit-maximizing problem and given a solving method. Entrepreneurs who provided services to privacy-concerned consumers faced a decision on privacy preserving technology (PPT). It would made customers churn if no PPT measure was taken. Kantarcioglu et al. [17] solved this puzzle by using a framework, which combined copula functions and a Stackelberg leader-follower game. Penetration testing was also used to gathering decision-making information to reduce uncertainty in security investment [18]. The above methods have their specific application scenarios and

these preconditions have limited their use in DDoS defense strategies selection. In addition to reduce the uncertainty of the decision making process, how to balance between multiple goals was also a big research hot spot. Researchers have modeled and analyzed security investment trade-off based on linear programming method [1] and i* framework [19]. Böhme et al. [9] reviewed and summarized existing approaches.

Cyber-insurance provided a way to transfer financial risk associated with network and computer incidents to a third party [10]. Although cyber-insurance was an effective risk-averse method and was proposed more than 10 years ago [11], it only attracted wide attention from scientists recently. Lelarge et al. [11] discussed how insurance can encourage entities to increase security investment. Böhme et al. [20] studied the cyber-risks' correlation and modeled the market for cyber-insurance from both the supply-side and the demand-side. A comprehensive framework towards cyber-insurance was proposed, which formalized five primary elements: network environment, demand side, supply side, information structure and organizational environment [10]. These methods confirmed cyber-insurance's impact on security investment, but they assumed that security administrators considered security insurance as the only strategy. However, cyber-insurance was only a risk-averse option [21]. In this paper, we introduced cyber-insurance into DDoS defense and put it as an optional strategy for defenders.

Game theory is a subject that discusses one player how to choose its strategy while taking other players' strategies into account. It has been applied in many fields of information security. Mahimkar et al. [14] used a game-based formal method to analyze DoS prevention protocols. Static and dynamic Bayesian game approaches were respectively used for intrusion detection in wireless ad hoc networks [22]. To encourage security administrators to increase information security investment, various game-based methods were presented [17, 21, 23]. Wang et al. [24] analyzed the confidentiality and integrity of enterprise networks with a stochastic game nets model, while Lin et al. [25] employed a repeated two-way signaling game to model the multi-step attack-defense scenarios on confidentiality. Roy et al. [26] summarized the major game-based works in information security. These works were enlightening to our work, but every model has its own scope of application, and cannot be used in DDoS attack and defense scenarios. As a result, we formalized a Bayesian game model for DDoS attack and defense and given the definition of Bayesian Nash equilibrium which could be used for strategies selection.

3 Bayesian Game Model for DDoS Attack and Defense

In this section, a Bayesian game model for DDoS attack and defense is described. We also present how to compute the Nash equilibrium.

During DDoS attacks, the attacker employs many clients to flood a target network, while the defender who is usually the security administrator of the target network adopts many measures to defeat the attack and guarantees the regular business of the network. Each player does not know his opponent's investment budget and can only

get a probability distribution of another player's investment budget. We also assume every player is rational, which means he will take the action that could maximize his utility respecting another player's type and actions.

Definition 1 (Bayesian Game Model for DDoS Attack and Defense). A Bayesian game model for DDoS attack and defense (BGM-DAE) is represented as an eight-tuple vector BGM-DAE= (T_a, T_d, A_a, A_d, P_a, P_d, U_a, U_d) and the attacker and the defender are represented as *a* and *d* respectively. The meaning of every element is as follows.

(1) T_a={t_i, 1<=*i*<=*n* and *n* is a positive integer} denotes the type space of the attacker where the attacker's type number is *n* and t_i is the *i*-th type. In this paper we put the investment of a DDoS attack as the attacker's type. Similarly, T_d={t_j, 1<=*j*<=*m* and *m* is a positive integer } is the type space of the defender and we use the information security investment budget as the defender's type.

(2) A_a={$a_a(t_i)$} and A_d={$a_d(t_j)$} represent the action space of the attacker and the defender separately, where $a_a(t_i)$ (or $a_d(t_j)$) is an action that can be adopted by the attacker(or the defender) based on its type t_i (or t_j). $a_a(t_i)$={*Attack*, *Not Attack*}, which means the attacker can take the following actions according to its type t_i: *Attack* or *Not attack*. $a_d(t_j)$ should fulfill the following requirements:
-$a_d(t_j)$={{DP_1, DP_2 ,…, DP_k ,…, DP_l}}, where DP_k represents the budget invested into the *k*-th defense policy and the number of defense policies can be adopted by the defender is *l*.
-$DP_k \in [0, Max_b]$, if the information security investment budget of the defender is Max_b,
-$\sum DP_k \leq Max_b$.

(3) P_a and P_d are the priori belief space for each player respectively. P_a= {$p_a(t_i)$} and stratifying:
- $p_a(t_i)$ is the prior probability that the defender think the attacker belongs to the type t_i,
-$\sum p_a(t_i)=1$.
P_d is similar to P_a and we will not repeat the details.

(4) U_a and U_d stand for the utility function of each player separately. We will explain their calculation methods in later chapters.

Players of different types have different utility functions. Every player knows its own type but it only has a prior probability of other player's type. Specify to a representative DDoS attack and defense scenario, the attacker does not know the defending effect which is determined by defender's information security investment budget. Similarly, the defender does not know the attacker's type which is determined by attacker's investment.

Before given the definition of Bayesian Nash equilibrium, we first define strategy of the BGM-DAE.

Definition 2 (Strategy). In the BGM-DAE, a strategy of the attacker (or the defender) is a function $s_a(t_i)$ (or $s_d(t_j)$), which means when the attacker's (or the defender's) type is t_i(*or* t_j), it will perform an action $a_a(t_i)$(*or* $a_d(t_j)$) in the action set A_a(or A_d).

All strategies of a player constitute its strategy set and each player can select any strategy form its strategy set. However given another player's strategy, a rational player tends to choose the strategy which can maximize its utility and this strategy is called its optimal strategy. If all players choose their optimal strategy, the game will reach its equilibrium.

Definition 3 (Bayesian Nash Equilibrium). Bayesian Nash equilibrium of the BGM-DAE is a strategy vector s=(s^*_a, s^*_d), satisfying:

-For the attacker (or the defender), given every type t_i*(or* t_j*)* of its type space T_a*(or* T_d*)*, the strategy s^*_a (or s^*_d) maximizes its utility function; i.e., $s^*_a(t_i)$ solves

$$\max_{A_a(t_i)\in A_a} \sum_{t_j} \{u_a[a_a(t_i), s*_d(t_j); t_i] p_a(t_j \mid t_i)\}$$

and $s^*_d(t_j)$ solves

$$\max_{A_d(t_j)\in A_d} \sum_{t_i} \{u_d[s*_a(t_i), a_d(t_j); t_j] p_d(t_i \mid t_j)\}$$

where $p_a(t_j/t_i)$ (or $p_d(t_i/t_j)$) is the probability that when the attacker's (or the defender's) type is t_i(or t_j), he think the defender's(or the attacker's) type is t_j*(or* t_i*)*.

Bayesian Nash equilibrium is a type dependent strategic combination. In other words, given its own type and the probability distribution of another player's type, each player chooses the strategy which can maximize its expected payoff. For a Bayesian Nash Equilibrium, no player has an incentive to deviate from its equilibrium strategy. So Bayesian Nash equilibrium of the BGM-DAE can guide security administrators to select the most applicable defense strategies.

4 Utility Function

The previous chapter presents definitions of the BGM-DAE and we next will explain the utility functions. In a DDoS attack and defense scenario, both the attacker and the defender have cost and rewards and the different between these two is the utility. The attacker employs attack flows but gets rewards from the attack. The defender invests to adopt defense measures and suffers the attack which is also economic-related.

4.1 Utility Function of the Attacker

In DDoS attacks, the attacker makes use of many compromised hosts to send plentiful packets to the target network with the aim of exhausting the bandwidth resources. The attacker can compromise hosts one by one and then use them as attack-client, but this work is challenging and time-consuming. Meanwhile, highly social division of labor has made some people or organizations to provide attack flows and services [27] and this is the mainstream of current DDoS attacks. So, in this paper we only consider the latter attack form.

The utility function U_a of the attacker is defined as follows:

$$U_a = r_a(v_a) - c_a(b,d) = \alpha * v_a - \sum_{x=1}^{m} \beta * b_x * d_x \quad (1)$$

where $r_a(.)$ represents the attacker's rewards function of the DDoS attacks and $c_a(.)$ is the implementation cost function of these attacks. There are total m attacks, and $c_a(.)$ is a function of bandwidth and duration time of the attack flow[27]. For the attacker, $r_a(.)$ is proportional to the estimated value of the victim's value. α and β are constant adjustment parameter.

4.2 Utility Function of the Defender

Different from the attacker, the defender can take many defense measures to reduce impact of an attack. However, every defense measure has its own cost and performance. Numerous researchers have analyzed and evaluated the performance of DDoS defense policies [5, 6, 12, 13, 14, 15]. These works evaluate the performance of DDoS defense measures from the point of technical efficiency, but rarely consider the implementation cost. Moreover, no security measures can achieve absolute security and the above works do not consider how to handle the residual risk. Because cyber-insurance is an effective risk-averse method for residual risks, so we introduce cyber-insurance into DDoS defense to make up for the deficiency of traditional defense measures.

If the number of defense measures is n, then the utility function of the defender U_d can be defined as follows:

$$U_d = \sum_{i=1}^{n} (f_i(s_i) - h_i(s_i)) \quad (2)$$

where the i-th measure is represented by s_i. Function $f_i(.)$ is used for computing rewards from a measure while function $h_i(.)$ expresses the cost of a measure. It should be noted that when a measure is not adopted by the security administrators, then its cost and rewards are zero.

The DDoS defense measures can be used alone or in combination, which depends on the concrete attack and defense scene, the security budget and benefit of every measure. So security administrators should compute the utility of every possible measure permutation and choose the one that can maximize the defender's expected utility. Before given the calculation method, we should review the definition of DDoS defense strategies in this paper. Given the defender's type (represented by its information security investment budget), a defense strategy is a possible measure or measures combination with which the cost of these measures is within the budget.

$$\begin{cases} \max \quad U_d = \sum_{i=1}^{n} (f_i(s_i) - h_i(s_i)) \\ s.t. \ \sum_{i=1}^{n} h_i(s_i) <= B \\ \qquad h_i(s_i) >= B_i \\ \qquad B_i >= 0 \end{cases} \tag{3}$$

Nonlinear programming (NLP) is an important branch of Operations Research and provides powerful mathematical method for optimization choice. Therefore, we can use it to solve the optimal defense strategy selection problem. A typical NLP for DDoS defense strategies selection is shown in formula 3. U_d is the maximum expected utility of the combined defense strategies. B is the security budget and every measure has a minimum investment which is denoted by B_i. Other constraints may also exist according to the actual situation. There is not a general method suitable for all kinds of NLP problems, so we will choose the appropriate method based the algorithm's performance and the application environment.

Every possible DDoS defense measure has its own cost and rewards calculation formula. In this paper, we select three major measures: increasing network bandwidth (I-N-B), deploying DDoS filter (D-D-F) and adopting cyber-insurance (A-C-I). The reason we select these three defense measures is that they are typical defense mechanisms. DDoS defense mechanisms can be divided into attack prevention and attack reactive based on the active level [7]. Therefore, we use I-N-B as a representative of attack prevention and select D-D-F as a typical attack reactive measure. In order to handle the residual risk well, we employ A-C-I as an option strategy for the DDoS defender. It should be point that the defender can use a combination of the three defense strategies.

(1) Utility Function of Increasing Network Bandwidth

The defense measure I-N-B means that security administrators should continue to rent much more bandwidth, in order to provide an abundance of bandwidth to defeat DDoS threats. So its utility function U_{d1} is defined as follows:

$$U_{d1} = f_1(I\text{-}N\text{-}B) - h_1(I\text{-}N\text{-}B) \tag{4}$$

where $h_1(I\text{-}N\text{-}B)$ is used for computing the rewards and $f_1(I\text{-}N\text{-}B)$ is the calculation formula of the total cost.

Defense measures can reduce the potential damage of attacks, so its utility can be computed through the damage before and after adopting this measure. Potential damages can be calculated from economic, time and legal aspects [1, 2], so security administrators can compute the damages of a successful DDoS attack (denoted by v_0) if he knows the attack information. Several metrics have been proposed to evaluate the effectiveness of the DDoS defense measures, such as the percentage of attack packages dropped [13]. As bandwidth is the critical resources, we use the bandwidth

consumed by attack flows as the evaluation metric in our paper and other metrics can be used but the calculated methods will not be changed. So, $f_1(I\text{-}N\text{-}B)$ is computed by the following formula:

$$f_1(I\text{-}N\text{-}B)=\begin{cases}\theta & if(b_c+b_i)<b_x \\ v_0*\sum_{x=1}^{m}(p_x*\frac{d_x}{24}*(\frac{b_x}{b_c}-\frac{b_x}{b_c+b_i})) & if\ b_c\le b_x\le(b_c+b_i) \\ 0 & if\ b_c>b_x\end{cases} \quad (5)$$

If the sum between current network bandwidth b_c and increased network bandwidth b_i is less than the network bandwidth consumed by the attack flow, the reward of deploying I-N-B is θ where θ is a very small number. When b_c is already bigger than b_x which means increased network bandwidth is useless for DDoS defense, the reward is zero. Otherwise, we compute the reward using the formula in the middle, where the DDoS attack number is m, the x-th attack's duration tome is d_x, the frequency of this attack is p_x and the attack flow (the flow that crossed the filter if a filter is deployed) is b_x. The total cost of deploying I-N-B is computed by following formula:

$$h_1(I\text{-}N\text{-}B)=c_i/N+c_u*b_i \quad (6)$$

where c_i is cost of network rollout, c_u is cost of the annual fee for renting unit bandwidth and N is the using years.

(2) Utility Function of Deploying DDoS Filter

Similarly to the utility function of I-N-B, the utility function of D-D-F is defined as follows.

$$U_{d2}=f_2(D\text{-}D\text{-}F)-h_2(D\text{-}D\text{-}F) \quad (7)$$

Deploying DDoS filter usually means buying or updating a filter. The cost of D-D-F consists of two parts: deploying and maintaining cost c_d and purchase fee c_p, so its formula is:

$$h_2(D-D-F)=c_d+c_p(1+I)^N/N \quad (8)$$

where rate of interest is I and the filter can be used for N years.

If the maximum throughput Max_t of this filter is lowered than the attack traffic, the reward of D-D-F is a small value θ. Otherwise, the reward depends on the filter's efficiency. As the attack traffic that crossed the filter consumed the network bandwidth, so we employ false negative rate (FNR) to depict the filter's efficiency.

$$f_2(D\text{-}D\text{-}F)=\begin{cases}v_0*\sum_{x=1}^{m}(p_x*\frac{d_x}{24}*\frac{(1-FNR)*b_x}{b_c+b_i}) & if\ b_x<Max_t \\ \theta & else\end{cases} \quad (9)$$

(3) Utility Function of Adopting Cyber-Insurance

When adopting cyber-insurance, security administrators expend premium (the cost) and obtain compensation (the reward) from insurance company. Therefore, when DDoS attacks happen, the formula of A-C-I is as follows:

$$U_{d3} = f_3(A\text{-}C\text{-}I)\text{-}h_3(A\text{-}C\text{-}I) = f_3(e) - e \tag{10}$$

where the insurance expenses is e. As different insurance ways have different calculation formulas. Security administrator need to calculation the utility of adopting cyber-insurance according to the actual situation, we denote the calculation function as $f_3(e)$ here.

5 Two Problems

In section 3, we model typical DDoS attack and defense scenarios into the BGM-DAD and illustrate how to calculate the attacker and the defender's utility in chapter 4. Next, we will specify two problems usually faced by security administrators when choosing suitable defense strategies and give the problem-solving process using our model and methods above.

Problem 1. (Investment Budget and Budget Allocation (IBBA)). If the defender knows information about the attacker's investment (e.g. the information of the attack flow), security administrators should be how to choose the right budget and how to allocate its budget in order to maximize its payoff.

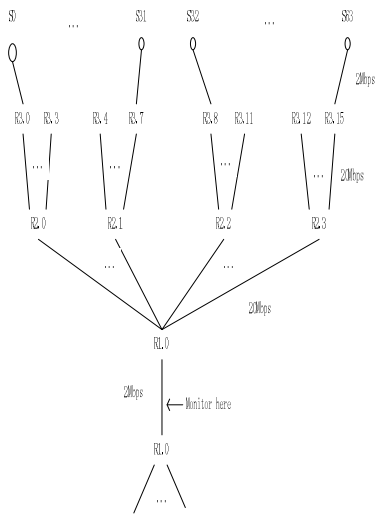


Fig. 1. The network topology

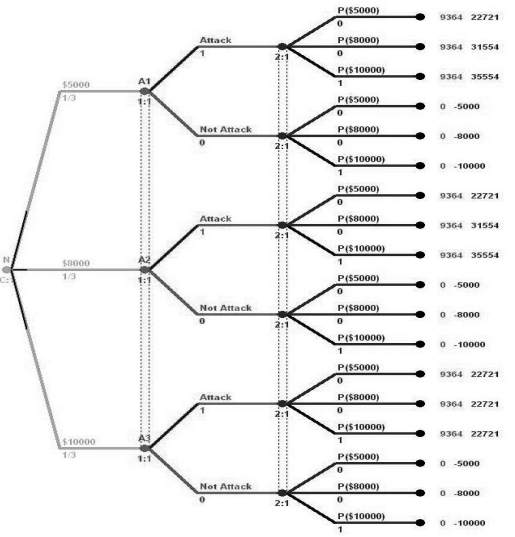


Fig. 2. The Bayesian game tree of IBBA problem

Problem 2. (Maximum Attack Information Estimation (MAIE)). If the defender's information security investment budget is known to the attacker, how to solve the attacker's most possible attacks.

In fact, the two problems above are two sides of a coin. The first problem assumes that the attacker's investment is fixed, so that the type of the attacker is known to the defender. In contrast, the type of the defender is common information in the second problem.

First of all, problem IBBA (or MAIE) can be modeled into the BGM-DAE in section 3 where the attacker (or the defender) only has one type and the defender (or the attacker) has a type set, of which the elements are the possible budgets (or the possible attack investment). Secondly, utilities of the attacker and the defender are computed using the formulas in chapter 4. And lastly, through computing the Bayesian Nash equilibrium of this BGM-DAE, we can get the solution of problem IBBA (or MAIE), which is the optimal strategy of the defender (or the attacker). We will illustrate the process in detail in the next chapter.

6 Experiment and Results

In this section, we will use an example to test and verify our model and methods. We borrowed a network topology from [28], of which 64 comprised hosts trying to flood a victim network with a bottleneck in bandwidth (2Mbps). The topology was implemented in simulator NS-2 [29]. We choose ns-2 because it can simulate the network topology and network flow well which is needed in our paper. The initial parameters of our experiments were described in table 1. The values of the initial parameters are adopted from a real experiment scene.

As described in section 5, security administrators usually confront two problems: IBBA and MAIE. We will next use IBBA as an example to illustrative our model and methods.

Table 1. Initial value and meaning of parameters in our experiments

Parameters	Meaning of parameters	Initial Value
α	Adjustment parameter of attacker's reward	0.1
β	Adjustment parameter of attacker's cost	0.1
θ	Reward of defense measures when they fail	0.01
b_c	Current bandwidth of the victim's network(Mbps)	2
b_i	Increased bandwidth by adopting I-N-B(Mbps)	1
B_i	Minimum budget of the *i*-th defense measure	0
v_a	Estimated value of the victim's value	100,000
v_0	Damages of a successful DDoS attack	10,000
c_i	Cost of network rollout	100
c_u	Annual network lease cost of unit bandwidth	1,000
c_d	Cost of deploying the filter	100
c_p	Cost of purchasing the filter	10,000
I	Annual interest rates	0.05
Max_t	Maximum throughput of the filter(Mbps)	1,000
FNR	False negative rate of the filter	0.02
N	Used years of the defense measure	5

(1) Resolving Process of IBBA

IBBA means the attacker's type is known to the defender. An estimate of the number of DDoS attacks was presented using a survey data from Arbor, McAfee and Forrester [30]. We use a similar data: there are two DDoS attacks, one happened 3.6 times and has a duration time 6 hours, the other happened 1.75 times and has a duration time 24 hours. Both these two attacks' traffic was 100Mbps. The results in this paper have been rounded to integer.

In this DDoS attack and defense scenario, the type (the attack investment) of the attacker is common information, which is equal to 0.1*100*(3.6*6+1.75*24) =636. The actions of the attacker are *Attack* and *Not Attack*. When the attacker chooses *Not Attack*, its utility is zero; otherwise its utility is 0.1*100000-636=9364. The type of the defender is its possible investment budgets and we assume there are three possible budgets, which are \$5,000, \$8,000 and \$10,000. The defender should schedule its information security investment budgets based on its type in order to maximize its expected utility. When investment budget is \$5,000, an NLP problem is showed in *NLP1*. Here we assume that $f_3(e)=3e$. Through resolving *NLP1*, we get its optimal solution, which is also the defender's maximum utility (22721). The optimal defense measures (determined by the investment of every defense measures) that the defender should be taken are: increasing network bandwidth with 2Mbps, deploying the DDoS filter and adopting cyber-insurance with premium \$407. Similarly, the maximum utilities of the defender are 31554(investment budget is \$8,000) and 35554 (investment budget is \$10,000) respectively. When the attacker does not carry out DDoS attacks, the utility of the attacker is zero and the payoff of the defender is its information security investment budget. Using the results above, we can form the following game tree using Gambit [31]. We assume the prior probability of the attacker is P_a= {*1/3*, *1/3*, *1/3*}.

$$\begin{cases} \max\ U_d = 26500 * \dfrac{2b_i}{2+b_i} + 2e - 2593 - 1000b_i \\ \text{s.t. } B_1 = 20 + 1000 * b_i >= 0 \\ \quad B_2 = 2573 \\ \quad B_3 = e >= 0 \\ \quad B_1 + B_2 + B_3 <= 5000 \end{cases} \tag{NLP1}$$

Through computing the Bayesian Nash equilibrium of the Bayesian game tree in Figure 2 using the definition 3, a strategy pair (Attack, (Not Defend with \$5,000, Not Defend with \$8,000, Defend with \$10,000), (1/3, 1/3, 1/3)) was found. This Nash equilibrium means: when the attacker chooses attack and the defender chooses defend with the security budget \$10,000, both the attacker and the defender gain their maximum utilities. Therefore, security administrators should choose defeat DDoS attacks with a budget \$10000 and deploying defense measures(denoted by P(\$10000) in the Bayesian game tree) are the optimal measures (determined by the investment of every defense measures: increasing network Bandwidth with \$4020, deploying DDoS filter with \$2573 and adopting cyber-insurance with premium \$3407) when the information security investment budget is \$10,000: increasing network Bandwidth (4Mbps), deploying the DDoS filter and adopting cyber-insurance (premium \$3407). So the model and methods in our paper can guide security administrators to select and deploy optimal strategies.

(2) The Effect of A-C-I

Cyber-insurance provides an effective way to transfer the residual risk to third-party. We next show its effect using an example.

We assume the DDoS attack flow is 1.2Gbps and the information security investment budget is $10,000. When security administrators did not employ A-C-I, its optimal strategies are: increasing network Bandwidth (I-N-B) with $7020, deploying DDoS filter (D-D-F) with $2573. As the maximum throughput of the filter is 1,000Mbps and the current network bandwidth is 9Mbps (the sum of current bandwidth(2Mbps) and increased bandwidth(7Mbps)), the utility of the defender is $2*\theta$=0.02. However, when cyber-insurance is used as a measure option, the optimal defense measures are showed above. The utility of the defender is 2*θ+3407*3-3407=6814.02, which is much more bigger than 2*θ. So cyber-insurance is an effective defense strategy for risk-averse, especially when attacks do not occur as predicted.

(3) Resolving Process of MAIE

MAIE means the defender's type is known to the attacker and its optimal defense measures are common information. We still assume the defender's budget is $10,000, so its utility is 35554 and its optimal measures are not changed. The type of the attacker is its possible attack investments which are related to the attack traffics and the duration time. We assume there are four possible attacks, of which the attack traffics are 100Mbps, 400Mbps, 700Mbps and 1Gbps. The duration time of these attacks are the same, which is 24 hours. The utilities of these attacks can be computed using formula 1 and the results are 9760, 9040, 8320, and 7600.

When the attack traffic is 100Mbps, the utility of the defender is 13554, which is computed by the following formula:

10000*1*(24/24)*(100*0.02/2-100*0.02/ (2+4))+10000*1*(24/24)*(1-100*0.02/(2+4))+3*3407-10000

When the attack traffic is 400Mbps and 700Mbps, the utilities are equal to 6888.01. This is because the traffics crossed the DDoS filter are already bigger than the network bandwidth. When the attack traffic is 1Gbps, both I-N-B and D-D-F have failed, so the utility of the defender changes to 221.02. We assume the prior probability of the defender is P_d= {*1/4*, *1/4*, *1/4*, *1/4*} and get a Bayesian Nash equilibrium ((Not Attack with 100Mbps, Attack with 400Mbps, Not Attack with 700Mbps, Not Attack with 1Gbps), Defend with $10,000, (*1/4*, *1/4*, *1/4*, *1/4*)). The Nash equilibrium means: the attacker chooses attack with 400Mbps and the defender chooses defend. This is because the target network's bottleneck is its bandwidth. Actually, when the attack traffic is equal to 300Mbps, the defense measure I-N-B has failed. As a result, if security administrators decide to increase its investment budgets, they should invest to increase the network bandwidth. Through resolve the problem MAIE, security administrators can get the most possible attacks (attacks with flow 400Mbps in this example) and find the way how to schedule his increased investment budgets.

Through the above experiments, we verified the effectiveness of model and methods in this paper and illustrated how the Bayesian Nash equilibrium can guide security administrators choose and deploy defense strategies.

7 Conclusion and Future Work

We formalize a Bayesian game model for DDoS attack and defense to help security administrators to choose the suitable defense measures. In the BGM-DAE, the attacker's private information is its investment and the security investment budget is the defender's type. Then two problems that security administrators usually faced when choose defense strategies are described, which can be model into our Bayesian game. Through compute the Nash equilibrium of the Bayesian game, security administrators can choose the optimal defense measures. Using a typical DDoS attack and defense scenario, our model and methods are verified.

The utility of attacker and defender depends on many factor, we will introduce much more influence factors, such as reputation and law penalty. In the practical environment, there may be more than one attack flow simultaneously and the attack flow can be changed over time, so we should extend our game model to include these scenarios. We will also carry out much more studies to improve the efficiency of our method.

Acknowledgement. This work was supported by the National Science-technology Support Plan of China (Grant No.2012BAK26B01), the National High-Tech Research and Development Plan of China (Grant No. SQ2013GX02D01211, 2011AA01A203), the National Natural Science Foundation of China (Grant No.61100226, 60970028), the Beijing Natural Science Foundation of China (Grant No.4122085).

References

1. Dewri, R., Poolsappasit, N., Ray, I., Whitley, D.: Optimal Security Hardening Using Multi-objective Optimization on Attack Tree Models of Networks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), pp. 204–213 (2007)
2. Butler, S.: Security Attribute Evaluation Method: A Cost-Benefit Approach. In: Proceedings of ICSE 2002 International Conference on Software Engineering, pp. 232–240 (2002)
3. Arora, K., Kumar, K., Sachdeva, M.: Impact Analysis of Recent DDoS Attacks. International Journal on Computer Science and Engineering (IJCSE) 3(2), 877–884 (2011)
4. Feinstein, L., Schnackenberg, D., Balupari, R., Kindred, D.: Statistical Approaches to DDoS Attack Detection and Response. In: Proceedings of the DARPA Information Survivability Conference and Exposition (2003)
5. Li, Q., Chang, E., Chan, M.: On the Effectiveness of DDoS Attacks on Statistical Filtering. In: Proceedings of INFOCOM 2005, pp. 1373–1383 (2005)
6. Kuznetsov, V., Sandström, H., Simkin, A.: An evaluation of Different IP Traceback Approaches. In: Deng, R., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 37–48. Springer, Heidelberg (2002)
7. Mirkovic, J., Reiher, P.: A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. ACM SIGCOMM Computer Communications Review 34(2), 39–54 (2004)
8. Douligeris, C., Mitrokotsa, A.: DDoS Attacks and Defense Mechanisms: Classification and State–of–the–Art. Computer Networks 44, 643–666 (2004)

9. Böhme, R.: Security Metrics and Security Investment Models. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) IWSEC 2010. LNCS, vol. 6434, pp. 10–24. Springer, Heidelberg (2010)
10. Böhme, R., Schwartz, G.: Modeling Cyber-Insurance: Towards A Unifying Framework. In: Workshop on the Economics of Information Security (WEIS). Harvard University, Cambridge (2010)
11. Lelarge, M., Bolot, J.: Economic Incentives to Increase Security in the Internet: The Case for Insurance. In: IEEE INFOCOM 2009, pp. 1494–1502 (2009)
12. Mao, Z., Sekar, V., Spatscheck, O., et al.: Analyzing Large DDoS Attacks Using Multiple Data Sources. In: Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense (LSAD), pp. 161–168 (2006)
13. Mirkovic, J., Arikan, E., Wei, S., Thomas, R., Fahmy, S., Reiher, P.: Benchmarks for DDoS defense evaluation. In: Military Communications Conference (2006)
14. Mahimkar, A., Shmatikov, V.: Game-based Analysis of Denial-of-Service Prevention Protocols. In: 18th IEEE Computer Security Foundations Workshop (CSFW), Aix-en-Provence, France, pp. 287–301. IEEE Computer Society, Los Alamitos (2005)
15. Ramachandran, V.: Analyzing DoS-Resistance of Protocols Using a Cost-Based Framework. Technical report, DCS/TR-1239, Yale University (2002)
16. Li, Z., Liao, Q., Striegel, A.: Botnet Economics: Uncertainty Matters. In: Johnson, M.E. (ed.) Managing Information Risk and the Economics of Security, pp. 245–267. Springer, New York (2008)
17. Kantarcioglu, M., Bensoussan, A., Hoe, S(C.): Investment in Privacy-Preserving Technologies under Uncertainty. In: Baras, J.S., Katz, J., Altman, E. (eds.) GameSec 2011. LNCS, vol. 7037, pp. 219–238. Springer, Heidelberg (2011)
18. Böhme, R., Félegyházi, M.: Optimal Information Security Investment with Penetration Testing. In: Alpcan, T., Buttyán, L., Baras, J.S. (eds.) GameSec 2010. LNCS, vol. 6442, pp. 21–37. Springer, Heidelberg (2010)
19. Elahi, G., Yu, E.: Modeling and Analysis of Security Trade-Offs - A Goal Oriented Approach. Data and Knowledge Engineering 68(7), 579–598 (2009); Special Issue: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.): ER 2007. LNCS, vol. 4801. Springer, Heidelberg (2007)
20. Böhme, R., Kataria, G.: Models and Measures for Correlation in Cyber-Insurance. In: Proceedings of the Fifth Annual Workshop on Economics and Information Security (WEIS 2006), Cambridge, UK (2006)
21. Johnson, B., Böhme, R., Grossklags, J.: Security Games with Market Insurance. In: Baras, J.S., Katz, J., Altman, E. (eds.) GameSec 2011. LNCS, vol. 7037, pp. 117–130. Springer, Heidelberg (2011)
22. Liu, Y., Comaniciu, C., Man, H.: A Bayesian Game Approach for Intrusion Detection in Wireless AD Hoc Networks. In: International Workshop on Game Theory for Communications and Networks (GameNets), pp. 3–14 (2006)
23. Huang, Y., Xianjun, G., Whinston, A.: Defeating DDoS Attacks by Fixing the Incentive Chain. ACM Transactions on Internet Technology 7(1), 1–5 (2007)
24. Wang, Y.Z., Lin, C., Wang, Y., Meng, K.: Security analysis of enterprise network based on Stochastic game nets model. In: ICC 2009 Communication and Information Systems Security Symposium (2009)
25. Lin, J., Liu, P., Jing, J.: Using Signaling Games to Model the Multi-step Attack-defense Scenarios on Confidentiality. In: Grossklags, J., Walrand, J. (eds.) GameSec 2012. LNCS, vol. 7638, pp. 118–137. Springer, Heidelberg (2012)

26. Roy, S., Ellis, C., Shiva, S., Dasgupta, D., Shandilya, V., Wu, Q.: A Survey of Game Theory as Applied to Network Security. In: 43rd Hawaii International Conference on System Sciences (HICSS), pp. 1–10 (2010)
27. Segura, V., Lahuerta, J.: Modeling the Economic Incentives of DDoS Attacks: Femtocell Case Study. In: Moore, T., et al. (eds.) Economic of Information Security and Privacy 2010, pp. 107–119. Springer Science + Business Media, LLC (2010)
28. Liu, P., Zang, W.: Incentive-based Modeling and Inference of Attacker Intent, Objectives, and Strategies. In: Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003), pp. 179–189. ACM, New York (2003)
29. Network Simulator, ns-2, http://nsnam.isi.edu/nsnam/index.php/Main_Page
30. Arbor Networks: The business Value of DDoS Protection. White Paper (2011)
31. McKelvey, R.D., McLennan, A.M., Turocy, T.L.: Gambit: Software Tools for Game Theory, Version 0.2010.09.01 (2010), http://www.gambit-project.org

Attribute Specified Identity-Based Encryption

Hao Xiong[1], Tsz Hon Yuen[1], Cong Zhang[1]*, Yi-Jun He[2,]**, and Siu Ming Yiu[1]

[1] Department of Computer Science, The University of Hong Kong, Hong Kong
{hxiong,thyuen,czhang2,smyiu}@cs.hku.hk
[2] Hong Kong R&D Centre for Logistics and Supply Chain Management Enabling Technologies, Hong Kong
ahe@lscm.hk

Abstract. Fine-grained access control of encrypted data without trusted third party is a challenging task. Using the simple attribute-based encryption has the problem of key escrow, since there exists a trusted authority who is able to generate the secret keys of all users. Delegating this ability to multiple authorities can only minimize the risk, but not eliminating the possibility that all authorities may collude. We develop a new cryptosystem called *Attribute Specified Identity-Based Encryption* (AS-IBE) to solve this problem. We employ the idea of certificateless encryption and extend it to the attribute-based setting. Each user chooses his own public and secret key pairs to use, in addition to his attribute-based secret key. Therefore, the authority cannot decrypt without the knowledge the user's own secret key. Yet, the resulting AS-IBE system has some fundamental differences with the original attribute-based encryption. In this paper, we give the security model for the new AS-IBE cryptosystems, and propose two variants for the construction, namely the key policy AS-IBE and ciphertext policy AS-IBE.

Keywords: identity-based encryption, attributes, certificateless public key encryption, key escrow.

1 Introduction

Identity-based cryptography was proposed by Shamir [12] in 1984. The public key is the identity of the user, such as his name, job title or email address. The advantage of identity-based cryptography is that the identity of the user is easily recognised by his public key, and hence it avoids the use of digital certificate to validate the public key in traditional public key cryptography. However, there are still some shortcomings for identity-based cryptography. Firstly, the identity-based secret key for every identity must be generated by a trusted third party called the *Private Key Generator* (PKG) using its own master secret key. Therefore the PKG can decrypt ciphertexts for all users in the system. This is known

* Supported by the National Natural Science Foundation of China under Grant No. 60970135 and 61170282.
** Corresponding author.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 60–74, 2013.

as the key escrow problem. Secondly, simply using the identity for encryption may not be sufficient for fine-grained access control. One may wants to encrypt a message to someone who satisfy a number of criteria at the same time.

There are some researches that works on the aforementioned problems. For the key escrow problem, the first practical identity-based encryption (IBE) [3] suggested to use multiple PKGs at the same time, such that the master secret key is distributed among multiple PKGs by a secret sharing scheme. Unless all PKGs collude, the users are protected from the key escrow problem. However, this does not provide complete protection against colluding PKGs. In addition, it requires an extra infrastructure and communication cost between users and different PKGs. Another line of research is to detect malicious PKGs. A malicious PKG can be caught if it sells the identity-based secret key of a victim [6], a blackbox which can decrypt arbitrary ciphertexts encrypted to the victim [7], or a signature signed on behalf of the victim [14]. However, there is no method to prevent or even detect if the PKG simply decrypts a ciphertext for the victim and reveals the plaintext.

For the problem of fine-grained access control, attributed-based encryption (ABE) [11] was proposed to allow messages to be encrypted by some policy or attributes, such that any user holding satisfying secret key can decrypt. For *key policy ABE* [8], ciphertexts are labeled with sets of attributes and secret keys are associated with access structures (i.e. policy) that control which ciphertexts a user is able to decrypt. For *ciphertext policy ABE* [2], an access structure would be associated to each ciphertext, while a user's secret key would be associated with a set of attributes. We demonstrate the power of ABE by the following example. Suppose Alice wants to encrypt a message to "Manager of Department A" or anyone in "Department B". Using ciphertext policy ABE, Alice can encrypt using the policy "(Manager AND Dept A) OR Dept B". Then Bob, having the attributes "Manager" and "Dept A" in his secret key, can decrypt the ciphertext. On the other hand, a manager in Department C and a secretary in Department A cannot collude to decrypt the ciphertext.

Similar to IBE, ABE also suffers from the key escrow problem, since the PKG is responsible to generate the secret keys for all users. The multi-authority solution is also extended to the ABE setting. Chase [4] proposed the use of multiple attribute authorities (AA) for managing each user's attributes. [4] assumed the presence of a single trusted central authority (CA) in addition to the AAs. This CA did not manage any attributes, but was responsible for issuing each user a unique key. Therefore, the key escrow problem does not apply to the AAs. Liu *et al.* [10] extended the multi-authority setting to the CAs. Chase and Chow [5] demonstrated how to remove the CA using a distributed pseudorandom function. Lewko and Waters [9] showed that any party can become an authority and there is no requirement for any global coordination other than the creation of an initial set of common reference parameters. In the security models of these schemes, it still requires the assumption that some authorities remain uncorrupted.

Our Contribution. In this paper, we avoid the key escrow problem and retain the benefit of attribute-based encryption at the same time. The motivation of our

scheme can be illustrated by the following example. We consider the applications that absolute secrecy is required and the sender wants to encrypt the message according to some policy (or attributes). For example, Carol wants to send a letter of complaint for a staff in the IT department of the company. Carol wants the letter to be read by "(Senior Manager AND HR Department) OR Company Director". Since the secret keys for all staff in the company may be distributed by the IT department, Carol does not want to simply use ABE with respect to the aforementioned policy. Eventually Carol is given a public key by someone from the company who claimed to satisfy the above policy. Now Carol faces a dilemma, either encrypts the message by ABE which faces the key escrow problem, or encrypts the message by certificateless encryption which the ciphertext may be decrypted by someone who actually does not satisfy the policy.

We propose the notion of *attribute specified identity-based encryption* (AS-IBE) as the solution to this problem. Only the person who has the secret key corresponding to *both* the public key *and* the attributes can decrypt the ciphertext. This feature cannot be achieved by either the certificateless encryption or ABE.

The proposed AS-IBE notion is similar to the certificateless encryption [1], except that we extend the identity-based construction to the more expressive attributes and policy. Therefore, AS-IBE provides a more fine-grained access control. In addition, the sender does not need to know the exact attributes held by the secret key owner; he only requires that his policy is satisfied by the secret key owner. The *attribute specified identify-based encryption* will also provide some benefits to the receiver. Certificateless encryption requires the identity to be tightly coupled with the public key. AS-IBE, on the other hand, is more flexible when there is multiple attributes (or identities) for an user. Using the aforementioned example of Bob, he needs to generate three different identity-based secret keys for "Manager", "Dept A" and "Manager AND Dept A" when using certificateless encryption, since Bob is not sure which identity the sender will use to encrypt a message in the future.

In this paper, we first define the notion of attribute specified identity-based encryption and the security model for it. Our notion is defined upon the *ciphertext policy* version, such that an access structure is associated to each ciphertext, while the secret key is associated with a set of attributes. We then give a construction of ciphertext policy AS-IBE. After that, we extend our definition to *key policy* and gives a construction of key policy AS-IBE.

Although our scheme is attribute specific, our AS-IBE has some fundamental differences with ABE, apart from the key escrow problem. We can view ABE as a one-to-many encryption scheme, since one sender can encrypt a message according to a policy (or a set of attributes) and multiple users with satisfying secret keys can decrypt. On the other hand, our AS-IBE is a one-to-one encryption scheme, since the sender also requires a specific public key to encrypt a message. Only the user with the secret key corresponding to the public key and a set of satisfying attributes (or a satisfying policy) can decrypt. We leave the one-to-many version of AS-IBE as an interesting open problem.

2 Backgrounds

2.1 Notation of Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map such that for all $g \in \mathbb{G}, a, b \in \mathbb{Z}_p$, we have $e(g^a, g^b) = e(g, g)^{ab}$. We require that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and the bilinear map e are computable in polynomial time.

Decisional q-parallel Bilinear Diffie-Hellman Exponent (BDHE) Assumption [13]. Given the following group $\mathbb{G}$ elements: $g, g^s, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, \{g^{asb_k/b_j}, \ldots, g^{a^q sb_k/b_j}\}_{j,k\in[1,q],j\neq q}, \{g^{sb_j}, g^{a/b_j} \ldots, g^{a^q/b_j}, g^{a^{q+2}/b_j}, \ldots, g^{a^{2q}/b_j}\}_{j\in[1,q]}$ and $T \in \mathbb{G}_T$, for some unknown $a, s, b_1, \ldots b_q \in \mathbb{Z}_p$, no polynomial time adversary can decide if $T = e(g, g)^{a^{q+1}s}$ or $T \overset{R}{\leftarrow} \mathbb{G}_T$.

Decisional Bilinear Diffie-Hellman (BDH) Assumption. Given $g, g^a, g^b, g^c \in \mathbb{G}$ for some unknown $a, b, c \in \mathbb{Z}_p$, no polynomial time adversary can decide if $T = e(g, g)^{abc}$ or $T \overset{R}{\leftarrow} \mathbb{G}_T$.

2.2 Access Structure

We review some definitions related to access structures from [13].

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

In our context, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes. We restrict our attention to monotone access structures. General access structures can be realized from them [13].

2.3 Linear Secret Sharing Schemes

We will make essential use of linear secret-sharing schemes on our first construction, ciphertext-policy AS-IBE. We adapt the definitions in [13]:

Definition 2. (Linear Secret-Sharing Schemes (LSSS)) *A secret-sharing scheme Π over a set of $\mathcal{P}$ is called* linear *(over $\mathbb{Z}_p$) if*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix an $\boldsymbol{M}$ with ℓ rows and n columns called the share-generating matrix for Π. For all $i = 1, ..., \ell$, the i'th row of $\boldsymbol{M}$ we let the function ρ defined the party labeling row i as $\rho(i)$. When we consider the column vector $\boldsymbol{v} = (s, r_2, ..., r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, ..., r_n \in \mathbb{Z}_p$ are randomly chosen, then $\boldsymbol{Mv}$ is the vector of ℓ shares of the secret s according to Π. The share $(\boldsymbol{Mv})_i$ belongs to party $\rho(i)$.*

Every LSSS scheme Π for the access structure $\mathbb{A}$ enjoys the *linear reconstruction* property, defined as follows: Let $S \in \mathbb{A}$ be any authorized set, and let $I := \{i : \rho(i) \in S\} \subset \{1, 2, ..., \ell\}$. Then, there exist constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} w_i \lambda_i = s$. Furthermore, these constants $\{w_i\}$ can be found in time polynomial in $|\boldsymbol{M}|$. We define the Lagrange coefficient $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$ for $i \in \mathbb{Z}_p$, $S \subset \mathbb{Z}_p$.

Note on Convention. We use the convention that the vector $(1, 0, 0, \ldots, 0)$ is the "target" vector for any LSSS. For any satisfying set of rows I in $\boldsymbol{M}$, the target vector is in the span of I. For any unauthorized set of rows I, the target vector is not in the span of the rows of the set I. Moreover, there will exist a vector w such that $w \cdot (1, 0, 0, \ldots, 0) = -1$ and $w \cdot M_i = 0$ for all $i \in I$.

2.4 Access Trees

Prior works on ABE (e.g., [8]) typically described access formulas in terms of binary trees. We will also use access tree for our second construction, key-policy AS-IBE. We review the definition of access tree from [8]. Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. To facilitate working with the access trees, we define a few functions. We denote the parent of the node x in the tree by parent(x). The function att(x) is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree. The access tree $\mathcal{T}$ also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to num. The function index(x) returns such a number associated with the node x. Where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Satisfying an Access Tree. Let $\mathcal{T}$ be an access tree with root r. Denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at the node x. Hence $\mathcal{T}$ is the same as $\mathcal{T}_r$. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$ recursively as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(\gamma)$ for all children x' of node x. $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 if and only if att$(x) \in \gamma$.

Conversion. Using standard techniques one can convert any monotonic boolean formula into an LSSS representation. An access tree of ℓ nodes will result in an LSSS matrix of ℓ rows. This conversion is in the appendix of [9].

3 Security Model

An attribute specific attribute-based encryption (AS-IBE) scheme consists of the algorithms: `Setup`, `Extract-Partial-Private-Key`, `Set-Secret-Value`, `Set-Private-Key`, `Set-Public-Key`, `Encrypt` and `Decrypt`.

- $\texttt{Setup}(1^\lambda, U)$. It takes as input a security parameter 1^λ and an attribute universe description U. It outputs the public parameters mpk and a master key msk.
- $\texttt{Extract-Partial-Private-Key}(\mathsf{msk}, S)$. It takes as input msk and a set of attributes S that describe the key. It outputs a private key d_S.
- $\texttt{Set-Secret-Value}(\mathsf{mpk}, S)$: It takes as input mpk and a set of attributes S. It outputs a secret value w_S.
- $\texttt{Set-Private-Key}(\mathsf{mpk}, d_S, w_S)$: It takes as input mpk, d_S, w_S and outputs a full private key sk_S.
- $\texttt{Set-Public-Key}(\mathsf{mpk}, w_S)$: It takes as input mpk, w_S, and outputs his public key pk_S.
- $\texttt{Encrypt}(\mathsf{mpk}, \mathcal{M}, \mathbb{A}, \mathsf{pk}_S)$. It takes as input mpk, a message $\mathcal{M}$, an access structure $\mathbb{A}$ over the universe of attributes and a public key pk_S. It encrypts $\mathcal{M}$ and produces a ciphertext CT for a user pk_S such that only a user that possesses a set of attributes S satisfying the access structure will be able to decrypt the message. Assume that the ciphertext implicitly contains $\mathbb{A}$.
- $\texttt{Decrypt}(\mathsf{mpk}, CT, \mathsf{sk}_S)$. It takes as input mpk, a ciphertext CT, which contains an access structure $\mathbb{A}$, and a private key sk_S for a set S of attributes. If S satisfies $\mathbb{A}$ then it decrypts the ciphertext and return a message $\mathcal{M}$.

3.1 Chosen Plaintext Security for AS-IBE

We now define the security model for the AS-IBE scheme. It is a modified version of the usual IND-CPA security for certificateless encryption.

We first define a list of oracles that the attacker can query. With a **Create** query, the attacker asks the challenger $\mathcal{C}$ to create a public key, a partial private key and a full private key for a set of attributes S. The attacker receives a unique handle-reference to the generated keys, so that the attacker can refer to it later. The challenger keeps a counter for the handle H and a set Υ that holds tuples of handles, attributes, public keys, partial private keys and full private keys. The oracles are described as follows:

Create: On input S, $\mathcal{C}$ runs $d_S \leftarrow \texttt{Extract-Partial-Private-Key}\ (\mathsf{msk}, S)$, $w_S \leftarrow \texttt{Set-Secret-Value}(\mathsf{mpk}, S)$, $\mathsf{sk}_S \leftarrow \texttt{Set-Private-Key}(\mathsf{mpk}, d_S, w_S)$ and $\mathsf{pk}_S \leftarrow \texttt{Set-Public-Key}(\mathsf{mpk}, w_S)$. It adds $(H+1, S, \mathsf{pk}_S, w_S, d_S, \mathsf{sk}_S)$ to the set Υ and returns $H+1$. It updates the handle counter $H \leftarrow H+1$.

Extract Partial Private Key: On input a handle h, $\mathcal{C}$ scans Υ to find the requested entry and returns the corresponding partial private key d_{ID}.

Extract Private Key: On input a handle h, $\mathcal{C}$ scans Υ to find the requested entry and returns the corresponding full private key sk_S. It is unreasonable to expect the $\mathcal{C}$ to return the private key if the corresponding pk_S is replaced.

Request Public Key: On input a handle h, $\mathcal{C}$ scans Υ to find the requested entry and returns the corresponding public key pk_S.

Replace Public Key: $\mathcal{A}$ can repeatedly replace the public key for any set of attributes S with any value pk' of its choice. $\mathcal{C}$ scans Υ to find the entry for $(h, S, \mathsf{pk}_S, w_S, d_S, \mathsf{sk}_S)$ and replaces it with $(h, S, \mathsf{pk}', w_S, d_S, \perp)$.

Denote the challenge public key as pk^* and the corresponding entry in Υ as $(h, S^*, \mathsf{pk}^*, w_S^*, d_S^*, \mathsf{sk}_S^*)$. Note that (h, pk^*) cannot be the input to the Replace Public Key oracle by $\mathcal{A}$ before the challenge phase and extract the partial key for the set of attributes S which satisfies the challenge access structure $\mathbb{A}^*$ in some phase: this would enable $\mathcal{A}$ to receive a challenge ciphertext under a public key for which it would compute the private key.

We will specify two types of adversary as follows:

Type I Adversary: Such an adversary $\mathcal{A}_I$ does not have access to the master secret key. However $\mathcal{A}_I$ may create handles, request public keys and replace public keys, extract partial private key and private keys. We make several restrictions on such a $\mathcal{A}_I$ adversary:
- $\mathcal{A}_I$ cannot extract the private key for S^* at any point.
- $\mathcal{A}_I$ cannot both replace the public key pk^* for S^* before the challenge phase and extract the partial private key for the set of attributes S which satisfies $\mathbb{A}^*$ in some phase.

Type II Adversary: Such an adversary $\mathcal{A}_{II}$ does have access to master key, but cannot replace any public keys of entities. $\mathcal{A}_{II}$ can compute partial private keys for itself by using master key. It can create handles and extract full private keys. The restrictions on this type of adversary are:
- $\mathcal{A}_{II}$ cannot extract the private key for S^* at any point.
- $\mathcal{A}_{II}$ cannot replace public key at any point.

We say that a AS-IBE scheme is semantically secure against an adaptive chosen plaintext attack if no polynomially bounded adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage in the following game:

- **Setup:** The challenger $\mathcal{C}$ takes the security parameters 1^λ and runs $\mathtt{Setup}(1^\lambda)$. It gives $\mathcal{A}$ the master public key mpk. If $\mathcal{A}$ is of Type I, then the challenger keeps the msk to itself. If $\mathcal{A}$ is of Type II, $\mathcal{C}$ gives msk to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ issues a sequence of requests which can be adaptive subject to the restrictions defined above.
- **Challenge Phase:** Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two challenge messages $\mathcal{M}_0^*, \mathcal{M}_1^*$, the challenge public key pk^*, and the challenge access structure $\mathbb{A}^*$. $\mathcal{C}$ then picks a random bit $b \in \{0, 1\}$ and computes $C^* \leftarrow \mathtt{Encrypt}(\mathsf{mpk}, \mathcal{M}_b^*, \mathbb{A}^*, \mathsf{pk}^*)$. It returns C^* to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ issues a second sequence of request as in Phase 1, against subject to the rules on defined above. Furthermore, no leak query is allowed.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

The adversary wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ in this game as $\mathsf{Adv}_{\mathcal{A}}(1^\lambda) = |\Pr[b = b'] - \frac{1}{2}|$, where the probability is over all random bits used by the challenger and the adversary.

Definition 3. *An AS-IBE is Type I IND-CPA secure if for all PPT Type I adversaries $\mathcal{A}_I$, it is true that $\mathsf{Adv}_{\mathcal{A}_I}(1^\lambda) \leq \mathsf{negl}(\lambda)$.*

An AS-IBE is Type II IND-CPA secure if for all PPT Type II adversaries $\mathcal{A}_{II}$, it is true that $\mathsf{Adv}_{\mathcal{A}_{II}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

We say that a system is *selectively* secure if we add an **Init** phase before **Setup** where the adversary commits to the challenge access structure $\mathbb{A}^*$.

3.2 Extension to Key Policy

The above definition uses the *ciphertext policy* version of AS-IBE. We can similarly define the *key policy* version for AS-IBE, simply by interchanging the position of attributes and access structures in the previous section. That is, each key is now associated with an access structure, and each ciphertext is now encrypted by a set of attributes.

For the chosen plaintext security game, the adversary $\mathcal{A}$ will output a set of challenge attributes S^* (instead of the challenge access structure). $\mathcal{A}$ cannot extract the private key for any access structure $\mathbb{A}$ such that S^* satisfies $\mathbb{A}$. All other restrictions to $\mathcal{A}$ are changed similarly. The selective security is defined as $\mathcal{A}$ commits to the set of challenge attributes S^* before the **Setup** phase.

4 Our Ciphertext Policy AS-IBE Construction

We now give our main construction that is efficient , realizes expressive functionality and is provably secure under a concrete, non-interactive assumption. Our construction is based on the ciphertext policy ABE in [13].

In our construction the encryption algorithm will take as input a LSSS access matrix M, a function ρ (as defined as §2.3) and distribute a random exponent $s \in \mathbb{Z}_p$ according to (M, ρ). Private keys are randomized to avoid collusion attack. Our construction is as follows.

$\texttt{Setup}(1^\lambda, U)$. It takes as input the number of attributes in the system. It then chooses a bilinear group $\mathbb{G}$ of prime order p, a generator g and U random group elements $h_1, ..., h_U \in \mathbb{G}$ that are associated with the U attributes in the system. In addition, it chooses random exponents α, $a \in \mathbb{Z}_p$.

The authority sets $\mathsf{msk} = g^\alpha$ as the master secret key. The public key is published as $\mathsf{mpk} = (g, e(g,g)^\alpha, g^a, h_1, ..., h_U)$.

$\texttt{Extract-Partial-Private-Key}(\mathsf{msk}, S)$. It takes as input the master secret key and a set S of attributes. The algorithm first chooses a random $t \in \mathbb{Z}_p$. It creates the partial private key d_S as

$$K = g^\alpha g^{at}, \quad L' = g^t, \quad \forall x \in S \; K_x = h_x^t.$$

$\texttt{Set-Secret-Value}(\mathsf{mpk}, S)$. It takes as input mpk and a set of attributes S. It randomly choose a number $b \in \mathbb{Z}_p$ and sets the a secret value $w_S = b$.

$\texttt{Set-Private-Key}(\mathsf{mpk}, d_S, w_S)$: It takes as input mpk, d_S, w_S and outputs a full private key

$$\mathsf{sk}_S = (K = g^\alpha g^{at}, \quad L = L'^b = g^{bt}, \quad K_x = h_x^t \quad \forall x \in S).$$

$\texttt{Set-Public-Key}(\mathsf{mpk}, w_S)$. It takes as input mpk, w_S, and outputs his public key $\mathsf{pk}_S = (e(g,g)^{\alpha b}, g^b)$.

$\texttt{Encrypt}(\mathsf{mpk}, \mathcal{M}, (M, \rho), \mathsf{pk}_S)$. It takes as input mpk, a message $\mathcal{M}$, an LSSS access structure (M, ρ) and the user's public key pk_S. Let M be an $\ell \times n$ matrix.

and the function ρ associates rows of M to attributes. The algorithm first chooses a random vector $\boldsymbol{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s. For i = 1 to ℓ, it calculates $\lambda_i = \boldsymbol{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M. In addition, the algorithm chooses random $r_1, ..., r_\ell \in \mathbb{Z}_p$. The ciphertext is $CT =$

$$(C_1 = g^{a\lambda_1} h_{\rho(1)}^{-r_1},\ D_1 = g^{br_1}), \quad \ldots, \quad (C_\ell = g^{a\lambda_\ell} h_{\rho(\ell)}^{-r_n},\ D_\ell = g^{br_\ell}),$$
$$C = \mathcal{M} \cdot e(g,g)^{\alpha bs}, \quad C' = g^s,$$

along with a description of (M, ρ).

$\mathtt{Decrypt}(\mathsf{mpk}, CT, \mathsf{sk}_S)$. It takes as input a ciphertext CT for access structure (M, ρ) and a private key for a set S. Suppose that S satisfies (M, ρ) and let $I := \{i : \rho(i) \in S\} \subset \{1, 2, ..., \ell\}$. Then, let $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} w_i \lambda_i = s$. (Note there could potentially be different ways of choosing the w_i values.)

The decryption algorithm first computes

$$\begin{aligned} & e(C', K) \Big/ \prod\nolimits_{i \in I} \left(e(C_i, L) e(D_i, K_{\rho(i)})\right)^{w_i}) \\ =& e(g,g)^{\alpha bs} e(g,g)^{abst} \Big/ \prod\nolimits_{i \in I} e(g,g)^{tab\lambda_i w_i} = e(g,g)^{\alpha bs}. \end{aligned}$$

It can then divide out this value from C and obtain the message $\mathcal{M}$.

Theorem 1. *Suppose the decisional q-parallel BDHE assumption holds. Then our ciphertext-policy AS-IBE is selectively secure against IND-CPA Type I adversary with a challenge matrix of size $\ell^* \times n^*$, where $\ell^*, n^* \leq q$.*

Proof. Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage ϵ in the selective security game against our construction. Moreover, suppose it chooses a challenge matrix M^* where both dimensions are at most q. There are two possible ways for $\mathcal{A}$ to win:

1. Win by the public key pk^* replaced by $\mathcal{A}$. Therefore $\mathcal{A}$ is not allowed to ask for the partial private key for the set of attributes S which satisfies $\mathbb{A}^*$.
2. Win by the public key pk^* which was not replaced by $\mathcal{A}$.

We show how to build a simulator, $\mathcal{B}$, to handle both cases. We first show the simulation for the first case.

Init. The simulator $\mathcal{B}$ takes in a decisional q-parallel BDHE problem instance. The adversary $\mathcal{A}$ gives the challenge access structure (M^*, ρ^*), where M^* is a $\ell^* \times n^*$ matrix. Denote M_i^* as the ith row of M^*.

Setup. $\mathcal{B}$ chooses random $\alpha' \in_R \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $e(g,g)^\alpha = e(g^a, g^{a^q}) e(g,g)^{\alpha'}$. For each $x \in [1, U]$, choose a random value z_x. Let X denote the set of indices i, such that $\rho^*(i) = x$. $\mathcal{B}$ programs h_x as:

$$h_x = g^{z_x} \prod_{i \in X} g^{aM_{i,1}^*/b_i} \cdot g^{a^2 M_{i,2}^*/b_i} \cdots g^{a^{n^*} M_{i,n^*}^*/b_i}.$$

Note that if $X = \emptyset$, then we have $h_x = g^{z_x}$. Also note that the parameters are distributed randomly due to the g^{z_x} value.

Phase 1. Consider the Create query. The simulator $\mathcal{B}$ runs `Set-Secret-Value`, `Set-Private-Key` and `Set-Public-Key` honestly according to the algorithm. $\mathcal{B}$ does not need to calculate the partial private key for the set of attributes S which satisfies $\mathbb{A}^*$. It leaves a $\perp$ symbol in the corresponding entry in Υ.

The simulator $\mathcal{B}$ only needs to calculate the partial private key for the set of attributes S which does not satisfy $\mathbb{A}^*$. $\mathcal{B}$ first chooses a random $r \in \mathbb{Z}_p$. Then it finds a vector $\boldsymbol{w} = (w_1, \ldots, w_{n^*}) \in \mathbb{Z}_p^{n^*}$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have that $M_i^* \cdot \boldsymbol{w} = 0$. By the definition of a LSSS such a vector must exist. The simulator begins by setting

$$L' = g^r \prod_{i=1}^{n^*} (g^{a^{q+1-i}})^{w_i}, \quad K = g^{\alpha'} g^{ar} \prod_{i=2}^{n^*} (g^{a^{q+2-i}})^{w_i}.$$

It implicitly defines $t = r + w_1 a^q + w_2 a^{q-1} + \ldots + w_{n^*} a^{q-n^*+1}$. Now we must calculate $K_x \forall x \in S$. First, we consider $x \in S$ for which there is no i such that $\rho(i) = x$. For those we can simply let $K_x = L^{z_x}$ for some $z_x \in_R \mathbb{Z}_p$. Let X be the set of all i such that $\rho^*(i) = x$. The simulator creates K_x in this case as follows.

$$K_x = L^{z_x} \prod_{i \in X} \prod_{j=1}^{n^*} \left(g^{(a^j/b_i)r} \prod_{\substack{k=1,\ldots,n^* \\ k \neq j}} (g^{aq+1+j-k/b_i})^{w_k} \right)^{M_{i,j}^*}.$$

Observe that there are no terms of the form g^{a^{q+1}/b_i} since $M_i^* \cdot \boldsymbol{w} = 0$. Therefore the partial private key is correctly generated for S.

Challenge. The adversary gives two messages $\mathcal{M}_0^*, \mathcal{M}_1^*$ and the challenge public key pk^* to $\mathcal{B}$. $\mathcal{B}$ flips a coin β and retrieves the corresponding w_S^* from Υ. It creates $C = \mathcal{M}_\beta^* \cdot (T \cdot e(g^s, g^{\alpha'}))^{w_S^*}$ and $C' = g^s$. $\mathcal{B}$ will choose random $y_2', \ldots, y_{n^*}', r_1', \ldots, r_{\ell^*}' \in \mathbb{Z}_p$. For $i = 1, \ldots, \ell^*$, we define R_i as the set of all $k \neq i$ such that $\rho^*(i) = \rho^*(k)$. In other words, the set of all other row indices that have the same attribute as row i. $\mathcal{B}$ calculates

$$C_i = h_{\rho^*(i)}^{r_i'} \left(\prod_{j=2}^{n^*} (g^a)^{M_{i,j}^* y_j'} \right) (g^{b_i s})^{-z_{\rho^*(i)}} \cdot \left(\prod_{k \in R_i} \prod_{j=1}^{n^*} (g^{a^j \cdot s \cdot (b_i/b_k)})^{M_{k,j}^*} \right).$$

and $D_i = g^{-r_i' w_S^*} g^{-s b_i w_S^*}$ Then it implicitly sets $r_i = -r_i' - s b_i$ and $\boldsymbol{v} = (s, sa + y_2', \ldots, sa^{n^*-1} + y_{n^*}')$.

Phase 2. Same as phase 1.

Output. $\mathcal{A}$ will eventually output a guess β' of β. $\mathcal{B}$ then outputs 0 to guess that $T = e(g,g)^{a^{q+1}s}$ if $\beta' = \beta$; otherwise, it outputs 1 to guess that T is a random group element in $\mathbb{G}_T$. If $\mathcal{A}$ can win this game with advantage ϵ, than $\mathcal{B}$ can solves the decisional q-parallel BDHE problem with probability ϵ.

We now show the simulation for the second case, where the public key pk^* was not replaced by $\mathcal{A}$.

Init. The simulator $\mathcal{B}$ takes in a decisional BDH challenge $(\boldsymbol{y} = (g, g^\alpha, g^\beta, g^\gamma), T)$. The adversary $\mathcal{A}$ gives the challenge access structure (M^*, ρ^*), where M^* is a $\ell^* \times n^*$ matrix. Denote M_i^* as the ith row of M^*.

Setup. $\mathcal{B}$ uses g^α as msk and generates mpk accordingly. $\mathcal{B}$ sends mpk to $\mathcal{A}$.

Phase 1. Consider the Create query. On input a set of attributes S, the simulator $\mathcal{B}$ randomly picks $t_S, r_S \in \mathbb{Z}_p$ and calculates

$$d_S = (K = g^\alpha g^{at_S}, \quad L' = g_S^t, \quad \forall x \in S \ K_x = h_x^{t_S}),$$

$\mathsf{sk}_S = (K, L = g^{\beta r_S t_S}, \ \forall x \in S \ K_x), \mathsf{pk}_S = (e(g^\alpha, g^\beta)^{r_S}, g^{\beta r_S})$. It adds the tuple $(H+1, S, \mathsf{pk}_S, \perp, d_S, \mathsf{sk}_S)$ to the set Υ and returns $H+1$. It updates the handle counter $H \leftarrow H+1$.

Challenge. The adversary gives two messages $\mathcal{M}_0^*, \mathcal{M}_1^*$ and the challenge public key $\mathsf{pk}^* = (e(g^\alpha, g^\beta)^{r_S^*}, g^{\beta r_S^*})$ to $\mathcal{B}$. $\mathcal{B}$ knows r_S^* since the public key was not replaced by $\mathcal{A}$. $\mathcal{B}$ flips a coin b. It creates $C = \mathcal{M}_b^* \cdot T^{r_S^*}$ and $C' = g^\gamma$. $\mathcal{B}$ chooses some random $y_2, \ldots, y_{n^*}, r_1, ..., r_{\ell^*} \in \mathbb{Z}_p$. Then for $i \in [1, \ell^*]$, calculate

$$C_i = (g^\gamma)^{aM_{i,1}} g^{a\sum_{j=2}^{n^*} y_j M_{i,j}^*} h_{\rho(i)}^{-r_i}, \quad D_i = g^{\beta r_S^* r_i}.$$

It implicitly defines $\boldsymbol{v} = (\gamma, y_2, \ldots, y_{n^*})$.

Phase 2. Same as phase 1.

Output. The adversary will eventually output a guess b' of b. The simulator then outputs 0 to guess that $T = e(g,g)^{\alpha\beta\gamma}$ if $b' = b$; otherwise, it outputs 1 to guess that T is a random group element in $\mathbb{G}_T$. If $\mathcal{A}$ can win this game with advantage ϵ, than $\mathcal{B}$ can solves the decisional BDH problem with probability ϵ.

Note that if one can solve the decisional BDH problem with the input instance $(g, g^a, g^{a^q}, g^s, T)$, then he can solve the decisional q-parallel BDHE problem. Therefore we only need the decisional q-parallel BDHE assumption holds. □

Theorem 2. *Suppose the decisional BDH assumption holds. Then our ciphertext-policy AS-IBE is selectively secure against IND-CPA Type II adversary.*

Proof. It is almost the same as the second case of the above proof, except that msk can be given to $\mathcal{A}$ directly in the Setup phase. □

5 Our Key Policy AS-IBE Construction

Apart from the ciphertext policy AS-IBE, it is constructive to give a concrete key policy AS-IBE scheme, since these two schemes are useful in different contexts. Our construction is based on the key policy ABE in [8]. The partial secret key

is defined upon certain access tree defined as §2.4, while the ciphertexts are associated with attributes. We give our construction as follows.

$\texttt{Setup}(1^\lambda, U)$. It takes as input the number of attributes in the system. Now, for each attribute $i \in [1, U]$, choose a number t_i uniformly at random from $\mathbb{Z}_p$. Finally, choose y uniformly at random in $\mathbb{Z}_p$. The master key $\mathsf{msk} = (t_1, ..., t_U, y)$. The public parameters are $\mathsf{mpk} = (T_1 = g^{t_1}, ..., T_U = g^{t_U}, Y = e(g, g)^y)$.

$\texttt{Extract-Partial-Private-Key}(\mathsf{msk}, \mathcal{T})$. It takes as input msk and a policy $\mathcal{T}$ and outputs a partial secret key that enables the user to decrypt a message encrypted under a set of attributes γ if and only if $\mathcal{T}(\gamma) = 1$. The algorithm proceeds as follows. First choose a polynomial q_x for each node x (including the leaves) in the tree $\mathcal{T}$. These polynomials are chosen in the following way in a top-down manner, starting from the root node r.

For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Now, for the root node r, set $q_r(0) = y$ and d_r other points of the polynomial q_r randomly to define it completely. For any other node x, set $q_x(0) = q_{\mathrm{parent}(x)}(\mathrm{index}(x))$ and choose d_x other points randomly to completely define q_x.

Once the polynomials have been decided, for each leaf node x, we give the following secret key to the user:

$$D'_x = g^{\frac{q_x(0)}{t_i}} \text{ where } i = \mathrm{att}(x).$$

The set of above secret values is the partial private key $d_\mathcal{T}$. Observe that these computation can be done by the knowledge of g^y only.

$\texttt{Set-Secret-Value}(\mathsf{mpk}, \mathcal{T})$. It takes as input mpk and a policy $\mathcal{T}$. It randomly choose a number $a \in \mathbb{Z}_p$ and sets the a secret value $w_\mathcal{T} = a$.

$\texttt{Set-Private-Key}(\mathsf{mpk}, d_\mathcal{T}, w_\mathcal{T})$. It takes as input $\mathsf{mpk}, d_\mathcal{T}, w_\mathcal{T}$ and outputs a full private key $\mathsf{sk}_\mathcal{T} = \{D_x = (D'_x)^a\}$.

$\texttt{Set-Public-Key}(\mathsf{mpk}, w_\mathcal{T})$. It takes as input $\mathsf{mpk}, w_\mathcal{T}$, and outputs his public key $\mathsf{pk}_\mathcal{T} = (e(g, g)^{ay})$.

$\texttt{Encrypt}(\mathsf{mpk}, \mathcal{M}, \gamma, \mathsf{pk}_\mathcal{T})$. It takes as input mpk, a message $\mathcal{M}$ to encrypt, a set of attributes γ and the user's public key $\mathsf{pk}_\mathcal{T}$. It chooses a random value $s \in \mathbb{Z}_p$ and publishes the ciphertext as:

$$CT = (\gamma, \quad E' = \mathcal{M} \cdot e(g, g)^{ays}, \quad \{E_i = T_i^s\}_{i \in \gamma}).$$

$\texttt{Decrypt}(\mathsf{mpk}, CT, \mathsf{sk}_\mathcal{T})$. It takes as input a ciphertext CT for the attributes γ and a private key for a tree $\mathcal{T}$. We specify our decryption procedure as a recursive algorithm. We first define a recursive algorithm $\mathrm{DecryptNode}(CT, \mathsf{sk}_\mathcal{T}, x)$ that takes as input the ciphertext $CT = (\gamma, E', \{E_i\}_{i \in \gamma})$, the private key $\mathsf{sk}_\mathcal{T}$ (we assume the access tree $\mathcal{T}$ is embedded in the private key), and a node x in the tree. It outputs a group element of $\mathbb{G}_T$ or $\perp$.

Let $i = \text{att}(x)$. If the node x is a leaf node then:

$$\text{DecryptNode}(CT, \mathsf{sk}_{\mathcal{T}}, x) = \begin{cases} e(g^{\frac{a \cdot q_x(0)}{t_i}}, g^{s \cdot t_i}) = e(g,g)^{a \cdot s \cdot q_x(0)} \text{ if } i \in \gamma, \\ \perp \text{ otherwise.} \end{cases}$$

When x is a non-leaf node, the algorithm DecryptNode(CT, $\mathsf{sk}_{\mathcal{T}}, x$) proceeds as follows: For all nodes z that are children of x, it calls DecryptNode(CT, $\mathsf{sk}_{\mathcal{T}}, z$) and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, we return:

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}, \text{where } \begin{matrix} i = \text{index}(z) \\ S'_x = \{\text{index}(z) : z \in S_x\} \end{matrix} \\ &= \prod_{z \in S_x} (e(g,g)^{a \cdot s \cdot q_{\text{parent}(z)}(\text{index}(z))})^{\Delta_{i,S'_x}(0)} \text{ (by construction)} \\ &= e(g,g)^{a \cdot s \cdot q_x(0)} \text{(using polynomial interpolation)} \end{aligned}$$

Now that we have defined our function DecryptNode, the decryption algorithm simply calls the function on the root of the tree. We observe that DecryptNode $(CT, \mathsf{sk}_{\mathcal{T}}, r) = e(g,g)^{ays}$ if and only if the ciphertext satisfies the tree. Since, $E' = \mathcal{M} \cdot e(g,g)^{asy}$ the decryption algorithm simply divides out $e(g,g)^{asy}$ and recovers the message $\mathcal{M}$.

Theorem 3. *Suppose the decisional BDH assumption holds. Then our key-policy AS-IBE is selectively secure against IND-CPA Type I adversary.*

Proof. Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage ϵ in the selective security game against our construction. Moreover, suppose it chooses a challenge set of attributes γ^*. There are two possible ways for $\mathcal{A}$ to win:

1. Win by the public key pk^* replaced by $\mathcal{A}$. Hence $\mathcal{A}$ is not allowed to ask for the partial private key for the access structure $\mathcal{T}$ such that γ^* satisfies $\mathcal{T}$.
2. Win by the public key pk^* which was not replaced by $\mathcal{A}$.

We show how to build a simulator, $\mathcal{B}$, to handle both cases. We first show the simulation for the first case.

Init. The simulator $\mathcal{B}$ takes in a decisional BDH challenge $(\boldsymbol{y} = (g, g^a, g^b, g^c), T)$. The adversary $\mathcal{A}$ gives the challenge set of attributes γ^*.

Setup. $\mathcal{B}$ sets $Y = e(g^a, g^b)$. For $i \in [1, U]$, if $i \in \gamma^*$, it chooses random $r_i \in \mathbb{Z}_p$ and sets $T_i = g^{r_i}$; otherwise, it chooses random $\beta_i \in \mathbb{Z}_p$ and sets $T_i = g^{b\beta_i}$. $\mathcal{B}$ returns mpk to $\mathcal{A}$.

Phase 1. Consider the Create query. The simulator $\mathcal{B}$ runs `Set-Secret-Value`, `Set-Private-Key` and `Set-Public-Key` honestly according to the algorithm. $\mathcal{B}$ does not need to calculate the partial private key for the access structure $\mathcal{T}$ such that γ^* satisfies $\mathcal{T}$. It leaves a $\perp$ symbol in the corresponding entry in Υ.

$\mathcal{B}$ only needs to calculate the partial private key for the access structure $\mathcal{T}$ such that γ^* does not satisfy $\mathcal{T}$. The simulation follows the proof of Theorem 1 in [8] and we omit it due to the space limit.

Challenge. The adversary gives two messages $\mathcal{M}_0^*, \mathcal{M}_1^*$ and the challenge public key pk^* to $\mathcal{B}$. $\mathcal{B}$ flips a coin β and retrieves the corresponding $w_{\mathcal{T}}^*$ from Υ. It creates $E' = \mathcal{M}_\beta^* \cdot T^{w_{\mathcal{T}}^*}$ and $E_i = g^{cr_i}$ for all $i \in \gamma^*$.

Phase 2. Same as phase 1.

Output. $\mathcal{A}$ will eventually output a guess β' of β. $\mathcal{B}$ then outputs 0 to guess that $T = e(g,g)^{abc}$ if $\beta' = \beta$; otherwise, it outputs 1 to guess that T is a random group element in $\mathbb{G}_T$. If $\mathcal{A}$ can win this game with advantage ϵ, than $\mathcal{B}$ can solves the decisional BDH problem with probability ϵ.

We now show the simulation for the second case, where the public key pk^* was not replaced by $\mathcal{A}$.

Init. The simulator $\mathcal{B}$ takes in a decisional BDH challenge $(\boldsymbol{y} = (g, g^a, g^b, g^c), T)$. The adversary $\mathcal{A}$ gives the challenge set of attributes γ^*.

Setup. $\mathcal{B}$ sets $Y = e(g, g^b)$ which implies $g^y = g^b$. $\mathcal{B}$ generates the rest of msk and mpk accordingly. $\mathcal{B}$ sends mpk to $\mathcal{A}$.

Phase 1. Consider the Create query. On input an access tree $\mathcal{T}$, $\mathcal{B}$ can compute most keys honestly using g^b. Except for one time, $\mathcal{B}$ implicitly sets the secret value of $\mathcal{T}^*$ as a by setting $\mathsf{pk}_{\mathcal{T}^*} = e(g^a, g^b)$. $\mathcal{B}$ can still compute the partial private key $d_{\mathcal{T}}$ using g^b, but it cannot compute $\mathsf{sk}_{\mathcal{T}^*}$. It adds the tuple $(H+1, \mathcal{T}^*, \mathsf{pk}_{\mathcal{T}^*}, \perp, d_{\mathcal{T}^*}, \perp)$ to the set Υ and returns $H+1$. It updates the handle counter $H \leftarrow H+1$.

If there is an Extract Private Key query for the handle $H+1$ in the future, $\mathcal{B}$ declares failure and exits.

Challenge. The adversary gives two messages $\mathcal{M}_0^*, \mathcal{M}_1^*$ and the challenge public key pk^* to $\mathcal{B}$. If $\mathsf{pk}^* \neq \mathsf{pk}_{\mathcal{T}^*}$, $\mathcal{B}$ declares failure and exits. $\mathcal{B}$ flips a coin β. It creates $E' = \mathcal{M}_\beta^* \cdot T$ and $E_i = (g^c)^{t_i}$. It implicitly defines $s = c$.

Phase 2. Same as phase 1.

Output. $\mathcal{A}$ will eventually output a guess β' of β. $\mathcal{B}$ then outputs 0 to guess that $T = e(g,g)^{abc}$ if $\beta' = \beta$; otherwise, it outputs 1 to guess that T is a random group element in $\mathbb{G}_T$. If $\mathcal{A}$ wins this game with advantage ϵ, than $\mathcal{B}$ solves the decisional BDH problem with probability ϵ/q_C, where q_C is the number of create query asked by $\mathcal{A}$.

Therefore our scheme is selectively secure under the Type I IND-CPA adversary if the decisional BDH assumption holds. □

Theorem 4. *Suppose the decisional BDH assumption holds. Then our key-policy AS-IBE is selectively secure against IND-CPA Type II adversary.*

Proof. It is almost the same as the second case of the above proof, except that msk can be given to $\mathcal{A}$ directly in the Setup phase. □

6 Conclusion

In this paper, we propose the notion of *attribute specified identity-based encryption* for fine-grained access control of encrypted data without trusted third party. We propose two variants, namely the ciphertext policy and the key policy, which are useful in different contexts. We define formal security models and prove the security of our schemes under these models. We consider the construction of fully secure AS-IBE as an interesting open problem.

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society (2007)
3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
4. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
5. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) CCS 2009, pp. 121–130. ACM (2009)
6. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg (2007)
7. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-box accountable authority identity-based encryption. In: Ning, P., Syverson, P.F., Jha, S. (eds.) CCS 2008, pp. 427–436. ACM (2008)
8. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) CCS 2006, pp. 89–98. ACM (2006)
9. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)
10. Liu, Z., Cao, Z., Huang, Q., Wong, D.S., Yuen, T.H.: Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 278–297. Springer, Heidelberg (2011)
11. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
12. 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)
13. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
14. Yuen, T.H., Susilo, W., Mu, Y.: How to construct identity-based signatures without the key escrow problem: Formal definitions and constructions. Int. J. Inf. Secur. 9(4), 297–311 (2010)

Leakage-Resilient Attribute-Based Encryption with Fast Decryption: Models, Analysis and Constructions

Mingwu Zhang[1,3], Wei Shi[1], Chunzhi Wang[1], Zhenhua Chen[2], and Yi Mu[4]

[1] School of Computers, Hubei University of Technology
[2] School of Computers, Shaanxi Normal University
[3] College of Information, South China Agricultural University
[4] Centre for Computer and Information Security Research,
School of Computer Science and Software Engineering, University of Wollongong
csmwzhang@gmail.com, ymu@uow.edu.au

Abstract. Traditionally, in attribute-based encryption (ABE), an access structure is constructed from a linear secret sharing scheme (LSSS), a boolean formula or an access tree. In this work, we encode the access structure as its minimal sets, which is equivalent to the existence of a smallest monotonic span program for the characteristic function of the same access structure. We present two leakage-resilient attribute-based encryption schemes, ciphertext-policy ABE (LR-CP-ABE) and key-policy ABE (LR-KP-ABE), that can tolerate private key and master key to be partially leaked. By using our encoding mechanism, we obtain short ciphertext in LR-CP-ABE and short key in LR-KP-ABE. Also, our schemes have higher decryption efficiency in that the decryption cost is independent to the depth of access structures. Meanwhile, our proposed schemes provide the tolerance of both master key leakage and continual leakage in the sense that there are many master keys for universal set Σ and many private keys per attribute set $\mathbb{S}$. We explicitly employ a refresh algorithm to update a (master) key while the leakage information will beyond the allowable leakage bound. The schemes are proven to be adaptively leakage-resilient secure in the standard model under the static assumptions in composite order bilinear groups.

Keywords: Leakage resilience, Attribute-based encryption, Minimal set, Monotone access structure.

1 Introduction

In encryption systems, we could imagine encrypting a data under a policy which specifies under what conditions key-holder is allowed to decrypt the data. Attackers are modeled as probabilistic polynomial time machines with input/output access to the algorithm, and the algorithm is considered secure if it is infeasible for any such adversary to break the system. Most existing public key encryptions allow a party to encrypt data to a particular user, but are unable to efficiently handle more expressive types of encrypted access policy. In attribute-based

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 75–90, 2013.

encryption (ABE), ciphertexts and keys are associated with sets of attributes and access policies over attributes. A key holder is able to decrypt a ciphertext if and only if the attributes satisfies the associated access policy. There are two kinds of ABE systems: cipertext-policy ABE (CP-ABE), where ciphertexts are associated with access policies and keys are associated with sets of attributes, and key-policy ABE (KP-ABE), where keys are associated with access policies and ciphertexts are associated with sets of attributes.

The original ABE construction proposed by Sahai and Waters [21] was limited to specify as threshold access policies, which was limited to implement formula consisting of one threshold gate. Goyal et al. [12] subsequently improved the expressibility of access policy by allowing the key to express any monotonic access structure over attributes. To achieve a more expressive access policy over many attributes, some ABE systems make use of techniques from linear secret-sharing schemes (LSSS) or boolean formulas as access policies.

Lewko et al. [15] employed monotone span programs (MSPs) as access structure and then constructed a CP-ABE and a KP-ABE respectively that are proven to be adaptively secure in composite bilinear groups. However, the ciphertext in CP-ABE and the key in KP-ABE are polynomial in size of MSPs, and the decryptions are inefficient since the pairings of decryption are linearly to the number of rows in MSPs. In [23], Waters introduced a new technique for realizing CP-ABE under concrete and noninteractive cryptographic assumptions, which allow any encryptor to specify access control in terms of an LSSS matrix. Goyal et al. [11] presented a *bounded* CP-ABE construction, in which they showed how to transform a KP-ABE system into a CP-ABE one. In particular, they provided a mapping onto a *universal* access tree of up to depth d formulas consisting of threshold gates of input size m.

Considering the attributes in access formulae or LSSS matrices, an attribute can be used once in an access policy. Although we can obtain multi-show attribute by setting a fixed bound on the maximum times of an attribute be used, however, this is inefficient since it causes the larger scale size of public key as well as the size of key in CP-ABE. Recently, Lewko and Waters [14] proposed a new selective proof technique to support multi-show attribute and obtains an adaptive security in CP-ABE system.

Many access policies in ABE are specified as LSSS. However, there is a close relation between LSSS and MSP. Beimel [4] proved that the existence of an efficient LSSS for a specific MSP access structure is equivalent to the existence of a smallest MSP. Later, Nikova et al. [19] provided a theoretical lower bound for any MSP by using some linear algebraic machineries, where the size of a MSP is at least the size of the critical set of minimal sets for the corresponding monotone access structure plus the size of the critical set for the minimal sets of the dual of access structure minus one, i.e., the computation complexity for an access structure Γ is bounded by $|\mathcal{H}| + |\mathcal{H}^{\perp}| - 1$ where $\mathcal{H}$ and $\mathcal{H}^{\perp}$ denote the critical set of minimal sets for an access structure Γ and $\Gamma^{\perp}$ respectively. Pandit and Barua [20] used minimal sets to describe general access structure in ABE systems and constructed the corresponding (hierarchical) encryption schemes.

Recent research shows that many cryptographic schemes are vulnerable to side-channel attacks on the keys by the interaction of an adversary by measuring the timing, power-consumption, temperature, radiation, acoustics and so on [1,2,3,6,8,9,10,24,25]. The concept of leakage resilience models security of a cryptographic algorithm in the presence of an adversary who uses non-traditional way learn information about the private key. The adversary is strengthened in this model and is allowed to observe leakage from the content of private key. Leakage-resilient cryptosystems are designed to remain secure even if some information about the private key is leaked. Instead, we should take into account the key leakage in ABE system and then construct leakage-resilient ABE schemes. Also, in order to provide an efficient decryption cost, we use the minimal set to describe the monotone access structure in our leakage-resilient ABE systems.

In this work, we focus on the model of memory attacks or relative-leakage model [1]. In this model, the attacker can learn any efficiently computable function of any private key, subject only to the restriction that the total amount of information learned is bounded by predetermined parameter ℓ. Our goal is to devise ABE schemes resilient to key leakage with: (i) comparable efficiency to previously known systems, (ii) construction and security in the standard model, and (iii) better leakage rate.

2 Mathematical Backgrounds

2.1 Monotone Access Structure and Minimal Set

Definition 1. *(Access structure(AS)) Let $P_1, \cdots, P_n$ be a set of parties. A collection $\Gamma \subseteq 2^{P_1,\cdots,P_n}$ is monotonic if $\forall B \in \Gamma$ and $B \subseteq C$, then $C \in \Gamma$. An access structure is a collection Γ of non-empty subsets of $\{P_1, \cdots, P_n\}$, i.e., $\Gamma \subseteq 2^{P_1,\cdots,P_n} \backslash \{\emptyset\}$. The member in Γ is called authorized set, and the set not in Γ is called unauthorized set.*

Remark 1. In an attribute-based encryption, the attributes will play the role of parties in set $\{P_1, \cdots, P_n\}$. In the remainder of the paper, we use $\Sigma = \{a_1, a_2, \cdots, a_n\}$ to describe a finite attribute set.

Definition 2. *(Minimal set of a monotonic access structure) Let Γ be a monotonic access structure over the set of attributes $\Sigma = \{a_1, a_2, \cdots, a_n\}$. $B \in \Gamma$ is a minimal authorized set if $\forall A \in \Gamma \backslash \{B\}$, we have $A \not\subset B$. The set of all minimal sets in Γ is called the basis of Γ.*

Definition 3. *(Dual of access structure) The dual access structure $\Gamma^{\perp}$ of an access structure Γ over Σ is defined as the collection of sets $A \subset \Sigma$ such that $\Sigma \backslash A = A^c \notin \Gamma$.*

Definition 4. *(Critical set of minimal sets)[19] Let $\mathcal{B} = \{X_1, \cdots, X_r\}$ be the set of minimal set of an access structure Γ, and $\mathcal{H} \subset \mathcal{B}$ be a subset of minimal sets. $\mathcal{H}$ is called a critical set of minimal sets for $\mathcal{B}$, if every $X_i \in \mathcal{H}$ contains a set $B_i \subset X_i$, $|B_i| \geq 2$, and the following conditions hold:*

1. *The set B_i uniquely determines X_i in the set $\mathcal{H}$. i.e., no other set in $\mathcal{H}$ contains B_i;*
2. *$\forall Y \subset B_i$, set $S_Y = \cup_{X_j \in \mathcal{H}, X_j \cap Y \neq \emptyset}(X_j \backslash Y)$ does not contain any element of $\mathcal{B}$.*

Assume that $\Sigma = \{a_1, a_2, a_3, a_4\}$ is the set of attributes, and $\mathcal{B} = \{X_1 = \{a_1, a_2\}, X_2 = \{a_3, a_4\}\}$ is the set of minimal sets for a monotone access structure Γ, then $\mathcal{H}(= \mathcal{B})$ is a critical set of minimal sets. Also, $\mathcal{B}^{\perp} = \{\{a_1, a_3\}, \{a_1, a_4\}, \{a_2, a_3\}, \{a_2, a_4\}\}$. We can find a critical set $\mathcal{H}^{\perp}$ for $\Gamma^{\perp}$ to be $\{\{a_1, a_3\}, \{a_1, a_4\}\}$. As instantiating as above, we have the following theorem that was proven in [19].

Theorem 1. *Let Γ be an access structure and $\Gamma^{\perp}$ be its dual, and $\mathcal{H}$ and $\mathcal{H}^{\perp}$ be the critical set of minimal sets for Γ and $\Gamma^{\perp}$ respectively. The size of any monotone span program computing Γ is bounded by $|\mathcal{H}| + |\mathcal{H}^{\perp}| - 1$.*

Remark 2. There existence of an efficient LSSS for a specific monotonic access structure is equivalent to existence of the smallest monotonic span program for the characteristic function of the same access structure [4].

2.2 Random Subspaces for Leakage Resilience over Arbitrary Functions

We provide an algebraic tool that is crucial to our leakage resilient constructions. More specifically, we give an algebraic theorem and its claim that essentially say that random subspaces are resilient to continual leakage.

Theorem 2. *[7] Let $m, l, d \in \mathbb{N}$, $2d \leq l \leq m$ and p be a large prime. Let $X_1 \xleftarrow{\$} \mathbb{Z}_p^{m \times l}$ and $X_2 \xleftarrow{\$} \mathbb{Z}_p^{m \times d}$, and $T \xleftarrow{\$} Rank_d(\mathbb{Z}_p^{l \times d})$. For any function $f : \mathbb{Z}_p^{m \times d} \to \varphi$, there exists*

$$Dist((X_1, f(X_1T)), (X_1, f(X_2))) \leq \mathsf{negl}(\cdot), \quad |\varphi| \leq 4(1 - \frac{1}{p}) \cdot p_2^{l-2d+1} \cdot \mathsf{negl}(\cdot)^2 \tag{1}$$

We note that, if the leakage $f(X_1T)$ reveals bounded information X_1, then $(X_1, f(X_1T))$ and $(X_1, f(X_2))$ are statistically close. X_2 is a random vector and the leakage function $f(X_2)$ reveals nothing about the space X_1. By setting $d = 1$ and $l = m - 1$, we have the following claim.

Claim. Let $\Delta, \boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^m$ and $\boldsymbol{\mu}'$ be selected uniformly randomly from the set of vector in $\mathbb{Z}_p^m$ which are orthogonal to Δ under the dot product modulo p. For any function $f : \mathbb{Z}_p^m \to \{0,1\}^{\ell}$, where the function output is bounded by the length ℓ, there exists

$$Dist((\Delta, f(\boldsymbol{\mu})), (\Delta, f(\boldsymbol{\mu}'))) \leq \mathsf{negl}(\cdot), \quad \ell \leq 4p_2^{m-3}(p-1) \cdot \mathsf{negl}(\cdot)^2 \tag{2}$$

2.3 Hardness Assumptions

Bilinear groups of composite order are groups with an efficient bilinear map where the group order is a product of two or more distinct primes. Such groups are constructed from pairing friendly curves over a finite field. The following hardness assumptions are based on the static subgroup decisional problems that have been analyzed in [17,15]

Definition 5. *(1-SDP assumption) 1-class Subgroup Decision Problem (1-SDP) is hard relative to* $\Theta = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \pounds(\kappa)$ *if for all PPT algorithm* $\mathscr{A}$*, there exists a negligible function* negl *such that*

$$|Pr[\mathscr{A}(\Theta, g_1, X_3, T_1) = 1] - Pr[\mathscr{A}(\Theta, g_1, X_3, T_2) = 1]| \leq \mathsf{negl}(\kappa)$$

where the probabilities are taken over the choices of $g_1 \in \mathbb{G}_{p_1}$*,* $X_3 \in \mathbb{G}_{p_3}$*,* $T_1 \in \mathbb{G}_{p_1p_2}$ *and* $T_2 \in \mathbb{G}_{p_1}$*.*

Definition 6. *(2-SDP assumption) 2-class Subgroup Decision Problem (2-SDP) is hard relative to* $\Theta = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \pounds(\kappa)$ *if for all PPT algorithm* $\mathscr{A}$*, there exists a negligible function* negl *such that*

$$\begin{aligned}|Pr[\mathscr{A}(\Theta, g_1, X_1X_2, X_3, Y_2Y_3, T_1) = 1] \\ - Pr[\mathscr{A}(\Theta, g_1, X_1X_2, X_3, Y_2Y_3, T_2) = 1]| \leq \mathsf{negl}(\kappa)\end{aligned}$$

where the probabilities are taken over the choices of $g_1 \in \mathbb{G}_{p_1}$*,* $X_2, Y_2 \in \mathbb{G}_{p_2}$*,* $X_3, Y_3 \in \mathbb{G}_{p_3}$*,* $T_1 \in \mathbb{G}_{p_1p_2}$ *and* $T_2 \in \mathbb{G}$*.*

Definition 7. *(BSDP assumption)[17,15] Bilinear Subgroup Decision Problem (BSDP) is hard relative to* $\Theta = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \pounds(\kappa)$ *if for all PPT algorithm* $\mathscr{A}$*, there exists a negligible function* negl *such that*

$$\begin{aligned}|Pr[\mathscr{A}(\Theta, g_1, g_1^{\alpha}X_2, X_3, g_1^{s}Y_2, Z_2, T_1) = 1] \\ - Pr[\mathscr{A}(\Theta, g_1, g_1^{\alpha}X_2, X_3, g_1^{s}Y_2, Z_2, T_2) = 1]| \leq \mathsf{negl}(\kappa)\end{aligned}$$

where $T_1 = \hat{e}(g_1^{\alpha}, g_1^{s})$ *and the probabilities are taken over the choices of* $s, \alpha \in \mathbb{Z}_N$*,* $g_1 \in \mathbb{G}_{p_1}$*,* $X_2, Y_2, Z_2 \in \mathbb{G}_{p_2}$*,* $X_3 \in \mathbb{G}_{p_3}$*, and* $T_2 \in \mathbb{G}_T$*.*

3 Leakage-Resilient Attribute-Based Encryption

In this section, we give the model and security defintion of leakage-resilient ciphertext-policy ABE (LR-CP-ABE), where the key is associated with an attribute set and the ciphertext is associated with an access structure. In section 6.2, we will give the model and concrete construction of leakage-resilient key-policy ABE (LR-KP-ABE).

3.1 Model of LR-CP-ABE

Definition 8. *(LR-CP-ABE) A leakage-resilient ciphertext-policy attribute-based encryption (LR-CP-ABE) for the general access structure Γ over the attribute universe Σ is comprised of five probabilistic polynomial-time algorithms.*

1. $(\mathtt{MPK}, \mathtt{MSK}) \leftarrow \mathbf{Setup}(1^\kappa, \Sigma, \ell)$ *The system setup algorithm takes a security parameter κ, a universe of attributes Σ and an allowable private-key leakage bound ℓ as inputs, and outputs system public key $\mathtt{MPK}$ and master key $\mathtt{MSK}$. Note that the system public key can be seen by all participants in the system and will be the input in all other algorithms.*
2. $\mathtt{SK}_\mathbb{S} \leftarrow \mathbf{KeyGen}(\mathtt{MSK}, \mathbb{S})$ *The key generation algorithm takes the master key $\mathtt{MSK}$, and a set of attributes $\mathbb{S} \subseteq \Sigma$ as inputs, and outputs a private key $\mathtt{SK}_\mathbb{S}$.*
3. $\mathtt{SK}'_\mathbb{S} \leftarrow \mathbf{KeyUpd}(\mathtt{SK}_\mathbb{S}, \mathbb{S})$ *The key update algorithm takes a private key $\mathtt{SK}_\mathbb{S}$ as input and outputs a updated and re-randomized key $\mathtt{SK}'_\mathbb{S}$.*
4. $\mathtt{CT}_\Gamma \leftarrow \mathbf{Enc}(M, \Gamma)$ *The encryption algorithm takes a message M and an access structure Γ as inputs, and outputs a ciphertext $\mathtt{CT}_\Gamma$.*
5. $M \leftarrow \mathbf{Dec}(\mathtt{CT}_\Gamma, \mathtt{SK}_\mathbb{S})$ *The decryption algorithm takes a ciphertext $\mathtt{CT}_\Gamma$ and a key $\mathtt{SK}_\mathbb{S}$ as inputs, and outputs M if and only if the set of attributes $\mathbb{S}$ satisfies the access structure Γ, i.e, $\Gamma(\mathbb{S}) = 1$.*

3.2 Security Properties in the Presence of Leakage

We follow the natural leakage-resilient security definition from [1], which roughly states that an encryption is ℓ-leakage-resilient if it remains secure despite the fact that an adversary can learn up to ℓ bits of arbitrary information on the private key of being attacked.

An attribute-based encryption scheme is key-leakage resilient if it is semantically secure when the adversary obtain partial information on the key. We model the key leakage by providing the adversary a function that taking the private key as input and obtaining the output of the key. In order to record the queried and leaked keys, we set two initially empty lists: $\mathcal{R} = \langle hd, \mathbb{S} \rangle$, $\mathcal{Q} = \langle hd, \mathbb{S}, \mathtt{SK}_\mathbb{S}, lb \rangle$ to store the records, where all records are associated with a handle hd.

Definition 9. *(Leakage-resilient experiment) The leakage-resilient experiment $\mathsf{Game}_R(1^\kappa, \Sigma, \ell)$ works between a challenger $\mathscr{C}$ and an adversary $\mathscr{A}$ as follows.*

Setup. *The challenger $\mathscr{C}$ runs setup algorithm to generate public key $\mathtt{MPK}$ and master key $\mathtt{MSK}$, and starts the interaction with $\mathscr{A}$ by providing the public key $\mathtt{MPK}$.*

Lunch Query. *In this stage, adversary $\mathscr{A}$ can perform the following queries:*

- *Key extraction query (Ω_E): $\mathscr{A}$ provides an attribute set $\mathbb{S}$ to request a key $\mathtt{SK}_\mathbb{S}$, and $\mathscr{C}$ answers with $\mathtt{SK}_\mathbb{S} \leftarrow \mathsf{KeyGen}(\mathtt{MSK}, \mathbb{S})$, and adds $(hd, \mathbb{S}, \mathtt{SK}_\mathbb{S}, 0)$ into queue $\mathcal{Q}$.Notice that in this query, the leaked bit of extracted key $\mathtt{SK}_\mathbb{S}$ is 0, which means that a new created key has no leakage.*

- *Key leakage query (Ω_L):* $\mathscr{A}$ *issues a key leakage query for* $\mathsf{SK}_{\mathbb{S}}$ *with a function* $f : \mathsf{SK} \to \{0,1\}^*$. $\mathscr{C}$ *at first seeks the record in* $\mathcal{Q}$, *and responds with* $f(\mathsf{SK}_{\mathbb{S}})$ *if* $lb + f(\mathsf{SK}_{\mathbb{S}}) \leq \ell$, *and updates* lb *with* $lb + f(\mathsf{SK}_{\mathbb{S}})$; *Outputs* ϕ *otherwise.*
- *Key update query (Ω_U):* $\mathscr{A}$ *issues a key update query for* $\mathsf{SK}_{\mathbb{S}}$. $\mathscr{C}$ *finds the record in* $\mathcal{Q}$. *If not found,* $\mathscr{C}$ *returns the key with key extraction oracle* Ω_E *and sets* $lb = 0$. *Otherwise,* $\mathscr{C}$ *returns with* $\mathsf{SK}'_{\mathbb{S}} \leftarrow \mathsf{KeyUpd}(\mathsf{SK}_{\mathbb{S}}, \mathbb{S})$ *and updates the corresponding* lb *with 0.*

Challenge. $\mathscr{A}$ *outputs two messages* $(M^{(0)}, M^{(1)})$ *and an access structure* Γ *such that for all* $\mathbb{S} \in \mathcal{R}$ $\Gamma(\mathbb{S}) = 0$. $\mathscr{C}$ *at random picks a bit* $b \in \{0,1\}$ *and then returns the challenge ciphertext* $\mathsf{CT}^{(b)} = \mathsf{Enc}(M^{(b)}, \Gamma)$.

Supper Query. $\mathscr{A}$ *continues to issues the queries like in Lunch query.*

Response. *Finally,* $\mathscr{A}$ *outputs a bit* $b' \in \{0,1\}$ *as the guess for the random coin* b *in the challenge phase. Adversary* $\mathscr{A}$*'s advantage in experiment* $\mathsf{Game}_R(1^\kappa, \Sigma, \ell)$ *is defined as* $\mathsf{Adv}_{\mathscr{A}}(1^\kappa, \Sigma, \ell) = |2Pr[(b = b')] - 1|$.

Definition 10. *(Adaptively leakage-resilient security) Suppose that the leakage bound is* ℓ *and a polynomial-time adversary has at most* Q *queries for keys. An attribute-based encryption scheme is adaptively* $(Q, \ell, \frac{\ell}{|\mathsf{SK}|})$*-leakage-resilient secure if the advantage of the adversary in winning* $\mathsf{Game}_R(\kappa, \Sigma, \ell)$ *is less than* $\mathsf{negl}(\kappa)$ *in security parameter* κ *and leakage bound* ℓ.

Definition 11. *(Selectively leakage-resilient security) An attribute-based encryption scheme is selectively* $(Q, \ell, \frac{\ell}{|\mathsf{SK}|})$*-leakage-resilient secure, if in experiment* $\mathsf{Game}_R(\kappa, \Sigma, \ell)$ *the challenge pair had to provide before the system public key and master key build, and the advantage of the adversary in the experiment is less than* $\mathsf{negl}(\kappa)$ *in security parameter* κ *and leakage bound* ℓ.

Definition 12. *(Leakage rate) The leakage rate* $\gamma = \ell/|\mathsf{SK}|$ *is defined as the relative leakage of a private key* SK, *where* ℓ *is an allowable leakage bound and* $|\mathsf{SK}|$ *is the number of bits needed to efficiently store private key* SK.

4 Construction of LR-CP-ABE

Let Σ be an attribute set, we denote the cardinality of set Σ by $|\Sigma|$. Let vectors $\boldsymbol{\rho} = (\rho_1, \rho_2, \cdots, \rho_n)$ and $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_n)$, we denote the inner product of vectors $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$ by $\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle$ and the bilinear group inner product by $\hat{e}_n(g^{\boldsymbol{\rho}}, g^{\boldsymbol{\sigma}})$. i.e.,

$$\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle = \sum_{i \in [n]} \rho_i \sigma_i$$

and

$$\hat{e}_n(g^{\boldsymbol{\rho}}, g^{\boldsymbol{\sigma}}) = \prod_{i \in [n]} \hat{e}(g^{\rho_i}, g^{\sigma_i}) = \hat{e}(g, g)^{\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle}.$$

LR-CP-ABE.Setup$(1^\kappa, \Sigma, \ell)$ At first PKG runs the bilinear group generator to produce $\Theta = (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, \hat{e})$, where p_1, p_2 and p_3 are distinct primes

and defines $\mathsf{negl} = p_2^{-\tau}$ as the allowable maximum probability in succeeding in leakage guess[1], and then computes[2]

$$\omega = \lceil 1 + 2\tau + \frac{\ell}{\log_2 p_2} \rceil$$

Also, PKG selects random generators $g_1 \in \mathbb{G}_{p_1}$ and $g_3 \in G_{p_3}$, and for each attribute $i \in \Sigma$, at random picks $t_i \in \mathbb{Z}_N$ and sets $T_i = g_1^{a_i t_i}$. It chooses $\alpha \in \mathbb{Z}_N, a, t, y_2, y_3 \in \mathbb{Z}_N$ and sets $Y = \hat{e}(g_1, g_1)^\alpha$. For $i = 1, \cdots, \omega$, it selects $\rho_i, y_{1,i} \in \mathbb{Z}_N$, and for $j = 1, \cdots, |\Sigma|$, selects $y_{4,j} \in \mathbb{Z}_N$, and then sets the master key as

$$\mathtt{MSK} = \langle \Sigma, \boldsymbol{w}_1, w_2, w_3, \boldsymbol{w}_4 \rangle = \langle \Sigma, g_1^{\boldsymbol{\sigma}} g_3^{\boldsymbol{y}_1}, g_1^{\alpha + at + \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle} g_3^{y_2}, g_1^t g_3^{y_3}, \forall i \in \Sigma \; T_i^t g_3^{y_{4,i}} \rangle \tag{3}$$

Finally, PKG publishes the system public key

$$\mathtt{MPK} = \langle \Theta, g_1, g_3, g_1^a, g_1^{\boldsymbol{\rho}}, Y, (T_i)_{i \in \Sigma} \rangle \tag{4}$$

LR-CP-ABE.KeyGen$(\mathtt{MSK}, \mathbb{S})$ On input an attribute set $\mathbb{S}$ and the master key $\mathtt{MSK} = \langle \Sigma, \boldsymbol{w}_1, w_2, w_3, \boldsymbol{w}_4 \rangle$, this algorithm selects $\triangle t, \triangle y_2, \triangle y_3 \in \mathbb{Z}_N$, and selects $y_{1,i} \in \mathbb{Z}_N$ for $i \in [\omega]$, and picks $y_{4,j} \in \mathbb{Z}_N$ for $j \in [|\mathbb{S}|]$ randomly, and returns the key $\mathtt{SK}_{\mathbb{S}}$ as:

$$\begin{aligned} \mathtt{SK}_{\mathbb{S}} &= \langle \mathbb{S}, \boldsymbol{k}_1, k_2, k_3, \boldsymbol{k}_4 \rangle \\ &= \begin{pmatrix} \mathbb{S}, \\ \boldsymbol{w}_1 * g_1^{\triangle \boldsymbol{\sigma}} * g_3^{\triangle \boldsymbol{y}_1}, \\ w_2 * g_1^{a \triangle t + \langle \boldsymbol{\rho}, \triangle \boldsymbol{\sigma} \rangle} * g_3^{\triangle y_2}, \\ w_3 * g_1^{\triangle t} * g_3^{\triangle y_3}, \\ \forall i \in \mathbb{S} \quad w_{4,i} * T_i^{\triangle t} * g_3^{\triangle y_{4,i}} \end{pmatrix}^\top = \begin{pmatrix} \mathbb{S}, \\ g_1^{\boldsymbol{\sigma} + \triangle \boldsymbol{\sigma}} g_3^{\boldsymbol{y}_1 + \triangle \boldsymbol{y}_1}, \\ g_1^{\alpha + a(t + \triangle t) + \langle \boldsymbol{\rho}, \boldsymbol{\sigma} + \triangle \boldsymbol{\sigma} \rangle} g_3^{y_2 + \triangle \boldsymbol{y}_2}, \\ g_1^{t + \triangle t} g_3^{y_3 + \triangle y_3}, \\ \forall i \in \mathbb{S} \quad T_i^{t + \triangle t} g_3^{y_{4,i} + \triangle y_{4,i}} \end{pmatrix}^\top \end{aligned} \tag{5}$$

Note that the components of private key are $\omega + |\mathbb{S}| + 2$ elements in subgroup $\mathbb{G}_{p_1 p_3}$.

LR-CP-ABE.KeyUpd$(\mathtt{SK}_{\mathbb{S}}, \mathbb{S})$ Let a private key $\mathtt{SK}_{\mathbb{S}} = \langle \mathbb{S}, \boldsymbol{k}_1, \; k_2, k_3, \boldsymbol{k}_4 \rangle = \langle \mathbb{S}, g_1^{\boldsymbol{\sigma}} g_3^{\boldsymbol{y}_1}, \; g_1^{\alpha + at + \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle} g_3^{y_2}, \; g_1^t g_3^{y_3}, (T_i^t g_3^{y_{4,i}})_{i \in \mathbb{S}} \rangle$. The key update algorithm at random selects $\triangle t, \triangle y_2, \triangle y_3 \in \mathbb{Z}_N$, and selects $y_{1,i} \in \mathbb{Z}_N$ for $i \in [\omega]$, and $y_{4,j} \in \mathbb{Z}_N$ for $j \in [|\mathbb{S}|]$, and outputs a new key $\mathtt{SK}'_{\mathbb{S}}$:

$$\begin{aligned} \mathtt{SK}'_{\mathbb{S}} &= \langle \mathbb{S}, \boldsymbol{k}'_1, k'_2, k'_3, \boldsymbol{k}'_4 \rangle \\ &= \begin{pmatrix} \mathbb{S}, \\ \boldsymbol{k}_1 * g_1^{\triangle \boldsymbol{\sigma}} * g_3^{\triangle \boldsymbol{y}_1}, \\ k_2 * g_1^{a \triangle t + \langle \boldsymbol{\rho}, \triangle \boldsymbol{\sigma} \rangle} * g_3^{\triangle y_2}, \\ k_3 * g_1^{\triangle t} * g_3^{\triangle y_3}, \\ \forall i \in \mathbb{S} \quad k_{4,i} * T_i^{\triangle t} * g_3^{\triangle y_{4,i}} \end{pmatrix}^\top = \begin{pmatrix} \mathbb{S}, \\ g_1^{\boldsymbol{\sigma}'} g_3^{\boldsymbol{y}'_1}, \\ g_1^{\alpha + at' + \langle \boldsymbol{\rho}, \boldsymbol{\sigma}' \rangle} g_3^{y'_2}, \\ g_1^{t'} g_3^{y_3}, \\ (T_i^{t'} g_3^{y_{4,i}})_{i \in \mathbb{S}} \end{pmatrix}^\top \end{aligned} \tag{6}$$

[1] We can denote this probability as the entropy loss when the private key leaks.

[2] As τ is a small positive constant, in practice we can eliminate this parameter in ω, i.e., $\omega \approx \lceil 1 + \frac{\ell}{\log_2 p_2} \rceil$.

where $t' = t + \triangle t$, $\boldsymbol{\sigma}' = \boldsymbol{\sigma} + \triangle \boldsymbol{\sigma}$, $\boldsymbol{y}_1' = y_1 + \triangle \boldsymbol{y}_1$, $y_2' = y_2 + \triangle y_2$, $y_3' = y_3 + \triangle y_3$, and $\boldsymbol{y}_4 = \boldsymbol{y}_4 + \triangle y_4$.

Remark 3. If $\mathbb{S} = \Sigma$ and $\mathtt{SK}_\Sigma = \mathtt{MSK}$, then the update algorithm can refresh the master key that generates a same distributed master key. Thus, we can consider the master key $\mathtt{MSK}$ as a special private key for universal set Σ.

Remark 4. In our scheme, we allow many private keys per attribute set $\mathbb{S}$ and many master keys for universal attribute set Σ.

LR-CP-ABE.Enc(M, Γ) At first, this algorithm converts the monotone access structure Γ to the set of minimal sets $\mathcal{B} = \{B_1, B_2, \cdots, B_m\}$, where $B_i \subset \Sigma$ for $i = 1, \cdots, m$. The algorithm also at random selects $s, s_1, \cdots, s_m \in \mathbb{Z}_N$, and outputs the ciphertext $\mathtt{CT}_\Gamma$:

$$\mathtt{CT}_\Gamma = \langle \mathcal{B}, c_0, \boldsymbol{c}_1, c_2, \boldsymbol{c}_3, \boldsymbol{c}_4 \rangle = \langle \mathcal{B}, MY^s, g_1^{s\boldsymbol{\rho}}, g_1^{-s}, g_1^{as}(\prod_{j \in B_i} T_j)_{i \in [m]}^{s_i}, (g_1^{s_i})_{i \in [m]} \rangle \tag{7}$$

LR-CP-ABE.Dec$(\mathtt{CT}_\Gamma, \mathtt{SK}_\mathbb{S})$ If attributes set $\mathbb{S}$ satisfies the access structure Γ specified by $\mathcal{B}$, then $\mathbb{S}$ must be a superset of a minimal set in $\mathcal{B}$. Let $S_k \subset \mathbb{S}$ for some $k \in [m]$. This algorithm calculates:

$$M \leftarrow c_0 \frac{\hat{e}_\omega(c_1, k_1)\hat{e}(c_2, k_2)\hat{e}(c_{3,k}, k_3)}{\hat{e}(c_{4,k}, \prod_{i \in B_k} k_{4,i})} \tag{8}$$

Remark 5. In our scheme, we are equipped with an update algorithm KeyUpd that takes in a (master) private key and outputs a new and re-randomized key from the same distribution generated by a fresh call to KeyGen algorithm, then the security will yield resilience to continual leakage "for free". In particular, the many master keys for universal set Σ and the many private keys per attribute set $\mathbb{S}$ allow to leak can be interpreted as refreshed versions of corresponding master/private keys.

Correctness. The correctness is described as follows:

(1) $\hat{e}(c_1, k_1) = \hat{e}(g_1^{s\boldsymbol{\rho}}, g_1^{\boldsymbol{\sigma}} g_3^{y_1}) = \hat{e}(g_1^{s\boldsymbol{\rho}}, g_1^{\boldsymbol{\sigma}}) = \hat{e}(g_1, g_1)^{s\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle}$

(2) $\hat{e}(c_2, k_2) = \hat{e}(g_1^{-s}, g_1^{\alpha + at + \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle} g_3^{y_2}) = \hat{e}(g_1, g_1)^{-s\alpha - ast - s\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle}$

(3) $\hat{e}(c_{3,k}, k_3) = \hat{e}(g_1^{as}(\prod_{j \in B_k} T_j)^{s_k}, g_1^t g_3^{y_3}) = \hat{e}(g_1, g_1)^{ast} \hat{e}(\prod_{j \in B_k} T_j, g_1)^{s_k t}$

(4) $\hat{e}(c_{4,k}, \prod_{j \in B_k} k_{4,j}) = \hat{e}(g_1^{s_k}, \prod_{j \in B_k} T_j^t g_3^{y_{4,j}}) = \hat{e}(g_1, \prod_{j \in B_k} T_j)^{s_k t}$

Then, the blind factor is calculated by

$$\frac{\hat{e}(c_1, k_1)\hat{e}(c_2, k_2)\hat{e}(c_{3,k}, k_3)}{\hat{e}(c_{4,k}, \prod_{j \in B_k} k_{4,j})} = \hat{e}(g_1, g_1)^{-s\alpha} = Y^{-s} \tag{9}$$

Remark 6. In the decryption, the algorithm only performs $\omega+3$ pairing operations, which is more efficient than the construction that uses LSSS to specify the access structure in [16]. We will discuss and compare the decryption performance in section 6.2.

5 Security

We will prove the adaptive security of LR-CP-ABE with the technique of dual system encryption. Our analysis of leakage resilience of our system will rely on Theorem 2 in [7], which is proven using the techniques in [5].

Our security employs the dual system encryption mechanism of [22,17]. Let Q be the number of key queries that the adversary makes, then our proof considers a sequence of $2Q+4$ games between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. By means of dual system encryption, we at first give the semi-functional ciphertext/key generation algorithms and convert the challenge ciphertext and queried keys into semi-functional form. We also define two types of semi-functional key. The semi-functional key and ciphertext algorithms are presented as follows:

- **KeyGenSF**. Let $\mathtt{SK}_{\mathbb{S}} = \langle \mathbb{S}, \boldsymbol{k}_1, k_2, k_3, \boldsymbol{k}_4 \rangle$ be a normal key, a semi-functional key is constructed as:
  1. Type 1: $\overline{\mathtt{SK}}_{\mathbb{S}} = \langle \mathbb{S}, \boldsymbol{k}_1 * g_2^{\boldsymbol{d}_1}, k_2 * g_2^{d_2}, k_3 * g_2^{d_3}, \boldsymbol{k}_4 * g_2^{\boldsymbol{d}_4} \rangle$, where g_2 is a random generator of $\mathbb{G}_{p_2}$ and $d_i (i = 1, \ldots, 4)$ is randomly picked from $\mathbb{Z}_N$.
  2. Type 2: $\overline{\mathtt{SK}}_{\mathbb{S}} = \langle \mathbb{S}, \boldsymbol{k}_1, k_2 * g_2^{d_2}, k_3, \boldsymbol{k}_4 \rangle$.
- **EncSF**. Let $\mathtt{CT}_{\Gamma} = \langle \mathcal{B}, c_0, \boldsymbol{c}_1, c_2, \boldsymbol{c}_3, \boldsymbol{c}_4 \rangle$ be a normal ciphertext, a semi-functional ciphertext is converted as: $\overline{\mathtt{CT}}_{\Gamma} = \langle \mathcal{B}, c_0, \boldsymbol{c}_1 * g_2^{\boldsymbol{e}_1}, c_2 * g_2^{e_2}, \boldsymbol{c}_3 * g_2^{\boldsymbol{e}_3}, \boldsymbol{c}_4 \rangle$, where $\boldsymbol{e}_1, e_2, \boldsymbol{e}_3$ are random elements in $\mathbb{Z}_N$.

Obviously, if we use a type-1 semi-functional key to decrypt a semi-functional ciphertext, we will obtain extra term $\hat{e}(g_2, g_2)^{\langle \boldsymbol{d}_1, \boldsymbol{e}_1 \rangle + d_2 e_2 + d_3 e_{3,k}}$. If $\langle \boldsymbol{d}_1, \boldsymbol{e}_1 \rangle + d_2 e_2 + d_3 e_{3,k} = 0$, we call the semi-functional key is a nominally semi-functional key w.r.t the ciphertext, otherwise we call the semi-functional key is truly semi-functional.

Our security proof has two steps: At first we use a series of indistinguishable games to prove that the scheme is adaptively secure in non-match key/ciphertext situation, which is derived from the idea of dual system encryption [22,17,14]. We do so by proving that, in the view of the adversary, the valid private keys are indistinguishable from keys that are random in the subgroup in which the message is embedded. Secondly, we prove that, even the adversary has at most ℓ bits leakage on each match key, he also has only negligible advantage to decrypt the challenge ciphertext. We give the following theorem:

Theorem 3. *If a dual system*

$$\prod_{D} = (\mathit{Setup}, \mathit{KeyGen}, \mathit{KeyUpd}, \mathit{Enc}, \mathit{Dec}, \mathit{KeyGenSF}, \mathit{EncSF})$$

has semi-functional ciphertext invariance, semi-functional key invariance, and semi-functional security under the leakage bound ℓ, then the LR-CP-ABE scheme $\prod = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyUpd}, \mathsf{Enc}, \mathsf{Dec})$ *is an* $(Q, \ell, \frac{\ell}{|SK|})$*-leakage secure attribute-based encryption scheme.*

Proof. We prove this theorem by a series of claims listed in Tab.1. The key and the ciphertext in real construction in Section 4 are normal forms. At first we show that the update procedure can be considered as a special key extraction procedure, and then we only consider key extraction oracle instead of update oracle. Next we convert the challenge ciphertext into semi-functional form, and then convert the keys into semi-functional forms one by one. By these conversions, all ciphertexts and keys are semi-functional. We also give a Claim 5 to demonstrate that an adversary has no advantage in changing a truly semi-functional key (can not decrypt a semi-functional ciphertext) to a nominally semi-functional key (can decrypt a semi-functional ciphertext). Finally, we also show that the message is indistinguishable from a random message in the challenge ciphertext, which means that the challenge message is fully hidden in the ciphertext. We provide the security by Claim 1 – Claim 6 and hybrid argument over the sequence of games to demonstrate the real security game Game_R is computationally indistinguishable from Game_4, in which the challenge message $M^{(b)}$ is masked with a random element in $\mathbb{G}_T$. We leave the detail proof in the full version.

Table 1. Claims from indistinguishable games

Claim	Result	Functionality
Claim 1	Ω_U can be answered by Ω_E	Update invariance
Claim 2	$\mathsf{Adv}_{\mathscr{A}}^{Game_R} - \mathsf{Adv}_{\mathscr{A}}^{Game_1} \leq \epsilon_1$	
Claim 3	$\mathsf{Adv}_{\mathscr{A}}^{Game_2} - \mathsf{Adv}_{\mathscr{A}}^{Game_1} \leq \epsilon_2$	Semi-functional ciphertext invariance
Claim 4	$\mathsf{Adv}_{\mathscr{A}}^{Game_{3,k+1}} - \mathsf{Adv}_{\mathscr{A}}^{Game_{3,k}} \leq \epsilon_{3,k}$	Semi-functional key invariance
Claim 5	$\mathsf{Adv}_{\mathscr{A}}^{Game_{3,k}^{L}} - \mathsf{Adv}_{\mathscr{A}}^{Game_{3,k}} \leq \epsilon_{3,k}^{L}$	Truly/nominally semi-functional inconvertibility
Claim 6	$\mathsf{Adv}_{\mathscr{A}}^{Game_4} - \mathsf{Adv}_{\mathscr{A}}^{Game_{3,Q}} \leq \epsilon_4$	Message hiding

6 Performance and Discussion

6.1 Master Key Leakage Tolerance

In our construction, we design the same key structure of master key $\mathtt{MSK}$ and user private key $\mathtt{SK}_{\mathbb{S}}$. Actually, the master key $\mathtt{MSK}$ can be considered as a special key of universal attribute set Σ. Implicitly, we can call KeyUpd algorithm to update and refresh the master key that takes the master key $\mathtt{MSK}$ and attribute set Σ as inputs. As we only re-randomize the randomness in the master key, the refreshed master key does not impact on the previous user key generated by it.

6.2 Leakage-Resilient Key-Policy ABE

In this section, we give the construction of key-policy attribute-based encryption with leakage resilience (LR-KP-ABE), which uses the same technique in section 4, i.e., using the set of minimal sets to describe the monotone access structure. In key-policy ABE, a key is associated with access structure and a ciphertext is associated with a set of attributes. The construction has the similar security proof method with LR-CP-ABE.

LR-KP-ABE.Setup$(1^\kappa, \Sigma, \ell)$ Like in section 4, this algorithm generates the description of composite-order bilinear group Θ, and selects randomness and then sets the master key and the master public key as:

$$\begin{aligned}
\mathtt{MPK} &= \langle \Theta, g_1, g_3, g_1^a, g_1^{\boldsymbol{\rho}}, Y, (T_i)_{i\in\Sigma}\rangle && (10)\\
\mathtt{MSK} &= \langle \boldsymbol{w}_1, w_2, \boldsymbol{w}_3, \boldsymbol{w}_4\rangle \\
&= \langle g_1^\sigma g_3^{\boldsymbol{y}_1}, g_1^{-t} g_3^{y_2}, (g_1^{\alpha+at+\langle\boldsymbol{\rho},\boldsymbol{\sigma}\rangle}(\prod_{j=[|\Sigma|]} T_j)^{t_i} g_3^{y_{3,i}})_{i\in[|\Sigma|]}, (g_1^{t_i} g_3^{y_{4,i}})_{i\in[|\Sigma|]}\rangle && (11)
\end{aligned}$$

LR-KP-ABE.KeyGen$(\mathtt{MSK}, \Gamma)$ Let $\mathtt{MSK} = \langle \boldsymbol{w}_1, w_2, \boldsymbol{w}_3, \boldsymbol{w}_4\rangle$. This algorithm first converts the monotone access structure Γ to the set of minimal sets $\mathcal{B} = \{B_1, B_2, \cdots, B_m\}$, where $B_i \subset \Sigma$ for $i = 1, \cdots, m$, and then generates the private key $\mathtt{SK}_\Gamma$ as

$$\begin{aligned}
\mathtt{SK}_\Gamma &= \langle \mathcal{B}, \boldsymbol{k}_1, k_2, \boldsymbol{k}_3, \boldsymbol{k}_4\rangle \\
&= \langle \mathcal{B}, \boldsymbol{w}_1 * g_1^{\triangle\sigma} * g_3^{\triangle \boldsymbol{y}_1}, w_2 * g_1^{-\triangle t} * g_3^{\triangle y_2}, \\
&\quad (w_{3,i} * g_1^{a\triangle t + \langle\boldsymbol{\rho}, \triangle\boldsymbol{\sigma}\rangle}(\prod_{j\in B_i} T_j)^{\triangle t_i} * g_3^{\triangle y_{3,i}})_{i\in[m]}, \\
&\quad (w_{4,i} * g_1^{\triangle t_i} * g_3^{\triangle y_{4,i}})_{i\in[m]}\rangle \\
&= \begin{pmatrix} \mathcal{B}, g_1^{\sigma+\triangle\sigma} g_3^{\boldsymbol{y}_1+\triangle\boldsymbol{y}_1}, \\ g_1^{-t-\triangle t} * g_3^{y_2+\triangle y_2}, \\ (g_1^{\alpha+a(t+\triangle t)+\langle\boldsymbol{\rho},\boldsymbol{\sigma}+\triangle\boldsymbol{\sigma}\rangle}(\prod_{j\in B_i} T_j)^{t_i+\triangle t_i} * g_3^{y_{3,i}+\triangle y_{3,i}})_{i\in[m]} \\ (g_1^{t_i+\triangle t_i} g_3^{y_{4,i}+\triangle t_{4,i}})_{i\in[m]} \end{pmatrix}^\top && (12)
\end{aligned}$$

where $\triangle t, \triangle\boldsymbol{\sigma}, \triangle t_i, \triangle\boldsymbol{y}_1, \cdots, \triangle\boldsymbol{y}_4$ are picked from $\mathbb{Z}_N$ randomly.

LR-KP-ABE.KeyUpd$(\mathtt{SK}_\Gamma, \Gamma)$ This algorithm at random selects $\triangle\boldsymbol{\sigma}, \triangle t$, $\triangle y_1, \triangle y_2$, $\triangle y_3, \triangle y_4$ from Z_N, and performs the refresh procedure like in LR-KP-ABE.KeyGen$(\mathtt{MSK}, \Gamma)$.

LR-KP-ABE.Enc$(M, \mathbb{S})$ Output the ciphertext for attribute set $\mathbb{S}$ as

$$\mathtt{CT}_\mathbb{S} = \langle \mathbb{S}, c_0, \boldsymbol{c}_1, c_2, c_3, \boldsymbol{c}_4\rangle = \langle \mathbb{S},\ MY^s,\ g_1^{s\boldsymbol{\rho}},\ g_1^{as},\ g_1^s,\ \forall i \in \mathbb{S}\ \ T_i^s\rangle \qquad (13)$$

where s is picked from $\mathbb{Z}_N$ randomly.

LR-KP-ABE.Dec$(\mathtt{CT}_\mathbb{S}, \mathtt{SK}_\Gamma)$ If $\mathbb{S}$ satisfies Γ specified by $\mathcal{B}$, then $\mathbb{S}$ must be a superset of a minimal set in $\mathcal{B}$. Find $S_k \subset \mathbb{S}$ for some $k \in [m]$, and calculate:

$$M \leftarrow c_0 \frac{\hat{e}_\omega(c_1, k_1)\hat{e}(c_{4,k}, \prod_{i\in B_k} k_{4,i})}{\hat{e}(c_2, k_2)\hat{e}(c_{3,k}, k_3)} \qquad (14)$$

Table 2. Performance

schemes	LRW11[16]	LR-CP-ABE	LR-KP-ABE
Encrypt	$2(\omega+2n_1)Mu$	$(\omega+2m)Mu$	$(\omega+\lvert\mathbb{S}\rvert+2)Mu$
Decrypt	$(\omega+2n_1+1)Pr+1Ex$	$(\omega+3)Pr$	$(\omega+3)Pr$
KeyUpdate	$2(\omega+\lvert\mathbb{S}\rvert+2)Mu$	$2(\omega+\lvert\mathbb{S}\rvert+2)Mu$	$2(\omega+2m+1)Mu$
# of MSK	$(\omega+\lvert\Sigma\rvert+2)\lvert\mathbb{G}\rvert$	$(\omega+\lvert\Sigma\rvert+2)\lvert\mathbb{G}\rvert$	$(\omega+2\lvert\Sigma\rvert+1)\lvert\mathbb{G}\rvert$
# of $\mathtt{SK}_{\mathbb{S}}$	$(\omega+\lvert\mathbb{S}\rvert+2)\lvert\mathbb{G}\rvert$	$(\omega+\lvert\mathbb{S}\rvert+2)\lvert\mathbb{G}\rvert$	$(\omega+2m+1)\lvert\mathbb{G}\rvert$
# of $\mathtt{CT}_{\Gamma}$	$(\omega+2n_1+1)\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert$	$(\omega+2m+1)\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert$	$(\omega+\lvert\mathbb{S}\rvert+2)\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert$
Master key leakage	$\surd$	$\surd$	$\surd$
User key leakage	$\surd$	$\surd$	$\surd$
Continual leakage	$\surd$	$\surd$	$\surd$
Multi-show attr	X	$\surd$	$\surd$
Leakage bound ℓ	$2+(\omega-1-2\tau)\log p_2$	$2+(\omega-1-2\tau)\log p_2$	$2+(\omega-1-2\tau)\log p_2$
Allowable probability	$p_2^{-\tau}$	$p_2^{-\tau}$	$p_2^{-\tau}$
Leakage rate γ	$\frac{\omega-1-2\tau}{(1+\beta_1+\beta_3)(\omega+2+\lvert\mathbb{S}\rvert)}$	$\frac{\omega-1-2\tau}{(1+\beta_1+\beta_3)(\omega+2+\lvert\mathbb{S}\rvert)}$	$\frac{\omega-1-2\tau}{(1+\beta_1+\beta_3)(\omega+2m+1)}$

ω: leakage parameter; τ: allowable leakage probability parameter; ℓ: leakage bound of a key; γ: leakage rate, i.e., $\gamma=\ell/|\mathtt{SK}|$; Pr: computation cost of pairing; Ex: exponent cost in $\mathbb{G}_T$; Mu: point multiplication; $|\mathbb{G}|$: size of an element in $\mathbb{G}$; $|\mathbb{G}_T|$: size of an element in $\mathbb{G}_T$; Σ: universal attribute set; $\mathbb{S}$: attribute set; $\mathcal{I}$: minimum #rows labeled by user's attributes to compute target vector in LSSS matrix with n_1 rows and n_2 columns; β_1,β_3: value of $|\mathbb{G}_{p_1}|/|\mathbb{G}|$ and $|\mathbb{G}_{p_3}/|\mathbb{G}|$;

6.3 Performance

In this section, we give the performance analysis and comparison between [16] and ours schemes, which are listed in Tab. 2.

[16] and our LR-CP-ABE are all ciphertext-policy attribute-based encryption schemes in the presence of key leakage model. [16] uses LSSS to denote the access structure, but it does not support attribute multi-show functionality[14]. Our schemes use the minimal set to denote the access structure and support attribute multi-show ability.

We evaluate the computation cost in decryption since it mainly depends on the bilinear pairing operation in this algorithm but the pairing operation is very time-consuming compared to the other operations such as point multiplication, exponent and so on. To express an access structure, row number n_1 in LSSS has the approximate size with the number of set m in minimal set method. However, as far as the decryption in our two schemes, they need constant $\omega+3$ pairing operation which is independent to the scale of access structure Γ and are more efficient than [16] that describes the access structure as LSSS.

On the side of leakage resilience, all schemes support master key leakage, user private key leakage and continual leakage. Also, the schemes have the same leakage bound $\ell=2+(\omega-1-2\tau)\log p_2$ and allowable probability $p=p_2^{-\tau}$. Thus, the leakage rate of our LR-CP-ABE and [16] are

$$\gamma=\frac{\omega-1-2\tau}{(1+\beta_1+\beta_3)(\omega+2+|\mathbb{S}|)} \tag{15}$$

The leakage rate of LR-KP-ABE is

$$\frac{\omega - 1 - 2\tau}{(1 + \beta_1 + \beta_3)(\omega + 2m + 1)} \tag{16}$$

Obviously, higher values of ω give a better leakage rate, but leads to larger public parameters, private keys, and ciphertexts. Smaller values of β_1 and β_3 provide a better leakage rate, but also give fewer bits of security in subgroup $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_3}$. We must choose the security parameter κ so that $\beta_1\kappa$ and $\beta_3\kappa$ are sufficiently large.

In the setup algorithm, we set

$$\omega = \lceil 1 + 2\tau + \frac{\ell}{\log p_2} \rceil \tag{17}$$

In particular, if $\omega = 1$ then $\ell = 0$ and $\tau = 0$. In this case, the scheme is simplified to be a fully secure non-leakage attribute-based encryption like in [14], which is straightforward to see that allowable leakage is zero.

7 Conclusions

We proposed two leakage-resilient attribute-based encryptions that can tolerate leakage on the master key, as well as leakage on several keys for each attribute set. We explicitly employ a update algorithm to periodically update the master/private key so that it tolerates continual (master) key leakage. In our schemes, the access structures are converted as the minimal set, which can provide fast decryption ability. We can give our construction in prime order groups by using the transformation mechanism from [13].

Grants. This work is supported by the National Natural Science Foundation of China under Grants (61272404, 61170135, 61103232), and the Guangdong Natural Science Foundation under Grant S2012010010383.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
4. Beimel, A., Gal, A., Paterson, M.: Lower bounds for monotone span programs. Computational Complexity 6(1), 29–45 (1997)

5. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
6. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-Key encryption under subgroup indistinguishability. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
7. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Publickey cryptography resilient to continual memory leakage. In: FOCS 2010, pp. 501–510 (2010)
8. Chow, S., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions. In: ACM-CCS 2010, pp. 152–161 (2010)
9. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010)
10. Dodis, Y., Lewko, A., Waters, B., Wichs, D.: Storing secrets on continually leaky devices. In: FOCS 2011, pp. 688–697 (2011)
11. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM-CCS 2006, pp. 89–98 (2006)
13. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
14. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)
15. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
16. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 70–88. Springer, Heidelberg (2011)
17. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
18. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
19. Nikov, V., Nikova, S., Preneel, B.: On the size of monotone span programs. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 249–262. Springer, Heidelberg (2005)
20. Pandit, T., Barua, R.: Efficient fully secure attribute-based encryption schemes for general access structures. In: Takagi, T., Wang, G., Qin, Z., Jiang, S., Yu, Y. (eds.) ProvSec 2012. LNCS, vol. 7496, pp. 193–214. Springer, Heidelberg (2012)
21. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

22. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
23. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
24. Yang, B., Zhang, M.: LR-UESDE: A continual-leakage resilient encryption with unbounded extensible set delegation. In: Takagi, T., Wang, G., Qin, Z., Jiang, S., Yu, Y. (eds.) ProvSec 2012. LNCS, vol. 7496, pp. 125–142. Springer, Heidelberg (2012)
25. Zhang, M., Yang, B., Takagi, T.: Bounded leakage-resilient funtional encryption with hidden vector predicate. The Computer Journal (2012), doi: www.doi.org/comjnl/bxs133

Identity-Based Multisignature with Message Recovery

Kefeng Wang, Yi Mu, and Willy Susilo*

Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Wollongong NSW 2522, Australia
{kw909,ymu,wsusilo}@uow.edu.au

Abstract. We present a new notion of short identity-based multisignature scheme with message recovery. We propose a concrete identity-based multisignature with message recovery scheme based on bilinear pairing in which multiple signers can generate a constant size multisignature on same message regardless of the number of signers. There is no requirement to transmit the original message to the verifier, since the original message can be recovered from the multisignature. Therefore, this scheme minimizes the total length of the original message and the appended multisignature. The proposed scheme is proven to be existentially unforgeable against adaptively chosen message attacks in the random oracle model under the assumption that the Computational Diffie-Hellman problem is hard.

Keywords: Multisignature, Message Recovery, ID-based Cryptography.

1 Introduction

In networks with limited bandwidth and lightweight mobile devices, long digital signatures will obviously be a drawback. Apart from shortening the signature itself, the other effective approach for saving bandwidth is to eliminate the need to transmit the signed original message for verifying a digital signature. In this work, we consider on the latter approach.

Consider n different signers. In order to allow any subgroup of them to produce a joint signature on a message m and convince a verifier that each member of the stated subgroup signed the message, two or more signers cooperate to generate a single compact digital signature in a multisignature scheme. A single multisignature can greatly save communication costs instead of transmitting several individual signatures. To verify the validity of a multisignature, one still needs public keys of all signers. In most applications these public keys will have to be transmitted along with the multisignature. In this case, it partially defeats the primary purpose of using a multisignature scheme, namely to save bandwidth. But the inclusion of some information that uniquely identifies the signers

* This work is supported by the ARC Future Fellowship (FT0991397).

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 91–104, 2013.

seems inevitable for verification. Fortunately, in an identity-based setting, this information can be represented in a more succinct way.

An identity-based signature scheme allows any pair of users to verify each other's signatures without exchanging public key certificates. It resembles an ideal mail system: If you know somebody's name and address you can send him messages that only he can read, and you can verify the signatures that only he could have produced. Compared to the public key of the signer is essentially a random bit string picked from a given set in traditional public key signature algorithms, in the identity-based scenario, the public key of a signer is simply his identity such as his name, email or IP address. The associated private key can only be computed by a trusted Private Key Generator (PKG) using a master secret. It can avoid using certificates which is a big burden to bandwidth in the verifying process of a signature. These features make the identity-based concept particularly appealing for use in conjunction with multisignatures.

When bandwidth is at a premium, another potential problem is that the combined length of the original message and the signature is too large. Signature schemes with total or partial message recovery provide a solution to this problem by embedding all or part of the message within the signature itself. That is, the message does not need to be hashed or sent along with the signature, which saves storage space and communication bandwidth.

Our Contributions. For the first time, this paper presents a provably secure (existentially unforgeable against adaptively chosen message attacks) identity-based multisignature with message recovery scheme based on bilinear pairing under the Computational Diffie-Hellman assumption in the random oracle model. Because the original message can be recovered from the multisignature, there is no need to transmit the original message to the verifier. This scheme minimizes the total length of the original message and the multisignature. We also present a concrete analysis of the reduction to prove the security of the proposed multisignature scheme. More precisely, we can show that if there is an attacker who can forge a valid multisignature to pass the verification, then the Computational Diffie-Hellman problem is solved.

Paper Organization. The rest of this paper is organized as follows: In Section 2, we introduce some related works that have been studied in the literature. In Section 3, we introduce some preliminaries used throughout this paper. In Section 4, we propose a notion of identity-based multisignature with message recovery scheme and present a concrete scheme based on bilinear pairing. We also present a security model and security proof of our scheme in this section. Section 5 concludes the paper.

2 Related Works

In 1984, Shamir introduced the notion of identity-based cryptography to simplify key management of certificate-based public key infrastructures and proposed an identity-based signature scheme [13]. Since then several practical identity-based

signature schemes have been devised [4,6,3,8]. Cha and Cheon [3] proposed an identity-based signature scheme using gap Diffie-Hellman (GDH) groups, and proved their scheme is secure against existential forgery on adaptively chosen message and ID attack under the random oracle model. Hess [8] also proposed an efficient identity-based signature scheme based on pairings. The security of their scheme relies on the hardness of the Diffie-Hellman problem in the random oracle model.

The notion of multisignatures was introduced by Itakura and Nakamura [9]. Several works on this topic have been done [2,10,12]. In [10], the first formalized strong notion of security for multisignatures was proposed. They modified the Schnorr-signature-based multisignature scheme originally proposed by Ohta and Okamoto [12] and proved its security. Gangishetti et al. [5] presented identity-based serial and parallel multisignature schemes using bilinear pairings. Harn and Ren [7] proposed an efficient RSA multisignature scheme based on Shamir's identity-based signature.

In order to minimize the total length of the original message and the appended signature, the message recovery schemes were introduced (e.g. [11]). Zhang et al. [14] proposed an identity-based message recovery signatures scheme. Their scheme can be regarded as the identity based version of Abe-Okamoto's scheme [1]. Their scheme was also extended to achieve an identity-based partial message recovery signature scheme. Based on the scheme due to Zhang et al. [14], we achieved the goal of minimizing the total length of the original message and the appended multisignature in an identity-based setting.

3 Preliminaries

3.1 Bilinear Pairing

Let $\mathbb{G}_1, \mathbb{G}_1'$ be cyclic additive groups generated by P_1, P_1', respectively, whose order are a prime q. Let $\mathbb{G}_2$ be a cyclic multiplicative group with the same order q. We assume there is an isomorphism $\psi : \mathbb{G}_1' \to \mathbb{G}_1$ such that $\psi(P_1') = P_1$. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1' \to \mathbb{G}_2$ be a bilinear mapping with the following properties:

- Bilinearity: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_1', a, b \in \mathbb{Z}_q$.
- Non-degeneracy: There exists $P \in \mathbb{G}_1, Q \in \mathbb{G}_1'$ such that $\hat{e}(P, Q) \neq 1$.
- Computability: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_1'$.

For simplicity, hereafter, we set $\mathbb{G}_1 = \mathbb{G}_1'$ and $P_1 = P_1'$. We note that our scheme can be easily modified for a general case, when $\mathbb{G}_1 \neq \mathbb{G}_1'$.

3.2 CDH Problem

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order the same prime order q. Let P be a generator of $\mathbb{G}_1$. Suppose there exists a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $\mathcal{A}$ be an attacker. $\mathcal{A}$ tries to solve the following problem: Given $(P,\ aP,\ bP)$ for some unknown $a, b \in \mathbb{Z}_q^*$, compute abP.

The success probability of $\mathcal{A}$, which is polynomially bounded with a security parameter l, is defined as

$$Succ^{CDH}_{\mathbb{G}_1,\mathcal{A}}(l) = \Pr[\mathcal{A}(P, aP, bP, abP) = 1; a, b \in \mathbb{Z}^*_q]$$

The CDH problem is said to be intractable, if for every probabilistic polynomial time algorithm $\mathcal{A}, Succ^{CDH}_{\mathbb{G}_1,\mathcal{A}}(l)$ is negligible.

4 Identity-Based Multisignature with Message Recovery

4.1 Definitions

In an identity-based multisignature with message recovery scheme, there is a trusted party Private Key Generator (PKG). PKG is required to generate all the users' private keys.

There are three parties in the system, the PKG, the signer and the verifier. The scheme is ideal for closed groups of users such as the executives of a multinational company or the branches of a large bank, since the headquarters of the corporation can serve as a key generation centre that everyone trusts. This scheme consists of the following four algorithms.

Setup: PKG sets up its secret key s with respect to a security parameter q as the master key of this scheme and publishes the corresponding public key P_{pub}. PKG should generate related groups and point out the generator of these groups. PKG also should describe which bilinear mapping and hash functions will be used in this scheme and publish these public information to all interested principals.

Extract: When a principal requires its private key S_{ID} corresponding its identity ID, this algorithm generates the private key using the master key and the principal's identity, and returns the private key to the principal.

Sign: This is an interactive algorithm. Several principals who got their private keys from the Extract algorithm can firstly generate their individual signatures (v_i, r, U_i) on a message m respectively, and one of them or other specified trusted principal can generate a single compact multisignature (r, U) on the message m corresponding to these principals who participate in this algorithm.

Verify: On receiving a multisignature (r, U) and several principal's identities $ID_1, ID_2, \cdots, ID_n$, this algorithm checks whether the multisignature is a valid multisignature corresponding to these principal's public keys. If the multisignature is checked as valid, the original message m can be recovered from this multisignature.

4.2 Security Model

Boldyreva [2] defined the notion of security for multisignature as no valid multisignature should keep an honest player that part of the alleged subgroup accountable if it did not participate in signing. That is to say, no adversary can

forge an alleged multisignature of some message corresponding to an alleged subgroup of signers so that a verifier can check the multisignature as valid when not all signers of the alleged subgroup did sign the message. In order to achieve its goal, an adversary is allowed to corrupt players and send arbitrary messages during multisignature generation process.

We use a similar definition of existential unforgeability against a chosen message attack of [2]. Our definition is strong enough to capture an adversary who can simulate and observe the scheme. It is defined using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Assume in a subgroup of n signers who want to participate in generating a multisignature, there is only one honest signer. All other $n-1$ members of the subgroup have been corrupted by the adversary. This means the adversary can get secret keys and public keys of corrupted signers. But the adversary only knows the public key of the single honest signer. The adversary can participate in the multisignature generation process on behalf of these $n-1$ corrupted signers. Its goal is to frame the honest signer.

Firstly, challenger $\mathcal{C}$ runs Setup algorithm to get the system's master-key s with respect to a security parameter l and sends the system's public key $P_{pub} = sP$ and other public parameters $\{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, H_1, H_2, F_1, F_2, k_1, k_2\}$ to adversary $\mathcal{A}$.

$\mathcal{A}$ can access the following oracles to start an attack.

$\mathbf{H}_1$ Oracle: For each H_1 hash query with respect to elements $v_1, v_2, \cdots, v_n$ in $\mathbb{G}_2$ and a message m, $\mathcal{C}$ returns a hash value $H_1(v) \in_R \mathbb{Z}_q^*$ corresponding to the product $v = \prod_{i=1}^{n} v_i$ of these elements $v_1, v_2, \cdots, v_n$.

$\mathbf{H}_2$ Oracle: For each H_2 hash query with respect to an user ID_i, $\mathcal{C}$ returns a hash value $Q_{ID_i} \in_R \mathbb{G}_1$ as the user ID_i's public key.

Extract Oracle: For each Extract query with respect to a user ID_i except for the honest user ID^*, $\mathcal{C}$ returns $S_{ID_i} = sQ_{ID_i}$ as the user's private key, in which the Q_{ID_i} is the H_2 hash value of the user ID_i's identity.

Sign Oracle: For each Sign query on arbitrary message m with respect to a subgroup of n signer's identities $ID_1, ID_2, \cdots, ID_n$, this oracle can be divided into two phases.

In the first phase, $n-1$ signers generate their individual v_i by randomly selecting an element K_i from $\mathbb{G}_1$ and then computing $v_i = \hat{e}(K_i, P)$. These $n-1$ signers send their v_i and a target signer's identity ID_t to $\mathcal{C}$. $\mathcal{C}$ outputs a random element $v_t \in_R \mathbb{G}_2$ corresponding to the target signer ID_t.

In the second phase, these $n-1$ signers compute v using v_t and all v_i as $v = \prod_{i=1}^{n-1} v_i \cdot v_t$. At the same time, $\mathcal{C}$ computes the same v using the same method. These $n-1$ signers generate and send their own individual signatures (v_i, r, U_i) and message m to $\mathcal{C}$. $\mathcal{C}$ returns a valid multisignature (r, U) on message m with respect to n signers include these $n-1$ signers and the target signer.

Output: $\mathcal{A}$ outputs an alleged multisignature (r, U) on a target message m^* with respect to a subgroup of n signers $ID_1, \cdots, ID^*, \cdots, ID_n$ in which includes

an honest signer ID^* who did not participate in the multisignature generation process. If there was no Sign queries with respect to the target message m^* and a subgroup of signers in which includes the honest signer ID^* have been queried to Sign Oracle, and there was no Extract query with respect to the honest signer ID^* has been queried to Extract Oracle, $\mathcal{A}$ wins the game if the multisignature (r, U) can be verified as a valid multisignature.

If there is no such polynomial-time adversary that can forge a valid multisignature with respect to a subgroup of signers which includes an honest signer, while the honest signer did not participate in the multisignature generation process in the game described above, we say that the multisignature scheme is secure against existential forgery under chosen message attack.

The success probability of an adversary to win the game is defined by

$$Succ_{\mathcal{A}}^{UF-IDMMR-CMA}(l).$$

We say that an identity-based multisignature with message recovery scheme is existentially unforgeable under a chosen message attack if the success probability of any polynomially bounded adversary in the above game is negligible. In other words,

$$Succ_{\mathcal{A}}^{UF-IDMMR-CMA}(l) \leq \epsilon.$$

4.3 Proposed Scheme

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of the same prime order q. Let P be a generator of $\mathbb{G}_1$. Suppose there exists a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

Setup: PKG chooses a random number $s \in \mathbb{Z}_q^*$ and keeps it as the master-key of this system. This master-key is known only by PKG itself. PKG sets $P_{pub} = sP$ as the system's public key and publishes this public key and other system parameters $\{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, H_1, H_2, F_1, F_2, k_1, k_2\}$.

Here $|q| = k_1 + k_2$. $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \to \mathbb{G}_1^*$, $F_1 : \{0,1\}^{k_2} \to \{0,1\}^{k_1}$ and $F_2 : \{0,1\}^{k_1} \to \{0,1\}^{k_2}$ are four cryptographic hash functions.

Extract: A user submits his/her identity information ID_i to PKG. PKG computes the user's public key as $Q_{ID_i} = H_2(ID_i)$, and returns $S_{ID_i} = sQ_{ID_i}$ to the user as his/her private key.

Sign: Let the message be $m \in \{0,1\}^{k_2}$.

Each signer randomly selects an element K_i in $\mathbb{G}_1$ and computes $v_i = \hat{e}(K_i, P)$. v_i is broadcast to other signers.

Once each signer's v_i are available through the broadcast channel. They compute their individual signatures as follows:

$$\begin{aligned} v &= \prod_{i=1}^{n} v_i = \hat{e}(K_1, P)\hat{e}(K_2, P) \cdots \hat{e}(K_n, P) = \hat{e}(\sum_{i=1}^{n} K_i, P) \\ f &= F_1(m)||(F_2(F_1(m)) \oplus m) \\ r &= H_1(v) + f \\ U_i &= K_i - rS_{ID_i} \end{aligned}$$

In the above computation, the symbol || denotes concatenation of two operands.

Each signer transmits its individual signature (v_i, r, U_i) to the clerk who may be one of these signers or other specified trusted principal.

Once the clerk receives an individual signature (v_i, r, U_i), he needs to verify the validity of this individual signature. The verification procedure of the clerk checks that

$$v_i = \hat{e}(U_i, P)\hat{e}(Q_{ID_i}, P_{pub})^r.$$

Once all individual signatures are received and verified by the clerk as valid, the multisignature of message m with respect to these signers who generate these individual signatures can be generated as (r, U), where

$$U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} K_i - r \sum_{i=1}^{n} S_{ID_i}$$

Verify: Given a multisignature (r, U) and n signer's identity $ID_1, ID_2, \cdots, ID_n$ who stated have signed a message, a verifier computes

$$r - H_1(\hat{e}(U, P)\hat{e}(\sum_{i=1}^{n} Q_{ID_i}, P_{pub})^r) = f$$

and

$$m = [f]_{k_2} \oplus F_2([f]^{k_1}).$$

In the above computation, the subscript k_2 of f denotes the least significant k_2 bits of f, and the superscript k_1 of f denotes the most significant k_1 bits of f.

The verifier checks whether $[f]^{k_1} = F_1(m)$ holds. If this equation holds, the verifier accepts this multisignature and recovers the original message m from this multisignature. Otherwise, the verifier rejects the multisignature.

4.4 Security Analysis

Theorem 1. *This identity-based multisignature with message recovery scheme is correct and sound.*

Proof. The correctness of this identity-based multisignature with message recovery scheme can be shown as follows.

When the individual signature (v_i, r, U_i) is verified,

$$\begin{aligned}
&\hat{e}(U_i, P)\hat{e}(Q_{ID_i}, P_{pub})^r \\
&= \hat{e}(K_i - rS_{ID_i}, P)\hat{e}(Q_{ID_i}, sP)^r \\
&= \hat{e}(K_i - rS_{ID_i}, P)\hat{e}(sQ_{ID_i}, P)^r \\
&= \hat{e}(K_i - rS_{ID_i}, P)\hat{e}(S_{ID_i}, P)^r \\
&= \hat{e}(K_i - rS_{ID_i}, P)\hat{e}(rS_{ID_i}, P) \\
&= \hat{e}(K_i, P) \\
&= v_i
\end{aligned}$$

This means if the individual signature (v_i, r, U_i) is indeed generated by signer ID_i, the equation $v_i = \hat{e}(U_i, P)\hat{e}(Q_{ID_i}, P_{pub})^r$ will always hold.

When the multisignature (r, U) is verified, we can recover v which is used by each signer in the multisignature generation from the following computation.

$$\begin{aligned}
&\hat{e}(U, P)\hat{e}(\sum_{i=1}^{n} Q_{ID_i}, P_{pub})^r \\
&= \hat{e}(\sum_{i=1}^{n} K_i - r\sum_{i=1}^{n} S_{ID_i}, P)\hat{e}(\sum_{i=1}^{n} Q_{ID_i}, sP)^r \\
&= \hat{e}(\sum_{i=1}^{n} K_i - r\sum_{i=1}^{n} S_{ID_i}, P)\hat{e}(s\sum_{i=1}^{n} Q_{ID_i}, P)^r \\
&= \hat{e}(\sum_{i=1}^{n} K_i - r\sum_{i=1}^{n} S_{ID_i}, P)\hat{e}(\sum_{i=1}^{n} S_{ID_i}, P)^r \\
&= \hat{e}(\sum_{i=1}^{n} K_i - r\sum_{i=1}^{n} S_{ID_i}, P)\hat{e}(r\sum_{i=1}^{n} S_{ID_i}, P) \\
&= \hat{e}(\sum_{i=1}^{n} K_i, P) \\
&= v
\end{aligned}$$

Then, using this v and part of the multisignature r, we can recover f from the following computation.

$$\begin{aligned}
&r - H_1(\hat{e}(U, P)\hat{e}(\sum_{i=1}^{n} Q_{ID_i}, P_{pub})^r) \\
&= r - H_1(v) \\
&= H_1(v) + f - H_1(v) \\
&= f
\end{aligned}$$

Since f is computed from $f = F_1(m)||(F_2(F_1(m)) \oplus m)$, we will try to recover the original message m from f like this:

$$\begin{aligned}
&[f]_{k_2} \oplus F_2([f]^{k_1}) \\
&= [F_1(m)||(F_2(F_1(m)) \oplus m)]_{k_2} \oplus F_2([F_1(m)||(F_2(F_1(m)) \oplus m)]^{k_1}) \\
&= F_2(F_1(m)) \oplus m \oplus F_2(F_1(m)) \\
&= m
\end{aligned}$$

As previously declared, the subscript k_2 and the superscript k_1 of f denote the least significant k_2 and the most significant k_1 bits of f respectively.

After recovering the alleged original message m, we need to check whether $[f]^{k_1} = F_1(m)$ to verify the validity of the multisignature. If this equation holds, the multisignature (r, U) is valid and the original message m is recovered. Otherwise, the multisignature (r, U) is a forged one. □

Theorem 2. *This identity-based multisignature with message recovery scheme is existentially unforgeable under a chosen message attack in the random oracle model, under the assumption that the Computational Diffie-Hellman problem is hard.*

Proof. Assume there is an algorithm $\mathcal{A}$ that can forge a multisignature under a chosen message attack. There will be another algorithm $\mathcal{B}$ that can run the algorithm $\mathcal{A}$ to solve the CDH problem.

In the process of $\mathcal{B}$ using $\mathcal{A}$ to solve the CDH problem, $\mathcal{B}$ needs to simulate all the oracles that $\mathcal{A}$ can query as follows.

Setup: $\mathcal{B}$ sets up $P_{pub} = aP$ as the system's public key and sends P_{pub} and other system parameters $\{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, H_1, H_2, F_1, F_2, k_1, k_2\}$ to adversary $\mathcal{A}$. In this case, $\mathcal{B}$ only knows the system's public key is aP, but he does not know the corresponding master-key s which is actually a in this concrete situation. Two hash functions F_1, F_2 of the four hash functions used in this scheme are published as normal hash functions. The other two hash functions H_1, H_2 are both treated as random oracles.

$\mathbf{H}_1$ Queries: $\mathcal{B}$ creates and keeps two lists of tuples to simulate H_1 Oracle. At the beginning of the simulation, both of these lists are empty.

One list is called Hv_n-List which is used to store tuples like

$$(v_1, v_2, \cdots, v_n, h).$$

In this type of tuples, the first n elements come from group $\mathbb{G}_2$ and the last element comes from $\mathbb{Z}_q^*$.

After receiving a H_1 hash query with respect to several elements $v_1, v_2, \cdots, v_n$ in $\mathbb{G}_2$ and a message m, if the first n elements $v_1, v_2, \cdots, v_n$ are not as a record in the v^*-List which is constructed in the Sign Oracle and not in a record in this Hv_n-List, $\mathcal{B}$ randomly selects $h \in \mathbb{Z}_q^*$ and returns h as the H_1 hash value of $v = \prod_{i=1}^{n} v_i$. Then, $\mathcal{B}$ records the tuple $(v_1, v_2, \cdots, v_n, h)$ in this Hv_n-List. If the first n elements $v_1, v_2, \cdots, v_n$ are already in a record in this Hv_n-List, $\mathcal{B}$ only returns the corresponding h in the record as the H_1 hash value. All in all, this list matches the situation that the honest signer is not required to participate in the multisignature generation.

The other list is called Hv^*-List which is used to store tuples like

$$(m, v_1, v_2, \cdots, v_{n-1}, v^*, y - f).$$

In this type of tuples, the first element m is an arbitrary message to be signed by a subgroup which includes the honest signer. The next n elements come from group $\mathbb{G}_2$ and the last element comes from $\mathbb{Z}_q^*$.

After receiving an H_1 hash query with respect to several elements $v_1, v_2, \cdots, v^*$ in $\mathbb{G}_2$ and a message m, if the first n elements $v_1, v_2, \cdots, v_{n-1}, v^*$ are as a record in the v^*-List which is constructed in the Sign Oracle but not as a record in this Hv^*-List, $\mathcal{B}$ returns $y - f$ as the H_1 hash value of $v = \prod_{i=1}^{n-1} v_i \cdot v^*$ in which y is got from the corresponding record in the v^*-List and f is computed by the

equation $f = F_1(m)||(F_2(F_1(m)) \oplus m)$ with respect to the message m. Then, $\mathcal{B}$ records the tuple $(m, v_1, v_2, \cdots, v_{n-1}, v^*, y-f)$ in this Hv^*-List. Note that for the same n elements $v_1, v_2, \cdots, v_{n-1}, v^*$ but different message m, the value y is same because it comes from the same record in the v^*-List, but the value f is different because it is computed by the equation $f = F_1(m)||(F_2(F_1(m)) \oplus m)$ for different message. So, the returned hash value $y - f$ is different. In this case, we need to add a new record in this Hv^*-List. If these elements $m, v_1, v_2, \cdots, v_{n-1}, v^*$ are already in a record in this Hv^*-List, $\mathcal{B}$ only returns the corresponding $y - f$ in the record as the H_1 hash value. In a word, this list matches the situation that the honest signer is required to participate in the multisignature generation.

H_2 Queries: $\mathcal{B}$ creates and keeps one list H_2-List to simulate H_2 Oracle. At the beginning of the simulation, this list is empty.

For each H_2 hash query with respect to a signer ID_i except for the honest signer ID^*, if ID_i is not in a record in this H_2-List, $\mathcal{B}$ randomly selects $k_i \in \mathbb{Z}_q^*$ and returns $Q_{ID_i} = k_i P$ as the H_2 hash value of ID_i. Then, $\mathcal{B}$ records the tuple (ID_i, k_i, Q_{ID_i}) in this H_2-List. If ID_i is already in a record in this H_2-List, $\mathcal{B}$ only returns the corresponding Q_{ID_i} in the record as the H_2 hash value.

For the H_2 hash query with respect to the honest signer ID^*, $\mathcal{B}$ returns $Q_{ID^*} = bP$ as the H_2 hash value of ID^*.

Extract Queries: $\mathcal{B}$ creates and keeps one list Ex-List to simulate Extract Oracle. At the beginning of the simulation, this list is empty.

For each Extract query with respect to a signer ID_i except for the honest signer ID^*, if ID_i is not in a record in this Ex-List, $\mathcal{B}$ looks up the H_2-List which is created by H_2 Oracle to find the record about ID_i. Because a signer needs to query H_2 Oracle prior to its any other operation, the Extract Oracle can always find out the record with respect to ID_i in the H_2-List. Using the k_i value in the record in the H_2-List with respect to ID_i, $\mathcal{B}$ returns

$$S_{ID_i} = k_i P_{pub} = k_i aP = ak_i P = aQ_{ID_i}$$

as the signer ID_i's private key. Then, $\mathcal{B}$ records the tuple (ID_i, S_{ID_i}) in this Ex-List. If ID_i is already in a record in this Ex-List, $\mathcal{B}$ only returns the corresponding S_{ID_i} in the record as the signer ID_i's private key.

Sign Queries: $\mathcal{B}$ creates and keeps two lists of tuples to simulate Sign Oracle. At the beginning of the simulation, both of these lists v_n-List and v^*-List are empty. v_n-List matches the situation that the honest signer is not required to participate in the multisignature generation. v^*-List matches the situation that the honest signer is required to participate in the multisignature generation. Without loss of generality, we assume that the target signer is always the last signer ID_n.

For each Sign query with respect to an arbitrary message m and a subgroup of n signers $ID_1, ID_2, \cdots, ID_n$, this oracle are divided into two phases.

In the first phase, $n-1$ signers $ID_1, ID_2, \cdots, ID_{n-1}$ generate their individual $v_i = \hat{e}(K_i, P)$ in which K_i is randomly selected from group $\mathbb{G}_1$ and send their v_i and the target signer's identity ID_n to $\mathcal{B}$.

If ID_n is not the honest signer ID^*, $\mathcal{B}$ can randomly select an element K_n from group $\mathbb{G}_1$ and compute $v_n = \hat{e}(K_n, P)$. $\mathcal{B}$ returns v_n to $\mathcal{A}$ and records the tuple

$$(v_1, v_2, \cdots, v_{n-1}, v_n, K_n)$$

in the v_n-List.

If ID_n is the honest signer ID^*, $\mathcal{B}$ can randomly select two integers $x, y \in_R \mathbb{Z}_q^*$. Then $\mathcal{B}$ computes

$$v^* = \hat{e}(aP, bP)^y \cdot \hat{e}(P, P)^x = \hat{e}((yab + x)P, P)$$

and returns this v^* to $\mathcal{A}$. In this case, the corresponding random element from group $\mathbb{G}_1$ is

$$K^* = (yab + x)P.$$

$\mathcal{B}$ records the tuple

$$(v_1, v_2, \cdots, v_{n-1}, v^*, y, x)$$

in the v^*-List.

In the second phase, $\mathcal{A}$ computes $f = F_1(m)||(F_2(F_1(m)) \oplus m)$ with respect to message m. $\mathcal{A}$ queries H_1 Oracle the H_1 hash value with respect to $(v_1, v_2, \cdots, v_{n-1}, v_n)$ or $(v_1, v_2, \cdots, v_{n-1}, v^*)$ and message m and uses this H_1 hash value to compute the second part of $n-1$ signer's individual signature (v_i, r, U_i) as $r = H_1(v) + f$. $\mathcal{A}$ computes the third part

$$U_i = K_i - rS_{ID_i} = K_i - raQ_{ID_i}$$

of $n-1$ signer's individual signatures by the real Sign algorithm using the previous r and the corresponding private key $S_{ID_i} = aQ_{ID_i}$ got from the Extract Oracle and sends these $n-1$ individual signatures and message m to $\mathcal{B}$.

$\mathcal{B}$ needs to compute $f = F_1(m)||(F_2(F_1(m)) \oplus m)$ at first. If ID_n is not the honest signer ID^*, $\mathcal{B}$ computes the individual signature (v_n, r, U_n) by the real Sign algorithm using the corresponding r which is computed the same as previous process and S_{ID_n} which is got from Extract Oracle. Then, $\mathcal{B}$ computes $U = \sum_{i=1}^{n} U_i$ and returns (r, U) as the multisignature on message m with respect to n signers $ID_1, ID_2, \cdots, ID_n$. In this case, both of the individual signature (v_n, r, U_n) of ID_n and the multisignature (r, U) can pass their own verification process. These verifications can be checked by using the method in Theorem 4.1.

If ID_n is the honest signer ID^*, $\mathcal{B}$ computes r by using H_1 Oracle as

$$r = H_1(v) + f = y - f + f = y,$$

and simulates the third part of the honest signer ID^*'s individual signature as

$$U^* = K^* - rS_{ID^*} = (yab + x)P - y \cdot abP = xP$$

in which the corresponding x can be found out in the v^*-List.

Then, $\mathcal{B}$ computes

$$U = \sum_{i=1}^{n-1} U_i + U^*$$

and returns (r, U) as the multisignature on message m with respect to n signers $ID_1, ID_2, \cdots, ID_{n-1}, ID^*$.

Verify: Both of the individual signature and the multisignature can pass the verifications. The individual signature (v^*, r, U^*) can pass the verification as follows.

$$\begin{aligned} & \hat{e}(U^*, P)\hat{e}(Q_{ID^*}, P_{pub})^r \\ = \ & \hat{e}(xP, P)\hat{e}(bP, aP)^y \\ = \ & \hat{e}(xP, P)\hat{e}(yabP, P) \\ = \ & \hat{e}((yab + x)P, P) \\ = \ & v^* \end{aligned}$$

The multisignature (r, U) can also pass the verification as follows.

$$\begin{aligned} & \hat{e}(U, P)\hat{e}(\sum_{i=1}^{n} Q_{ID_i}, P_{pub})^r \\ = \ & \hat{e}(\sum_{i=1}^{n-1} U_i + U^*, P)\hat{e}(\sum_{i=1}^{n-1} Q_{ID_i} + Q_{ID^*}, aP)^y \\ = \ & \hat{e}(\sum_{i=1}^{n-1} U_i, P)\hat{e}(U^*, P)\hat{e}(\sum_{i=1}^{n-1} Q_{ID_i}, aP)^y\hat{e}(Q_{ID^*}, aP)^y \\ = \ & \hat{e}(\sum_{i=1}^{n-1} U_i, P)\hat{e}(\sum_{i=1}^{n-1} Q_{ID_i}, aP)^y\hat{e}(U^*, P)\hat{e}(Q_{ID^*}, aP)^y \\ = \ & \hat{e}(\sum_{i=1}^{n-1} K_i - ya\sum_{i=1}^{n-1} Q_{ID_i}, P)\hat{e}(ya\sum_{i=1}^{n-1} Q_{ID_i}, P)\hat{e}(xP, P)\hat{e}(yabP, P) \\ = \ & \hat{e}(\sum_{i=1}^{n-1} K_i, P)\hat{e}((yab + x)P, P) \\ = \ & \prod_{i=1}^{n-1} v_i \cdot v^* \end{aligned}$$

Since we have assumed that adversary $\mathcal{A}$ can forge a multisignature under a chosen message attack, after the simulation process above, $\mathcal{A}$ can output a valid multisignature (r_1, U_1) on message m with respect to a subgroup of n signers which includes the honest signer ID^* who did not participate in the multisignature generation. There are two restrictions about this multisignature generation. The first one is there is no query to Extract Oracle with respect to the honest signer ID^*. The second one is there is no query to Sign Oracle with respect to the message m and a subgroup of signers which includes the honest signer ID^*. $\mathcal{B}$ can compute the third part U_1^* of the honest signer ID^*'s individual signature (v_i^*, r_1^*, U_1^*) from the valid multisignature (r_1, U_1) as follows.

$$U_1^* = U - \sum_{i=1}^{n-1} U_i$$

All these U_i come from $\mathcal{A}$ in the second phase of the Sign Query.

$\mathcal{B}$ can reset all the oracles and runs $\mathcal{A}$ for the second time. At the end of the simulation, with a non-negligible probability $\mathcal{B}$ can get another different individual signature (v_2^*, r_2^*, U_2^*) on the same message m and with respect to the same honest signer ID^* when v_1^* equals to v_2^*. That means for two different random integer pairs (x_1, y_1) and (x_2, y_2),

$$K_1^* = (y_1 ab + x_1)P \text{ is equal to } K_2^* = (y_2 ab + x_2)P.$$

Both (r_1^*, U_1^*) and (r_2^*, U_2^*) can pass the verification process, and K_1^* equals K_2^*. So,

$$\begin{aligned}
\hat{e}(U_1^*, P)\hat{e}(Q_{ID^*}, P_{pub})^{r_1^*} &= \hat{e}(U_2^*, P)\hat{e}(Q_{ID^*}, P_{pub})^{r_2^*} \\
\hat{e}(U_1^* + r_1^* S_{ID^*}, P) &= \hat{e}(U_2^* + r_2^* S_{ID^*}, P) \\
U_1^* + r_1^* S_{ID^*} &= U_2^* + r_2^* S_{ID^*} \\
(r_1^* - r_2^*) S_{ID^*} &= U_2^* - U_1^* \\
S_{ID^*} &= (r_1^* - r_2^*)^{-1}(U_2^* - U_1^*)
\end{aligned}$$

In this case, $\mathcal{B}$ can compute the honest signer ID^*'s private key S_{ID^*} when he only knows the honest signer ID^*'s public key Q_{ID^*} and the system's public key P_{pub}. Because S_{ID^*} is expressed as abP, Q_{ID^*} is expressed as bP, P_{pub} is expressed as aP, $\mathcal{B}$ can solve an CDH problem if $\mathcal{A}$ is able to forge valid multisignatures.

If there is no such polynomial-time adversary that can forge a valid multisignature corresponding to a subgroup of signers that include an honest signer, we say that this identity-based multisignature with message recovery scheme is secure against existential forgery under chosen message attack. □

5 Conclusion

We proposed a new notion of short identity-based multisignature scheme. The notion of short identity-based multisignature scheme can be viewed as identity-based multisignature with message recovery scheme. In order to sign short messages using a scheme that minimizes the total length of the original message and the appended signature, we proposed an concrete identity-based multisignature with message recovery scheme based on bilinear pairing in which multiple signers can generate a constant size multisignature on same message regardless the number of signers and there is no need to transmit the original message to verifier, because it can be recovered from the multisignature. We also proved that our scheme is secure against existential forgery on adaptively chosen message attack in the random oracle model, under the hardness assumption of CDH problem.

References

1. Abe, M., Okamoto, T.: A signature scheme with message recovery as secure as discrete logarithm. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 378–389. Springer, Heidelberg (1999)
2. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
3. Cha, J., Cheon, J.: An identity-based signature from gap diffie-hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2002)
4. 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)
5. Gangishetti, R., Gorantla, M., Das, M., Saxena, A.: Identity based multisignatures. Informatica 17(2), 177–186 (2006)
6. Guillou, L.C., Quisquater, J.-J.: A paradoxical indentity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
7. Harn, L., Ren, J.: Efficient identity-based rsa multisignatures. Computers & Security 27(1), 12–15 (2008)
8. Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
9. Itakura, K., Nakamura, K.: A public-key cryptosystem suitable for digital multisignatures. NEC Research & Development 71, 1–8 (1983)
10. Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 245–254. ACM (2001)
11. Nyberg, K., Rueppel, R.: A new signature scheme based on the dsa giving message recovery. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 58–61. ACM (1993)
12. Okamoto, T.: Multi-signature schemes secure against active insider attacks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 82(1), 21–31 (1999)
13. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
14. Zhang, F., Susilo, W., Mu, Y.: Identity-based partial message recovery signatures (or how to shorten ID-based signatures). In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 45–56. Springer, Heidelberg (2005)

Improving the Message-Ciphertext Rate of Lewko's Fully Secure IBE Scheme*

Dingding Jia[1], Bao Li[1], Yamin Liu[1], and Qixiang Mei[2]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences
[2] College of Information, GuangDong Ocean University
{ddjia,lb,ymliu}@is.ac.cn,nupf@163.com

Abstract. In Eurocrypt 2012, Lewko presented a fully secure IBE scheme in the prime order setting based on the decisional linear assumption. We note that in Lewko's scheme, some random factor involved in the ciphertext can further be used to hide yet another message, and hence get a new fully secure IBE scheme with better message-ciphertext rate. Similar to Lewko's scheme, we use dual pairing vector space in prime order bilinear groups to simulate the canceling and parameter hiding properties of composite order settings. The security of our scheme is based on the subspace assumption, which can be reduced to the decisional linear assumption. We employ the dual system encryption technique in our security proof.

Keywords: DLIN assumption, fully secure IBE, canceling, parameter hiding, dual system encryption.

1 Introduction

In 1984, Shamir [16] introduced the notion of Identity-Based Encryption (IBE). An IBE is a public key encryption scheme in which the public key can be set to any string representing one's identity. A trusted authority holds a master secret key which allows it to create secret keys for any identity. There are two kinds of security requirements for IBE schemes: a weaker one called selective-ID security in which the adversary selects an ID priori to other moves and attacks the fixed ID; and a stronger one called fully security in which the adversary adaptively selects the ID to be attacked during the security game. IBE schemes are first realized in the random oracle model, by Boneh and Franklin [3] using bilinear groups and Cocks [7] under quadratic residue assumption. Later, realization in the standard model was proposed by Boneh and Boyen [2] and Canetti, Halevi and Katz [6], but only selective-ID security was achieved in [2,6].

* This work is Supported by the National Basic Research Program of China (973 project)(No.2013CB338002), the National Nature Science Foundation of China (No.61070171, No.61272534), the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702 and the State Key Laboratory of Information Security IIEs Research Project on Cryptography (No. Y2Z0021103).

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 105–116, 2013.

The first fully secure IBE scheme that has a tight reduction in the standard model was proposed by Waters in 2009 [17], in which a new proof technique called dual system encryption was used. The IBE scheme of [17] was constructed with bilinear maps in the prime order setting, and its security was based on the decisional linear (DLIN) assumption and bilinear decisional Diffie-Hellman (BDDH) assumption. However, the scheme was bothered by too long parameters and complicated structure. In [15] Ramannal, et al. gave a simplification of Waters' scheme in asymmetric bilinear groups, but based on assumptions which are not so standard.

Shortly later another fully secure IBE scheme was given by Lewko and Waters in 2010 [12] in the composite order setting. Their scheme has simple structure resembles that of [2]. In the security proof they used two properties of composite order groups: one is called "canceling", that is, for any $g_1 \in G_{p_1}, g_2 \in G_{p_2}$, $e(g_1, g_2) = 1$; the other is called "parameter hiding", which means g_1^a information-theoretically hides $a \ mod \ p_2$.

Since the computation of bilinear map in composite order groups is less efficient, much effort has been contributed to finding transformations to prime order settings. In 2010, Freeman [8] provided a generic method for transforming schemes in composite order settings [4,9,11] to prime order settings, but the method can not be applied to some schemes. Lewko [10] observed that the method of [8] perfectly simulated the "canceling" property, yet was not a useful approach to achieve the "parameter hiding" property. Lewko [10] used dual pairing vector space which was proposed by Okamoto and Takashima [13,14] to simulate both properties in the prime order setting, and got a fully secure IBE scheme akin to the one in [2]. Specially, to achieve the canceling property, a pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ was used in [10]; to achieve the parameter hiding property, for a matrix A to be hidden, a pair of random dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ was chosen, then A was embedded in $(\mathbb{B}, \mathbb{B}^*)$, and a new pair of dual orthonormal bases $(\mathbb{B}_A, \mathbb{B}_A^*)$ was generated, which looks random to the adversary who does not know $(\mathbb{B}, \mathbb{B}^*)$. The scheme in [10] has simple structure and its security is based on the DLIN assumption.

In the IBE sheme in [10], both ciphertext and secret key employ two parameters, s_1, s_2 and r_1, r_2. The two exponents play almost the same roles, except that in the ciphertext, the element hiding the message only uses s_1. We find that both parameters could be used to hide messages, thus more messages could be encrypted without adding too many elements to the ciphertext.

Our Result. In this paper, we improve the IBE scheme presented in [10] by using both parameters to hide messages in the ciphertext, hence get an IBE scheme that can encrypt two elements in the target group simultaneously. As in [10], we use dual orthonormal bases of dimension 6. Compared to Lewko's scheme, we only add one element from the target group to the public parameters and ciphertexts, double the length of the decryption key. In Table 1. we give a comparison of Lewko's scheme and ours. The columns #msg, #cpr, #key provide the number of group elements in the messages, ciphertexts and decryption keys. Encryption efficiency counts the number of scalar multiplications in G for

encrypting every group element while decryption efficiency counts the number of pairings that are required for decrypting every group element. Key generation efficiency is given by the number of scalar multiplications in G. We can see that we encrypt double messages on the cost of double key length, and our scheme has better message-ciphertext rate. Similar to Lewko's paper, we use dual pairing vector space in prime order bilinear groups to realize the canceling and parameter hiding properties.

Table 1. A comparison of Lewko's IBE scheme and ours

Scheme	#msg	#cpr	#key	enc eff	dec eff	keygen eff	msg-cpr rate
Lewko's	1	7	6	25	6	6	1:7
Ours	2	8	12	13	6	12	2:8

In the security proof we use the dual system encryption technique. Firstly we change the challenge ciphertext to be semi-functional. Secondly answers to key extraction queries are changed to be semi-functional one by one. Here we change every key in two steps: temporary semi-functional first, then semi-functional. Finally we change the challenge ciphertext to a semi-functional encryption of a random message. We argue that any polynomial time adversary can not tell the difference between two adjacent games.

Moreover, in the last game the challenge ciphertext is independent of the identity, so our scheme is anonymous. The IBE scheme in Lewko's paper is also anonymous for the same reason. Anonymous IBE [5] is a useful component to construct public key encryption with keyword search (PEKS) schemes [1].

The rest of our paper is organized as follows: in section 2 we give the formal definition of IBE and the security definition; in section 3 we give the complexity assumptions; in section 4 we describe our construction and prove its security; section 5 is the conclusion of the whole paper.

2 Definitions

2.1 IBE

Definition 1 (IBE). *An Identity-Based Encryption scheme (IBE) [16] is a tuple of four probabilistic polynomial time (PPT) algorithms: (**Setup, Keygen, Encrypt, Decrypt.**)*

Setup(1^λ)**:** *take as input the security parameter* λ *and output public parameters* PP *and the master secret key* Msk.

Keygen(Msk, ID)**:** *take as input the master secret key* Msk*, identity* ID *and output a private key* Sk_{ID}.

Encrypt(PP, M, ID)**:** *take as input public parameters* PP*, message* M *and identity* ID *and output a ciphertext* C.

Decrypt(C, Sk_{ID})**:** *take as input the ciphertext* C *and secret key* Sk_{ID} *and output the message* M*.*

For correctness, we require that all properly generated ciphertexts can be decrypted correctly.

2.2 Security Definition

Here we give the fully security definition of IBE. The security of an IBE scheme is defined using the following game between an adversary $\mathcal{A}$ and a challenger.

Setup: The challenger runs the *Setup* algorithm, gives public parameters PP to the adversary $\mathcal{A}$ and keeps the master secret key Msk to itself.
Phase 1: $\mathcal{A}$ adaptively issues identity queries ID, the challenger responds with Sk_{ID} by calling the *Keygen* algorithm.
Challenge: $\mathcal{A}$ gives two messages and a challenge identity (M_0, M_1, ID^*) to the challenger. The challenge identity should never be queried in phase 1. The challenger picks a random bit b and responds with $Encrypt(PP, M_b, ID^*)$.
Phase 2: $\mathcal{A}$ adaptively issues additional queries as in Phase 1, with the restriction that ID^* is never allowed to be queried.
Guess: $\mathcal{A}$ outputs a guess b' of b.

The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{IBE} = \left| Pr[b' = b] - \frac{1}{2} \right|$.

Definition 2 (Fully Security). *An IBE scheme is fully secure if for all PPT adversary* $\mathcal{A}$*,* $Adv_{\mathcal{A}}^{IBE}$ *is negligible in* λ*.*

3 Complexity Assumptions

In this section we introduce the complexity assumptions that will be used in our proof. As in [10], we use dual pairing vector space to achieve the canceling and parameter hiding properties in the prime order setting.

3.1 Prime Order Symmetric Bilinear Maps

Let G, G_T be cyclic groups of prime order p, with a bilinear map $e : G \times G \to G_T$ satisfying the following properties:

- (Bilinear) $\forall u_1, u_2 \in G, \forall a, b \in \mathbb{Z}_p, e(u_1^a, u_2^b) = e(u_1, u_2)^{ab}$.
- (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order p in G_T.

We assume that group operations in G and $\mathbb{G}_T$ as well as the bilinear map e can be efficiently computed.

Vector computation rules are defined as follows:

- For $\mathbf{v} = (v_1, v_2, ..., v_n) \in \mathbb{Z}_p^n$ and $g \in G, g^{\mathbf{v}} := (g^{v_1}, g^{v_2}, ..., g^{v_n})$.
- For any $a \in \mathbb{Z}_p$ and $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p^n$,

$$g^{a\mathbf{v}} := (g^{av_1}, ..., g^{av_n}),\ g^{\mathbf{v}+\mathbf{w}} := (g^{v_1+w_1}, ..., g^{v_n+w_n}).$$

- e_n is used to denote the product of the component-wise pairings:

$$e_n(g^{\mathbf{v}}, g^{\mathbf{w}}) := \prod_{i=1}^{n} e(g^{v_i}, g^{w_i}) = e(g, g)^{\mathbf{v}\cdot\mathbf{w}}.$$

Dual Pairing Vector Spaces. Next let us review the concept of dual pairing vector spaces from [13,14,10]. For a dimension n, we say two random chosen bases $\mathbb{B} := (\mathbf{b}_1, ..., \mathbf{b}_n), \mathbb{B}^* := (\mathbf{b}_1^*, ..., \mathbf{b}_n^*)$ are dual orthonormal if for $i \neq j$,

$$\mathbf{b}_i \cdot \mathbf{b}_j^* = 0 (mod\ p)$$

and $\mathbf{b}_i \cdot \mathbf{b}_i^* = \psi$ for all i, where ψ is a uniformly random element in $\mathbb{Z}_p$. In the following we let $Dual(\mathbb{Z}_p^n)$ be the set of dual orthonormal bases and let $(\mathbb{B}, \mathbb{B}^*) \overset{R}{\leftarrow} Dual(\mathbb{Z}_p^n)$ denote choosing random dual orthonormal bases $\mathbb{B}$ and $\mathbb{B}^*$ from $\mathbb{Z}_p^n$.

Canceling Property. For a generator $g \in G$, we note that $e_n(g^{\mathbf{b}_i}, g^{\mathbf{b}_j^*}) = 1$ whenever $i \neq j$. We call this property as "canceling" and it will play an important role in our scheme.

Parameter Hiding Property. Next we will introduce the other property called "parameter hiding". Generally speaking, one can apply an invertible matrix A to a random pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*) \overset{R}{\leftarrow} Dual(\mathbb{Z}_p^n)$, and get a new pair of dual orthonormal bases which is randomly distributed for adversaries who do not know $(\mathbb{B}, \mathbb{B}^*)$. Hence the newly generated bases information-theoretically hide the matrix A. Next we describe how the new pair of dual orthonormal bases is generated.

For $(\mathbb{B}, \mathbb{B}^*) \overset{R}{\leftarrow} Dual(\mathbb{Z}_p^n)$, let $m < n$ be a fixed number, and A be an invertible $m \times m$ matrix. We use S_m to denote a subset of $[n]$ satisfying $|S_m| = m$, let $\mathbb{B}_m$ be an $n \times m$ matrix consists of $\mathbf{b}_i$ for $i \in S_m$. Associated with S_m we define $\mathbb{B}_A$ as follows: for $i \notin S_m$, keep $\mathbf{b}_i$ unchanged; for $i \in S_m$, swap $\mathbf{b}_i$ for the corresponding column in $\mathbb{B}_m A$. We get $\mathbb{B}_A^*$ in a similar way except that for $i \in S_m$, we swap $\mathbf{b}_i^*$ for the corresponding column in $\mathbb{B}_m^*(A^{-1})^T$. From Lemma 1 in [10] we get that $(\mathbb{B}_A, \mathbb{B}_A^*)$ is distributed as random dual orthonormal bases as long as $(\mathbb{B}, \mathbb{B}^*)$ is randomly chosen. Especially, the pair $(\mathbb{B}_A, \mathbb{B}_A^*)$ information-theoretically hides A.

3.2 Complexity Assumptions

Decisional Linear Assumption (DLIN). To formally define our assumption, we let $\mathcal{G}$ denote a group generation algorithm, which takes in a security parameter λ and outputs a symmetric bilinear map e together with G, G_T of order p.

Let $\mathcal{G}$ be a group generator, run $\mathcal{G}(1^\lambda)$ to get (p, G, G_T, e), and randomly choose $g, f, v \in G, c_1, c_2, w \in \mathbb{Z}_p$, $T_0 = g^{c_1+c_2}, T_1 = g^w$. The advantage of $\mathcal{A}$ is defined as

$$Adv_{\mathcal{A}}^{DLIN} = \Big|Pr[\mathcal{A}(g, f, v, f^{c_1}, v^{c_2}, T_1) = 1] - Pr[\mathcal{A}(g, f, v, f^{c_1}, v^{c_2}, T_0) = 1]\Big|.$$

Definition 3 (DLIN). *We say that $\mathcal{G}$ satisfies DLIN assumption if for all PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{DLIN}$ is negligible in λ.*

Next we describe the subspace assumption in [10], here we require that $k \leq \frac{n}{3}$.
Subspace Assumption. Let $\mathcal{G}$ be a group generator algorithm as above, run $\mathcal{G}(1^\lambda)$ to get (p, G, G_T, e), and randomly choose

$$(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} Dual(\mathbb{Z}_p^n), g \xleftarrow{R} G,$$

$$\eta, \beta, \mu_1, \mu_2, \mu_3, \tau_1, \tau_2, \tau_3 \xleftarrow{R} \mathbb{Z}_p,$$

$$U_1 := g^{\mu_1 \mathbf{b}_1 + \mu_2 \mathbf{b}_{k+1} + \mu_3 \mathbf{b}_{2k+1}}, ..., U_k := g^{\mu_1 \mathbf{b}_k + \mu_2 \mathbf{b}_{2k} + \mu_3 \mathbf{b}_{3k}},$$

$$V_1 := g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_{k+1}^*}, ..., V_k := g^{\tau_1 \eta \mathbf{b}_k^* + \tau_2 \beta \mathbf{b}_{2k}^*},$$

$$W_1 := g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_{k+1}^* + \tau_3 \mathbf{b}_{2k+1}^*}, ..., W_k := g^{\tau_1 \eta \mathbf{b}_k^* + \tau_2 \beta \mathbf{b}_{2k}^* + \tau_3 \mathbf{b}_{3k}^*},$$

$$D := (g^{\mathbf{b}_1}, g^{\mathbf{b}_2}, ..., g^{\mathbf{b}_{2k}}, g^{\mathbf{b}_{3k+1}}, ..., g^{\mathbf{b}_n}, g^{\eta \mathbf{b}_1^*}, ..., g^{\eta \mathbf{b}_k^*},$$

$$g^{\beta \mathbf{b}_{k+1}^*}, ..., g^{\beta \mathbf{b}_{2k}^*}, g^{\mathbf{b}_{2k+1}^*}, ..., g^{\mathbf{b}_n^*}, U_1, ..., U_k, \mu_3)$$

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

$$Adv_{\mathcal{A}}^{SA} = \Big|Pr[\mathcal{A}(D, V_1, ..., V_k) = 1] - Pr[\mathcal{A}(D, W_1, ..., W_k) = 1]\Big|.$$

In this paper we use subspace assumption with $n = 6$ and $k = 2$ or $k = 1$.

Definition 4. *We say that $\mathcal{G}$ satisfies the subspace assumption if for all PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{SA}$ is negligible in λ.*

It was shown in [10] that the subspace assumption can be reduced to DLIN assumption.

Lemma 1. *[10] If there is an adversary $\mathcal{A}$ that can break the subspace assumption with probability ϵ, then we can build an algorithm $\mathcal{B}$ having the same advantage in solving the DLIN problem.*

4 An IBE Scheme with Better Message-Ciphertext Rate

4.1 Our Construction

In this section we describe our construction of IBE scheme. The structure of our scheme is similar to that in [10], however, by using both random parameters in the ciphertext to hide messages, we can get an IBE scheme that can encrypt two elements from G_T. Also we note that in Lewko's scheme they used θ, σ along

with $\mathbf{d}_i^*, i = 1, ..., 4$, but these two parameters is useless in the proof, so we avoid using θ, σ in our scheme. This modification does not decrease the efficiency, and results in a more elegant structure.

Here we assume messages are from the target group G_T^2 and identities are from $\mathbb{Z}_p$.

$Setup(1^\lambda)$: The setup algorithm runs $\mathcal{G}(1^\lambda)$ to obtain (p, G, G_T, e). It samples random dual orthonormal basis $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{R} Dual(\mathbb{Z}_p^6)$, chooses random $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ and computes $\Omega_1 = e(g,g)^{\alpha_1 \mathbf{d}_1 \cdot \mathbf{d}_1^*}$ and $\Omega_2 = e(g,g)^{\alpha_2 \mathbf{d}_1 \cdot \mathbf{d}_1^*}$. The public parameters are: $PP = (G, p, \Omega_1, \Omega_2, g^{\mathbf{d}_1}, g^{\mathbf{d}_2}, g^{\mathbf{d}_3}, g^{\mathbf{d}_4})$.
The master secret key is $Msk = (g^{\mathbf{d}_1^*}, g^{\alpha_1 \mathbf{d}_1^*}, g^{\mathbf{d}_2^*}, g^{\mathbf{d}_3^*}, g^{\alpha_2 \mathbf{d}_3^*}, g^{\mathbf{d}_4^*})$.

$Keygen(Msk, ID)$: The key generation algorithm chooses random $r_1, r_2, r_3, r_4 \in \mathbb{Z}_p$ and sets the secret key Sk_{ID} as:

$$K_1 = g^{(\alpha_1 + r_1 ID)\mathbf{d}_1^* - r_1 \mathbf{d}_2^* + r_2 ID \mathbf{d}_3^* - r_2 \mathbf{d}_4^*},$$

$$K_2 = g^{r_3 ID \mathbf{d}_1^* - r_3 \mathbf{d}_2^* + (\alpha_2 + r_4 ID)\mathbf{d}_3^* - r_4 \mathbf{d}_4^*}.$$

$Encrypt(PP, M_1 \| M_2, ID)$: The encryption algorithm chooses random $s_1, s_2 \in \mathbb{Z}_p$ and computes the ciphertext C as:

$$C_1 = M_1 \Omega_1^{s_1}, C_2 = M_2 \Omega_2^{s_2}, C_3 = g^{s_1 \mathbf{d}_1 + s_1 ID \mathbf{d}_2 + s_2 \mathbf{d}_3 + s_2 ID \mathbf{d}_4}.$$

$Decrypt(C, Sk_{ID})$: The decryption algorithm computes the message as:

$$M_1 = C_1 / e_n(C_3, K_1), M_2 = C_2 / e_n(C_3, K_2).$$

Correctness can be easily verified since $e(C_3, K_1) = \Omega_1^{s_1}, e(C_3, K_2) = \Omega_2^{s_2}$. Here $\mathbf{d}_5$ and $\mathbf{d}_6$ are used in the proof to form semi-functional ciphertexts and keys.

4.2 Security Proof

In the security proof of our IBE scheme, we use *semi-functional ciphertexts* and *semi-functional keys*, which are widely used in previous literatures [12,17,10,15].

Semi-functional Ciphertexts. To create a semi-functional ciphertext, we first choose random $s_1, s_2, , z_1, z_2 \in \mathbb{Z}_p$ and set

$$C_1 = M_1 \Omega_1^{s_1}, C_2 = M_2 \Omega_2^{s_2}, C_3 = g^{s_1 \mathbf{d}_1 + s_1 ID \mathbf{d}_2 + s_2 \mathbf{d}_3 + s_2 ID \mathbf{d}_4 + z_1 \mathbf{d}_5 + z_2 \mathbf{d}_6}.$$

Semi-functional Keys. To create a semi-functional key, we first choose random $r_1, r_2, r_3, r_4, t_1, t_2, t_3, t_4 \in \mathbb{Z}_p$ and set:

$$K_1 = g^{(\alpha_1 + r_1 ID)\mathbf{d}_1^* - r_1 \mathbf{d}_2^* + r_2 ID \mathbf{d}_3^* - r_2 \mathbf{d}_4^* + t_1 \mathbf{d}_5^* + t_2 \mathbf{d}_6^*},$$

$$K_2 = g^{r_3 ID \mathbf{d}_1^* - r_3 \mathbf{d}_2^* + (\alpha_2 + r_4 ID)\mathbf{d}_3^* - r_4 \mathbf{d}_4^* + t_3 \mathbf{d}_5^* + t_4 \mathbf{d}_6^*}.$$

We can see that a normal ciphertext can be correctly decrypted by a semi-functional key, and a semi-functional ciphertext can be correctly decrypted by a normal key, but when a semi-functional key is used to decrypt a semi-functional ciphertext, we will get the blinding factor multiplied by the additional term $e(g,g)^{\mathbf{d}_1 \cdot \mathbf{d}_1^* (t_1 z_1 + t_2 z_2)}$ and $e(g,g)^{\mathbf{d}_1 \cdot \mathbf{d}_1^* (t_3 z_1 + t_4 z_2)}$.

Theorem 1. *If DLIN assumption holds, then our IBE scheme is fully secure.*

Proof. To prove the security of our scheme, we define a sequence of games that any PPT adversary can not tell the difference between two adjacent games. Let q denote the number of key extraction queries that the adversary makes during the whole game.

$Game_{Real}$: the real fully security game.
$Game_0$: the same as $Game_{Real}$ except that the challenge ciphertext is semi-functional.
$Game_j$: for j from 1 to q, $Game_j$ is like $Game_0$ except that the first j key extraction queries are answered with semi-functional keys. The rest of the keys are normally generated.
$Game_{Final}$: the same as $Game_q$, except that the challenge ciphertext is a semi-functional encryption of a random message.

Let $Adv_{\mathcal{A}}^{Real}$ denote $\mathcal{A}$'s advantage in $Game_{Real}$, $Adv_{\mathcal{A}}^{i}$ denote $\mathcal{A}$'s advantage in $Game_i$ and $Adv_{\mathcal{A}}^{Final}$ denote $\mathcal{A}$'s advantage in $Game_{Final}$. It is clear that $Adv_{\mathcal{A}}^{Final} = 0$.

Lemma 2. *Suppose that there exists a PPT adversary $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{Real} - Adv_{\mathcal{A}}^{0} = \epsilon$, then there exists a PPT adversary $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption, with $n = 6$ and $k = 2$.*

Proof. $\mathcal{B}$ receives

$$D = (g^{\mathbf{b}_1}, ..., g^{\mathbf{b}_4}, g^{\eta\mathbf{b}_1^*}, g^{\eta\mathbf{b}_2^*}, g^{\beta\mathbf{b}_3^*}, g^{\beta\mathbf{b}_4^*}, g^{\mathbf{b}_5^*}, g^{\mathbf{b}_6^*}, U_1, U_2, \mu_3),$$

along with T_1, T_2, and its task is to decide whether T_1, T_2 is independent of $g^{\mathbf{b}_5^*}, g^{\mathbf{b}_6^*}$. $\mathcal{B}$ picks random invertible matrix $A \in \mathbb{Z}_p^{2\times 2}$. We define dual orthonormal bases $\mathbb{F}, \mathbb{F}^*$ as follows:

$$\mathbf{f}_1 = \eta\mathbf{b}_1^*, \mathbf{f}_2 = \eta\mathbf{b}_2^*, \mathbf{f}_3 = \beta\mathbf{b}_3^*, \mathbf{f}_4 = \beta\mathbf{b}_4^*, \mathbf{f}_5 = \mathbf{b}_5^*, \mathbf{f}_6 = \mathbf{b}_6^*,$$

$$\mathbf{f}_1^* = \eta^{-1}\mathbf{b}_1, \mathbf{f}_2^* = \eta^{-1}\mathbf{b}_2, \mathbf{f}_3^* = \beta^{-1}\mathbf{b}_3, \mathbf{f}_4^* = \beta^{-1}\mathbf{b}_4, \mathbf{f}_5^* = \mathbf{b}_5, \mathbf{f}_6^* = \mathbf{b}_6.$$

Then $\mathcal{B}$ implicitly sets $\mathbb{D} = \mathbb{F}_A, \mathbb{D}^* = \mathbb{F}_A^*$, that is, for $i = 1, 2, 3, 4, \mathbf{d}_i = \mathbf{f}_i, \mathbf{d}_i^* = \mathbf{f}_i^*, (\mathbf{d}_5, \mathbf{d}_6) = (\mathbf{f}_5, \mathbf{f}_6)A, (\mathbf{d}_5^*, \mathbf{d}_6^*) = (\mathbf{f}_5^*, \mathbf{f}_6^*)(A^{-1})^T$. Following from Lemma 1 in [10], we get that $(\mathbb{D}, \mathbb{D}^*)$ is randomly distributed and reveals no information about A.

Next $\mathcal{B}$ chooses random α_1', α_2' and implicitly sets $\alpha_1 = \eta\alpha_1', \alpha_2 = \beta\alpha_2'$, then $\mathcal{B}$ can compute $e(g,g)^{\alpha_1\mathbf{d}_1\cdot\mathbf{d}_1^*} = e_n(g^{\mathbf{b}_1}, g^{\eta\mathbf{b}_1^*})^{\alpha_1'}, e(g,g)^{\alpha_2\mathbf{d}_1\cdot\mathbf{d}_1^*} = e_n(g^{\mathbf{b}_3}, g^{\beta\mathbf{b}_3^*})^{\alpha_2'}$. Thus $\mathcal{B}$ sets up the public parameters and sends them to $\mathcal{A}$. $\mathcal{B}$ only knows the $g^{\alpha_1\mathbf{d}_1^*}, g^{\alpha_2\mathbf{d}_3^*}$ part of the master secret key and $g^{\eta\mathbf{d}_1^*}, g^{\eta\mathbf{d}_2^*}, g^{\beta\mathbf{d}_3^*}, g^{\beta\mathbf{d}_4^*}$. When $\mathcal{A}$ submits a key extraction query ID, $\mathcal{B}$ first chooses r_1', r_2', r_3', r_4' and implicitly sets $r_1 = \eta r_1', r_2 = \beta r_2', r_3 = \eta r_3', r_4 = \beta r_4'$. $\mathcal{B}$ sets the secret key as:

$$K_1 = g^{(\alpha_1' + r_1' ID)\eta\mathbf{d}_1^* - r_1'\eta\mathbf{d}_2^* + r_2' ID\beta\mathbf{d}_3^* - r_2'\beta\mathbf{d}_4^*},$$

$$K_2 = g^{r'_3 ID\eta \mathbf{d}^*_1 - r'_3\eta \mathbf{d}^*_2 + (\alpha'_2 + r'_4 ID)\beta \mathbf{d}^*_3 - r'_4\beta \mathbf{d}^*_4}.$$

At some point, $\mathcal{A}$ sends $\mathcal{B}$ the challenge identity ID^* and $(\mathbf{M_0}, \mathbf{M_1})$. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$ and computes the challenge ciphertext as follows:

$$C_1 = M_{b,1} e_n(T_1, g^{\mathbf{b_1}})^{\alpha'_1}, C_2 = M_{b,2} e_n(T_1, g^{\mathbf{b_3}})^{\alpha'_2}, C_3 = T_1 (T_2)^{ID^*}.$$

and gives the answer to $\mathcal{A}$.

If $(T_1, T_2) = (V_1, V_2)$, then the respond is a normal ciphertext with $s_1 = \tau_1, s_2 = \tau_2$. If $(T_1, T_2) = (W_1, W_2)$, then the respond is a semi-functional ciphertext with $s_1 = \tau_1, s_2 = \tau_2$, and $(z_1, z_2)^T = \tau_3 A^{-1}(1, ID^*)^T$. Thus when $(T_1, T_2) = (V_1, V_2)$ we properly simulate $Game_{Real}$ and when $(T_1, T_2) = (W_1, W_2)$ we simulate $Game_0$. So $\mathcal{B}$ can leverage $\mathcal{A}$'s advantage in distinguishing $Game_{Real}$ and $Game_0$ to achieve the same advantage against the subspace assumption. □

Lemma 3. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{j-1}_{\mathcal{A}} - Adv^{j}_{\mathcal{A}} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $n = 6$ and $k = 2$.*

To prove the lemma, we define $Game'_j$ as an intermediate game and let $Adv^{j'}_{\mathcal{A}}$ be $\mathcal{A}$'s advantage in $Game'_j$.

$Game'_j$: for j from 1 to q, $Game'_j$ is like $Game_{j-1}$ except that the j-th key query is answered by a temporary semi-functional key. A temporary semi-functional key is generated as follows: we first choose random $r_1, r_2, r_3, r_4, t_1, t_2 \in \mathbb{Z}_p$ and set:

$$K_1 = g^{(\alpha_1 + r_1 ID)\mathbf{d}^*_1 - r_1 \mathbf{d}^*_2 + r_2 ID \mathbf{d}^*_3 - r_2 \mathbf{d}^*_4 + t_1 \mathbf{d}^*_5 + t_2 \mathbf{d}^*_6},$$

$$K_2 = g^{r_3 ID \mathbf{d}^*_1 - r_3 \mathbf{d}^*_2 + (\alpha_2 + r_4 ID)\mathbf{d}^*_3 - r_4 \mathbf{d}^*_4}.$$

Note that half part of the temporary semi-functional key is generated like semi-functional keys, and the other half is like normal keys. Here we change the extracted key to semi-functional in 2 steps to make r_1, r_2, r_3, r_4 randomly distributed.

Lemma 4. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{j-1}_{\mathcal{A}} - Adv^{j'}_{\mathcal{A}} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $n = 6$ and $k = 2$.*

Proof. $\mathcal{B}$ receives

$$D = (g^{\mathbf{b}_1}, ..., g^{\mathbf{b}_4}, g^{\eta \mathbf{b}^*_1}, g^{\eta \mathbf{b}^*_2}, g^{\beta \mathbf{b}^*_3}, g^{\beta \mathbf{b}^*_4}, g^{\mathbf{b}^*_5}, g^{\mathbf{b}^*_6}, U_1, U_2, \mu_3),$$

along with T_1, T_2, and its task is to decide whether T_1, T_2 is independent of $g^{\mathbf{b}^*_5}, g^{\mathbf{b}^*_6}$. $\mathcal{B}$ picks random invertible matrix $A \in \mathbb{Z}_p^{2\times 2}$. We implicitly set dual orthonormal bases $\mathbb{D} = \mathbb{B}_A, \mathbb{D}^* = \mathbb{B}^*_A$, that is, for $i = 1, 2, 3, 4, \mathbf{d}_i = \mathbf{b}_i, \mathbf{d}^*_i = \mathbf{b}^*_i, (\mathbf{d}_5, \mathbf{d}_6) = (\mathbf{b}_5, \mathbf{b}_6)A, (\mathbf{d}^*_5, \mathbf{d}^*_6) = (\mathbf{b}^*_5, \mathbf{b}^*_6)(A^{-1})^T$.

Next $\mathcal{B}$ chooses random α'_1, α'_2 and implicitly sets $\alpha_1 = \eta\alpha'_1, \alpha_2 = \beta\alpha'_2$, then $\mathcal{B}$ can compute $e(g,g)^{\alpha_1 \mathbf{d}_1 \cdot \mathbf{d}^*_1} = e_n(g^{\mathbf{b}_1}, g^{\eta \mathbf{b}^*_1})^{\alpha'_1}, e(g,g)^{\alpha_2 \mathbf{d}_1 \cdot \mathbf{d}^*_1} = e_n(g^{\mathbf{b}_3}, g^{\beta \mathbf{b}^*_3})^{\alpha'_2}$, so $\mathcal{B}$ sets up the public parameters and sends them to $\mathcal{A}$.

When $\mathcal{A}$ submits key extraction queries:

For $i < j$, $\mathcal{B}$ can answer it since $\mathcal{B}$ can create normal keys as in Lemma 2 and knows $\mathbf{d}_5^*, \mathbf{d}_6^*$.
For $i > j$, $\mathcal{B}$ can answer it as in Lemma 2.
For $i = j$, $\mathcal{B}$ first chooses random $r_3', r_4' \in \mathbb{Z}_p$ and sets:

$$K_1 = g^{\alpha_1'\eta\mathbf{d}_1^*}(T_1^{ID_j})T_2^{-1}, K_2 = g^{r_3'ID_j\eta\mathbf{d}_1^* - r_3'\eta\mathbf{d}_2^* + (\alpha_2' + r_4'ID_j)\beta\mathbf{d}_3^* - r_4'\beta\mathbf{d}_4^*}.$$

At some point, $\mathcal{A}$ sends $\mathcal{B}$ the challenge identity ID^* and $(\mathbf{M_0}, \mathbf{M_1})$. $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$ and computes the challenge ciphertext as follows:

$$C_1 = M_{b,1}e_n(U_1, g^{\eta\mathbf{b_1}^*})^{\alpha_1'}, C_2 = M_{b,2}e_n(U_1, g^{\beta\mathbf{b_3}^*})^{\alpha_2'}, C_3 = U_1(U_2)^{ID^*}.$$

and gives the answer to $\mathcal{A}$. Here we implicitly set $s_1 = \mu_1, s_2 = \mu_2, (z_1, z_2)^T = \mu_3 A^{-1}(1, ID^*)^T$.

When $(T_1, T_2) = (V_1, V_2)$ we properly simulate $Game_{j-1}$ and when $(T_1, T_2) = (W_1, W_2)$ we simulate $Game_{j'}$. So $\mathcal{B}$ can leverage $\mathcal{A}$'s advantage in distinguishing $Game_{j-1}$ and $Game_{j'}$ to achieve the same advantage against the subspace assumption. □

Lemma 5. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{j'} - Adv_{\mathcal{A}}^{j} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $n = 6$ and $k = 2$.*

Proof. The proof of this lemma is similar to that of Lemma 4.

Lemma 6. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{q} - Adv_{\mathcal{A}}^{Final} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $n = 6$ and $k = 1$.*

In order to prove this lemma, we define $Game_{q_a}, Game_{q_b}, Game_{q_c}$ as intermediate games:

$Game_{q_a}$: It is exactly like $Game_q$ except that in the C_3^* part of the challenge ciphertext, the coefficient of $\mathbf{d}_2$ is changed to a random value in $\mathbb{Z}_p$ instead of $s_1 ID^*$.
$Game_{q_b}$: It is exactly like $Game_{q_a}$ except that in the C_3^* part of the challenge ciphertext is independent of s_1.
$Game_{q_c}$: It is exactly like $Game_{q_b}$ except that in the C_3^* part of the challenge ciphertext, the coefficient of $\mathbf{d}_4$ is changed to a random value in $\mathbb{Z}_p$ instead of $s_2 ID^*$.

We denote the advantage in these games as $Adv_{\mathcal{A}}^{q_a}, Adv_{\mathcal{A}}^{q_b}, Adv_{\mathcal{A}}^{q_c}$.

Lemma 7. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{q} - Adv_{\mathcal{A}}^{q_a} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $n = 6$ and $k = 1$.*

Lemma 8. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{q_a} - Adv_{\mathcal{A}}^{q_b} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $n = 6$ and $k = 1$.*

Lemma 9. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{q_b} - Adv_{\mathcal{A}}^{q_c} = \epsilon$, then we can construct a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $n = 6$ and $k = 1$.*

Lemma 10. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{q_c} - Adv_{\mathcal{A}}^{Final} = \epsilon$, then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking the subspace assumption with $k = 1$ and $n = 6$.*

The proof of the above lemmata is similar to that of lemmata in [10] and readers can refer to our full version for details.

The previous lemmata show that the real security game is indistinguishable from $Game_{Final}$, in which the value of b is information-theoretically hidden from the attacker, hence the attacker can only get negligible advantage in breaking the security of our IBE scheme.

4.3 Anonymity

We note that in the final game, the challenge ciphertext is independent of the challenge identity, so our scheme is anonymous. Lewko's IBE scheme is anonymous for the same reason. Anonymous IBE is a useful component to construct public key encryption with keyword search (PEKS) schemes.

5 Conclusion

In this paper, we improve the IBE scheme presented by Lewko in [10], and get a fully secure anonymous IBE scheme in the prime order setting that has a better message-ciphertext rate. Similar to Lewko's scheme, we use dual pairing vector space in prime order bilinear groups to realize the canceling and parameter hiding property. The security of our scheme is based on the subspace assumption, which can be reduced to the decisional linear assumption. We use the dual system encryption in the security proof.

Acknowledgments. We are very grateful to anonymous reviewers for their helpful comments. We also thank Xianhui Lu and Yu Chen for helpful discussions.

References

1. Abdalla, M., et al.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
3. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

4. Boneh, D., Sahai, A., Waters, B.: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
5. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
6. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext Security from Identity-based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
7. Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
8. Freeman, D.M.: Converting Pairing-based Cryptosystems from Composite-order Groups to Prime-order Groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
9. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive Zaps and New Techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
10. Lewko, A.: Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
11. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute based Encryption and (Hierarichical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
12. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
13. Okamoto, T., Takashima, K.: Homomorphic Encryption and Signatures from Vector Decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008)
14. Okamoto, T., Takashima, K.: Hierarchical Predicate Encryption for Inner Products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
15. Ramanna, S.C., Chatterjee, S., Sarkar, P.: Variants of Waters' Dual System Primitives Using Asymmetric Pairings. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 298–315. Springer, Heidelberg (2012)
16. Shamir, A.: Identity-based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) Advances in Cryptology - CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
17. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE Under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)

Accountable Trapdoor Sanitizable Signatures

Junzuo Lai[1,2], Xuhua Ding[1], and Yongdong Wu[3]

[1] School of Information Systems, Singapore Management University, Singapore
{junzuolai,xhding}@smu.edu.sg
[2] Department of Computer Science, Jinan University, China
[3] Institute for Infocomm Research, Singapore
wydong@i2r.a-star.edu.sg

Abstract. Sanitizable signature (SS) allows a signer to partly delegate signing rights to *a predetermined* party, called sanitizer, who can later modify certain designated parts of a message originally signed by the signer and generate a new signature on the sanitized message without interacting with the signer. One of the important security requirements of sanitizable signatures is *accountability*, which allows the signer to prove, in case of dispute, to a third party that a message was modified by the sanitizer. Trapdoor sanitizable signature (TSS) enables a signer of a message to delegate the power of sanitization to *any parties at anytime* but at the expense of losing the accountability property. In this paper, we introduce the notion of accountable trapdoor sanitizable signature (ATSS) which lies between SS and TSS. As a building block for constructing ATSS, we also introduce the notion of accountable chameleon hash (ACH), which is an extension of chameleon hash (CH) and might be of independent interest. We propose a concrete construction of ACH and show how to use it to construct an ATSS scheme.

Keywords: Trapdoor Sanitizable Signature, Accountability, Chameleon Hash.

1 Introduction

Ateniese et al. [1] introduced the notion of sanitizable signature (SS) and presented a generic construction based on chameleon hash (CH) [18]. Sanitizable signatures allow a signer to partly delegate signing rights to a predetermined party, called a sanitizer. During signature generation on a message, the signer chooses a specific sanitizer who, with the knowledge of the putative signature, can later modify certain designated parts of the message and generate a new signature on the sanitized message without interacting with the signer. The capability of modification renders sanitizable signatures useful in many applications, such as authenticated multicast, authenticated database outsourcing and authenticated multimedia content distribution.

Sanitizable signatures are required to possess the following five security properties [1]:

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 117–131, 2013.

Unforgeability. An outsider (i.e., neither the signer nor the sanitizer) should not be able to forge the signer's or the sanitizer's signature.
Immutability. The sanitizer should not be able to produce valid signatures for messages where it has modified other than the designated parts.
Privacy. Sanitized messages and their signatures should not reveal the original data.
Transparency. An outsider should not be able to decide whether a message has been sanitized or not.
Accountability. In case of a dispute, the signer can prove to a trusted third party that a certain message was modified by the sanitizer.

Subsequently Canard et al. [8] introduced the notion of trapdoor sanitizable signature (TSS), which is an extension of SS. TSS enables a signer to delegate the power of sanitization for a signed message to any party. They also proposed a generic construction of TSS based on identity-based chameleon hash (IBCH) [2]. However, TSS does not satisfy the security requirement of accountability.

In this paper, we introduce the notion of accountable TSS (ATSS) which lies between TSS and SS. Like SS, our ATSS scheme satisfies the accountability property; it allows a signer to generate an ATSS signature on a message for a predetermined candidate sanitizer. The ATSS signature alone does not provide the candidate sanitizer the power to modify the underlying message and generate a valid signature on the modified message. In order to generate a valid signature on a modified message, similar to TSS, the candidate sanitizer needs to obtain a trapdoor from the signer, in addition to the signer's signature.

One possible application of ATSS is content authentication in tiered multimedia distribution systems [12]. Such a system consists of a top-tier primary content provider and a number of lower-tier affiliating providers each with its own users. An example is a movie distributer in Hollywood that has a number of country distributers worldwide. Whenever a new movie is released, the Hollywood distributer signs the video stream for a country distributor using ATSS and sends the video stream and the signature to the country distributor for previewing. Upon receiving the video stream, the country distributor verifies the authenticity of the movie using the signature. If the country distributor is interested in distributing the movie, it enters a contract with or make a payment to the Hollywood distributor. The latter in turn sends a trapdoor to the former. With the knowledge of the trapdoor, the country distributor can then modify/adapt the movie for its local market (e. g., adding subtitles in the local language) and generate a valid signature on the modified video stream.

1.1 Our Contribution

Contributions of the paper can be summarized as follows:

1. We introduce the notion of ATSS. In an ATSS scheme, a signer needs to predetermine a user as a candidate sanitizer during the signature generation. This signature alone does not allow the candidate sanitizer to produce a new signature on a sanitized message. To generate a new signature on a sanitized message, the candidate sanitizer needs to obtain a trapdoor from the original signer.

2. We extend the notion of CH by introducing the notion of accountable CH (ACH) and define its security requirements. We propose a concrete construction of ACH that satisfies the security requirements in the random oracle model [6].
3. Based on ACH, we present a generic construction of ATSS. Instantiating the generic construction with our concrete ACH scheme, we can obtain the first ATSS scheme.

1.2 Related Work

Sanitizable Signature. The notion of SS was introduced by Ateniese et al. [1]. Such signatures allow a sanitizer to modify certain designated parts of a signed message and generate a new signature on the sanitized message without interacting with the signer. Klonowski and Lauks [17] presented several extensions of SS, including limitation of the set of possible modifications of a single mutable block and limitation of the number of modifications of mutable blocks. Pöhls et al. [22] integrated SS schemes into the XML signature specification.

Ateniese et al. [1] identified five security requirements of SS schemes, *unforgeability*, *immutability*, *privacy*, *transparency* and *accountability*. Brzuska et al. [7] revisited these security requirements and investigated their relationships, showing for example that transparency implies privacy.

Miyazaki et al. [21] used the notion of SS in a slightly different vein. Their SS schemes [21,15,20] allow a sanitizer to only *delete* predetermined parts of a signed message.

The notion of incremental cryptography [5] and homomorphic signatures, which encompass transitive [19], redactable [16] and context-extraction signatures [24], are also related to SS. We refer the reader to [1] for details.

Trapdoor Sanitizable Signature. Canard et al. [8] introduced the notion of TSS, in which the power of sanitization is given to possibly several entities. Based on IBCH, Canard et al. [8] proposed a generic construction of TSS. Recently, Yum et al. [25] presented another generic construction of TSS from ordinary signature schemes; therefore, one-way functions imply TSS. Bao et al. [4] extended TSS for the hierarchical setting.

Chameleon Hash. Our work is also related to CH functions, which are randomized collision-resistant hash functions with the additional property that given a trapdoor, one can efficiently generate collisions. CH was first introduced by Krawczyk and Rabin [18]. Other CH constructions were proposed subsequently [9,3,14,13,11] .

Ateniese and Medeiros [2] extended CH to identity-based setting [23] and introduced the notion of IBCH. Zhang et al. [26] and Chen et al. [10] followed their work.

1.3 Organization

The rest of the paper is organized as follows. Some preliminaries are given in Section 2. The notion and security requirements of ATSS are introduced in Section 3. In Section 4, we present the notion and security requirements of ACH, and propose a concrete construction. In Section 5, we propose a generic construction of ATSS from ACH and present a specific ATSS scheme based on our ACH scheme. Finally, we state our conclusion in Section 6.

2 Preliminaries

If L is a positive integer, then $[1, L] = \{1, 2, \ldots, L\}$. If A, B are two sets, $A \backslash B = \{x \in A | x \notin B\}$. If $x_1, x_2, \ldots$ are strings, then $x_1 \| x_2 \| \ldots$ denotes their concatenation. We denote by $\mathcal{R}$ the range of random number. We say that a function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists an λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.

2.1 Bilinear Pairings

Let $\mathbb{G}$ be a cyclic multiplicative group of prime order p and $\mathbb{G}_T$ be a cyclic multiplicative group of the same order p. A bilinear pairing is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

- Bilinearity: $\forall g_1, g_2 \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p^*$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- Non-degeneracy: There exist $g_1, g_2 \in \mathbb{G}$ such that $e(g_1, g_2) \neq 1$;
- Computability: There exists an efficient algorithm to compute $e(g_1, g_2)$ for $\forall g_1, g_2 \in \mathbb{G}$.

2.2 Computational Diffie-Hellman (CDH) Assumption

The security of our ACH scheme will be reduced to the hardness of the computational Diffie-Hellman (CDH) problem in the bilinear map group system $\langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$ in which the ACH scheme is constructed.

Definition 1. *Given a bilinear map group system $\langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$, a generator g of $\mathbb{G}$ and elements $g^a, g^b \in \mathbb{G}$ where a, b are selected uniformly at random from $\mathbb{Z}_p^*$, the CDH problem in the bilinear map group system is to compute g^{ab}. We say that the CDH assumption holds in a bilinear map group system $\langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$ if no probabilistic polynomial-time algorithm can solve the CDH problem in the bilinear map group system with non-negligible probability.*

3 Accountable Trapdoor Sanitizable Signature and Its Security Requirements

ATSS lies between sanitizable signature and trapdoor sanitizable signature. Like a TSS scheme, an ATSS scheme includes the algorithms: GlobalSetup, KeyGen,

Sign, Trapdoor, Sanitize and Verify. However, in ATSS, besides the private key of the signer, the inputs of the Sign algorithm include the public key of a candidate sanitizer and a transaction identifer TID. In order to generate a new signature on a sanitized message, in ATSS, the inputs of the Sanitize algorithm include the private key of the candidate sanitizer and a trapdoor associated with the transaction identifier TID generated by the signer using Trapdoor algorithm, not just a trapdoor associated with the transaction identifier as in TSS or just the private key of the sanitizer as in SS.

In addition, to settle disputes about the origin of a message-signature pair, an algorithm Proof enables the signer to produce a proof π. The proof π is generated from a set of previously signed messages. A Judge algorithm then uses the proof π to decide if a valid message-signature pair was created by the signer or the sanitizer (the lack of such a proof is interpreted as a signer origin).

Concretely, an ATSS scheme is a tuple of algorithms described as follows:

GlobalSetup takes as input a security parameter λ. It produces a common public parameter $param$ to be used by all parties in the system.

KeyGen takes as input a security parameter λ and the common public parameter $param$. It generates a public/private key pair (pk, sk). Every party in the system uses this randomized algorithm to generate a private/public key pair himself or herself.

For presentation simplicity, we assume there exist a single singer and multiple sanitizers in the system. We denote by (pk_{sig}, sk_{sig}) the key pair of the signer, and by (pk_{san}, sk_{san}) the key pair of a sanitizer.

Sign takes as input a sanitizer's public key pk_{san}, a message $m = m_1 \| \cdots \| m_L$, a set of indices $I \subseteq [1, L]$ that are sanitizable, a transaction identifier TID and the signer's private key sk_{sig}. It outputs a signature σ on m.

We assume that each message signed has a unique transaction identifier.

Trapdoor takes as input a message m, a set of indices I that are sanitizable, a transaction identifier TID, a valid signature σ on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$ and the signer's private key sk_{sig}. It outputs a trapdoor td_{TID}.

Sanitize takes as input the signer's public key pk_{sig}, a message m, a set of the indices I that are sanitizable, a transaction identifier TID, a valid signature σ on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$, a trapdoor td_{TID} associated with TID, the sanitizer's private key sk_{san} and a new message m'. It outputs a new signature σ' on $(pk_{sig}, pk_{san}, m', I, \mathsf{TID})$.

Verify takes as input $param$, the signer's public key pk_{sig}, a sanitizer's public key pk_{san}, a message m, a set of the indices I that are sanitizable, a transaction identifier TID and a putative signature σ. It outputs 1 if the signature σ on m is valid and 0 otherwise.

Proof takes as input $param$, the signer's private key sk_{sig}, a valid signature σ on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$, and a set of (polynomially many) additional message-signature pairs $((pk_{sig}, pk_{san}^{(i)}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}), \sigma^{(i)})_{i=1,2,\ldots,q}$. It outputs a proof $\pi \in \{0,1\}^*$.

Judge takes as input $param$, the signer's public key pk_{sig}, the sanitizer's public key pk_{san}, a valid signature σ on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$ and a proof π. It

outputs a decision $d \in \{\texttt{Sig}/\texttt{San}\}$ indicating whether the message-signature pair $((pk_{sig}, pk_{san}, m, I, \mathsf{TID}), \sigma)$ was created by the signer or the sanitizer.

The usual correctness properties should hold for an ATSS scheme, saying that genuinely signed or sanitized messages are accepted. Formally, for any security parameter λ, any message $m = m_1 \| \cdots \| m_L$, any set of indices $I \subseteq [1, L]$, any transaction identifier TID, $param \leftarrow \mathsf{GlobalSetup}(\lambda)$, $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, $(pk_{san}, sk_{san}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, $\sigma \leftarrow \mathsf{Sign}(pk_{san}, m, I, \mathsf{TID}, sk_{sig})$, $td_{\mathsf{TID}} \leftarrow \mathsf{Trapdoor}(m, I, \mathsf{TID}, \sigma, sk_{sig})$, and $\sigma' \leftarrow \mathsf{Sanitize}(pk_{sig}, m, I, \mathsf{TID}, \sigma, m', td_{\mathsf{TID}}, sk_{san})$, we must have $\mathsf{Verify}(param, pk_{sig}, pk_{san}, m, I, \mathsf{TID}, \sigma) = 1$ and $\mathsf{Verify}(param, pk_{sig}, pk_{san}, m', I, \mathsf{TID}, \sigma') = 1$.

The security requirements of an ATSS scheme include *unforgeability*, *indistinguishability* and *accountability*. The *unforgeability* and *indistinguishability* requirements of ATSS are extended from the security requirements of TSS [8]. Informally, unforgeability requires that an outsider be not able to forge a signature on the original or the sanitized message, and indistinguishability requires that an outsider be not able to decide whether a message has been sanitized or not.

Accountability requires that the origin of a (sanitized) signature be undeniable. We distinguish between *sanitizer-* and *signer-accountability*, as did in [7]. Informally, *sanitizer-accountability* implies that if a message has not been signed by the signer, then even a malicious sanitizer should not be able to make a judge accuse the signer, and *signer-accountability* implies that if a signed message has not been sanitized, then even a malicious signer should not be able to make the judge accuse the sanitizer.

Unforgeability: An ATSS scheme is existentially unforgeable under adaptive chosen message attacks, if for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game between $\mathcal{A}$ and a challenger is negligible in the security parameter λ:

Setup. The challenger runs $param \leftarrow \mathsf{GlobalSetup}(\lambda)$, $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, $(pk_{san}, sk_{san}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, and sends the common public parameter $param$, the signer's public key pk_{sig} and the sanitizer's public key pk_{san} to the adversary $\mathcal{A}$.

Query Phase. The adversary $\mathcal{A}$ adaptively issues queries:

1. $\mathcal{O}_{ATSS}^{\mathsf{Sign}}$ query on (m, I, TID), where $I \subseteq [1, L]$ is a set of indices and TID is a transaction identifier: The challenger forwards the valid signature σ on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$ to the adversary.
2. $\mathcal{O}_{ATSS}^{\mathsf{Trapdoor}}$ query on $(m, I, \mathsf{TID}, \sigma)$, where σ is a valid signature on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$: The challenger forwards the corresponding trapdoor td_{TID} to the adversary.
3. $\mathcal{O}_{ATSS}^{\mathsf{Sanitize}}$ query on $(m, I, \mathsf{TID}, \sigma, m')$, where $m_i = m'_i$ for all $i \notin I$: The challenger forwards the new valid signature σ' on $(pk_{sig}, pk_{san}, m', I, \mathsf{TID})$ to the adversary.

Output. $\mathcal{A}$ outputs $(m^*, I^*, \mathsf{TID}^*, \sigma^*)$ and succeeds if the following conditions hold.

1. $\mathsf{Verify}(param, pk_{sig}, pk_{san}, m^*, I^*, \mathsf{TID}^*, \sigma^*) = 1$.
2. $\mathcal{A}$ never queries $\mathcal{O}_{ATSS}^{\mathsf{Sign}}$ oracle on $(m^*, I^*, \mathsf{TID}^*)$.
3. (m^*, σ^*) does not come from $\mathcal{O}_{ATSS}^{\mathsf{Sanitize}}$ oracle, i.e., $\mathcal{A}$ never queries $\mathcal{O}_{ATSS}^{\mathsf{Sanitize}}$ oracle on $(m, I^*, \mathsf{TID}^*, \sigma, m^*)$, where σ is a valid signature on $(pk_{sig}, pk_{san}, m, I^*, \mathsf{TID}^*)$ and $m_i = m_i^*$ for all $i \notin I^*$.
4. $\mathcal{A}$ never queries $\mathcal{O}_{ATSS}^{\mathsf{Trapdoor}}$ oracle on $(m, I^*, \mathsf{TID}^*, \sigma)$, where σ is a valid signature on $(pk_{sig}, pk_{san}, m, I^*, \mathsf{TID}^*)$ and $m_i = m_i^*$ for all $i \notin I^*$.

Indistinguishability: The indistinguishability of an ATSS scheme requires that the output distributions of Sign algorithm and $\mathsf{Sanitize}$ algorithm be computational indistinguishable. In other words, for all sufficiently large λ, any $param \leftarrow \mathsf{GlobalSetup}(\lambda)$, $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, $(pk_{san}, sk_{san}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, any set of indices $I \subseteq [1, L]$, any message pairs m, m' such that $m_i = m'_i$ for all $i \notin I$, any transaction identifier TID, the following distribution ensembles $\mathcal{D}_{\mathsf{Sanitize}}$ and $\mathcal{D}_{\mathsf{Sign}}$ are computational indistinguishable:

$$\begin{aligned}\mathcal{D}_{\mathsf{Sanitize}} &= \{(m', \hat{\sigma}) | \sigma \leftarrow \mathsf{Sign}(pk_{san}, m, I, \mathsf{TID}, sk_{sig}),\\ &\quad td_{\mathsf{TID}} \leftarrow \mathsf{Trapdoor}(m, I, \mathsf{TID}, \sigma, sk_{sig}),\\ &\quad \hat{\sigma} \leftarrow \mathsf{Sanitize}(pk_{sig}, m, I, \mathsf{TID}, \sigma, m', td_{\mathsf{TID}}, sk_{san})\}_{\lambda, param, pk_{sig}, pk_{san}, I, \mathsf{TID}},\\ \mathcal{D}_{\mathsf{Sign}} &= \{(m', \sigma') | \sigma' \leftarrow \mathsf{Sign}(pk_{san}, m', I, \mathsf{TID}, sk_{sig})\}_{\lambda, param, pk_{sig}, pk_{san}, I, \mathsf{TID}}.\end{aligned}$$

Sanitizer-accountability: An ATSS scheme is sanitizer-accountable, if for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game between $\mathcal{A}$ and a challenger is negligible in the security parameter λ:

Setup. The challenger runs $param \leftarrow \mathsf{GlobalSetup}(\lambda)$, $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, and sends the common public parameter $param$ and the signer's public key pk_{sig} to the adversary $\mathcal{A}$.

Query Phase. The adversary $\mathcal{A}$ adaptively issues queries:

1. $\mathcal{O}_{ATSS}^{\mathsf{Sign}}$ query on $(pk_{san}, m, I, \mathsf{TID})$, where pk_{san} is a sanitizer's public key chosen by $\mathcal{A}$, $I \subseteq [1, L]$ is a set of indices and TID is a transaction identifier: The challenger forwards the valid signature σ on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$ to the adversary.
2. $\mathcal{O}_{ATSS}^{\mathsf{Trapdoor}}$ query on $(pk_{san}, m, I, \mathsf{TID}, \sigma)$, where σ is a valid signature on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$: The challenger forwards the corresponding trapdoor td_{TID} to the adversary.

Output. $\mathcal{A}$ outputs $(pk_{sig}, pk_{san}^*, m^*, I^*, \mathsf{TID}^*, \sigma^*)$ and succeeds if the following conditions hold.

1. $\mathsf{Verify}(param, pk_{sig}, pk_{san}^*, m^*, I^*, \mathsf{TID}^*, \sigma^*) = 1$.
2. $((pk_{sig}, pk_{san}^*, m^*, I^*, \mathsf{TID}^*), \sigma^*) \neq ((pk_{sig}, pk_{san}^{(i)}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}), \sigma^{(i)})$ for all $i = 1, 2, \ldots, q$, where $(pk_{san}^{(i)}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)})$ and $\sigma^{(i)}$ for $i = 1, 2, \ldots, q$ denote the queries and answers to and from oracle $\mathcal{O}_{ATSS}^{\mathsf{Sign}}$.
3. $\mathsf{Sig} \leftarrow \mathsf{Judge}(param, ((pk_{sig}, pk_{san}^*, m^*, I^*, \mathsf{TID}^*), \sigma^*), \pi^*)$, where $\pi^* \leftarrow \mathsf{Proof}(param, sk_{sig}, ((pk_{sig}, pk_{san}^*, m^*, I^*, \mathsf{TID}^*), \sigma^*), ((pk_{sig}, pk_{san}^{(i)}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}), \sigma^{(i)})_{i=1,2,\ldots,q})$.

Signer-accountability: An ATSS scheme is signer-accountable, if for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game between $\mathcal{A}$ and a challenger is negligible in the security parameter λ:

Setup. The challenger runs $param \leftarrow \mathsf{GlobalSetup}(\lambda)$, $(pk_{san}, sk_{san}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, and sends the common public parameters $param$ and the sanitizer's public key pk_{san} to the adversary $\mathcal{A}$.

Query Phase. The adversary $\mathcal{A}$ adaptively issues $\mathcal{O}_{ATSS}^{\mathsf{Sanitize}}$ query on $(pk_{sig}, m, I, \mathsf{TID}, \sigma, td_{\mathsf{TID}}, m')$, where pk_{sig} is a singer's public key chosen by $\mathcal{A}$, σ is a valid signature on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$, td_{TID} is the trapdoor associated with TID and $m_i = m'_i$ for all $i \notin I$: The challenger forwards the new valid signature σ' on $(pk_{sig}, pk_{san}, m', I, \mathsf{TID})$ to the adversary.

Output. $\mathcal{A}$ outputs $(pk^*_{sig}, pk_{san}, m^*, I^*, \mathsf{TID}^*, \sigma^*, \pi^*)$ and succeeds if the following conditions hold.

1. $\mathsf{Verify}(param, pk^*_{sig}, pk_{san}, m^*, I^*, \mathsf{TID}^*, \sigma^*) = 1$.
2. $\mathsf{San} \leftarrow \mathsf{Judge}(param, ((pk^*_{sig}, pk_{san}, m^*, I^*, \mathsf{TID}^*), \sigma^*), \pi^*)$.
3. $((pk^*_{sig}, pk_{san}, m^*, I^*, \mathsf{TID}^*), \sigma^*) \neq ((pk^{(i)}_{sig}, pk_{san}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}), \sigma^{(i)})$ for all $i = 1, 2, \ldots, q$, where $((pk^{(i)}_{sig}, pk_{san}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}), \sigma^{(i)})$ for $i = 1, 2, \ldots, q$ denote the answers from oracle $\mathcal{O}_{ATSS}^{\mathsf{Sanitize}}$.

4 Accountable Chameleon Hash and Its Construction

In this section, we first introduce and formulate accountable chameleon hash (ACH). Then, we present a construction of ACH and analyze its security in the random oracle model.

4.1 Accountable Chameleon Hash

ACH is a new paradigm which lies between chameleon hash (CH) and identity-based chameleon hash (IBCH). The inputs of the Hash algorithm of an ACH include two users' public keys and a transaction identifier TID, not just the public key of a single user as in a CH scheme or just an identity as in an IBCH scheme. In order to find a collision, the inputs of the Forge algorithm of an ACH scheme include one user's private key and a trapdoor information associated with the transaction identifer TID generated by the other user, not just a single user's private key as in a CH scheme or just a trapdoor information associated with an identity as in an IBCH scheme.

Concretely, an ACH scheme consists of the following algorithms:

$\mathsf{GlobalSetup}$ takes as input a security parameter λ. It produces a common public parameter $param$ to be used by all parties in the system.

KeyGen takes as input a security parameter λ and the common public parameter $param$. It generates a public/private key pair (pk, sk). All parties in the system use this randomized algorithm to generate a private/public key pair himself or herself.

Hash takes as input $param$, user i's public key pk_i, user j's public key pk_j, a message m and a unique transaction identifier TID. It chooses a random r and outputs a hash value h.

Trapdoor takes as input user i's private key sk_i and a transaction identifier TID. It outputs the trapdoor information $td_{i,\mathsf{TID}}$ associated with user i and the transaction identifier TID.

Forge takes as input user j's private key sk_j, the trapdoor information $td_{i,\mathsf{TID}}$ associated with user i and a transaction identifier TID, the hash value h on a message m with user i's public key pk_i, user j's public key pk_j, the transaction identifier TID, random r, and a message m'. It outputs a random r'.

For correctness, we require that $\mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}, m, r) = h = \mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}, m', r')$ and $m' \neq m$, where $r' \leftarrow \mathsf{Forge}(sk_j, td_{i,\mathsf{TID}}, pk_i, pk_j, \mathsf{TID}, m, r, h, m')$, $td_{i,\mathsf{TID}} \leftarrow \mathsf{Trapdoor}(sk_i, \mathsf{TID})$. The security of an ACH scheme consists of two requirements: *resistance to collision forgery under active attacks* and *forgery indistinguishability.*

Resistance to collision forgery under active attacks: The accountable chameleon hash scheme is secure against (existential) collision forgery under active attacks if, for any probabilistic polynomial-time algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game between $\mathcal{A}$ and a challenger is negligible in the security parameter λ:

Setup. The challenger runs $param \leftarrow \mathsf{GlobalSetup}(\lambda)$, $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(\lambda, param)$, $(pk_j, sk_j) \leftarrow \mathsf{KeyGen}(\lambda, param)$, and sends the common public parameter $param$, user i's public key pk_i and user j's public/private key pair (pk_j, sk_j) to the adversary $\mathcal{A}$.

Query Phase. The adversary $\mathcal{A}$ adaptively issues queries $\mathcal{O}_{ACH}^{\mathsf{Trapdoor}}$ on a transaction identifier TID. The challenger forwards the trapdoor information $td_{i,\mathsf{TID}}$ associated with user i and the transaction identifier TID to the adversary.

Output. $\mathcal{A}$ outputs $(\mathsf{TID}^*, m, r, m', r')$ and succeeds if the following conditions hold.

1. $\mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}^*, m, r) = \mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}^*, m', r')$ and $m' \neq m$.
2. $\mathcal{A}$ never queries $\mathcal{O}_{ACH}^{\mathsf{Trapdoor}}$ on TID^*.

Forgery indistinguishability: An ACH scheme is said to be forgery indistinguishable if, for all sufficiently large λ, any $param \leftarrow \mathsf{GlobalSetup}$, $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(\lambda, param)$, $(pk_j, sk_j) \leftarrow \mathsf{KeyGen}(\lambda, param)$, all transaction identifier TID, and all pairs of messages m and m', the following distribution ensembles are computational indistinguishable:

$$\begin{aligned}\mathcal{D}_{\mathsf{Forge}} = \{&(m', \hat{r}, h) | r \xleftarrow{\$} \mathcal{R}, h \leftarrow \mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}, m, r),\\ &td_{i,\mathsf{TID}} \leftarrow \mathsf{Trapdoor}(sk_i, \mathsf{TID}),\\ &\hat{r} \leftarrow \mathsf{Forge}(sk_j, td_{i,\mathsf{TID}}, pk_i, pk_j, \mathsf{TID}, m, r, h, m')\}_{\lambda, param, pk_i, pk_j, \mathsf{TID}},\end{aligned}$$

$$\begin{aligned}\mathcal{D}_{\mathsf{Hash}} = \{&(m', r', h') | r' \xleftarrow{\$} \mathcal{R},\\ &h' \leftarrow \mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}, m', r')\}_{\lambda, param, pk_i, pk_j, \mathsf{TID}}.\end{aligned}$$

4.2 Construction

Our specific construction of an ACH scheme consists of the following algorithms:

- $\mathsf{GlobalSetup}$ Given a security parameter λ, it first generates a bilinear map group system $\langle p, \mathbb{G}, \mathbb{G}_T, e\rangle$. Then, it picks a generator g of $\mathbb{G}$ and chooses a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}$. The common public parameter is $param = (p, \mathbb{G}, \mathbb{G}_T, e, g, H)$.
- KeyGen Given a security parameter λ and the common public parameter $param$, user i first chooses $x_i \in \mathbb{Z}_p^*$ randomly. Then, set his public key as $pk_i = g^{x_i}$, and the private key as $sk_i = x_i$.
- Hash Given $param$, user i's public key pk_i, user j's public key pk_j, a message $m \in \mathbb{Z}_p^*$ and a unique transaction identifier TID, it chooses $R \in \mathbb{G}$ uniformly at random and computes $h = e(R, g) \cdot e(H(\mathsf{TID})^m, pk_i \cdot pk_j)$. Finally, it outputs the hash value h.
- $\mathsf{Trapdoor}$ Given user i's private key $sk_i = x_i$ and a transaction identifier TID, it computes $td_{i,\mathsf{TID}} = H(\mathsf{TID})^{sk_i} = H(\mathsf{TID})^{x_i}$, and outputs the trapdoor information $td_{i,\mathsf{TID}}$ associated with user i and transaction identifier TID.
- Forge Given user j's private key sk_j, the trapdoor information $td_{i,\mathsf{TID}}$ associated with user i and transaction identifier TID, the hash value h on a message m with user i's public key pk_i, user j's public key pk_j, transaction identifier TID, random R, and a message m', it computes and outputs $R' = R \cdot (H(\mathsf{TID})^{sk_j} \cdot td_{i,\mathsf{TID}})^{m-m'}$.

Note that, for a forgery, we have

$$\begin{aligned}
&\mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}, m', R') \\
&= e(R', g) \cdot e(H(\mathsf{TID})^{m'}, pk_i \cdot pk_j) \\
&= e(R \cdot (H(\mathsf{TID})^{sk_j} \cdot td_{i,\mathsf{TID}})^{m-m'}, g) \cdot e(H(\mathsf{TID})^{m'}, pk_i \cdot pk_j) \\
&= e(R, g) \cdot e(H(\mathsf{TID})^{m}, pk_i \cdot pk_j) \\
&= \mathsf{Hash}(param, pk_i, pk_j, \mathsf{TID}, m, R).
\end{aligned}$$

So, the above scheme satisfies correctness. We now state the security theorems of the scheme. The proofs will be given in the full version of the paper due to the space limitation.

Theorem 1. *In the random oracle model, the above construction of accountable CH is secure against (existential) collision forgery under active attacks, assuming that the CDH assumption holds in the bilinear map group system* $\langle p, \mathbb{G}, \mathbb{G}_T, e\rangle$.

Theorem 2. *The above construction of ACH is forgery indistinguishable.*

5 Generic Construction of ATSS from ACH

Based on IBCH, Canard et al. [8] proposed a generic construction of TSS. In their construction, to sign a message $m = m_1 \| \cdots \| m_L$, the signer first

sets $\tilde{m} = \tilde{m}_1 \| \cdots \| \tilde{m}_L$, where $\tilde{m}_i = m_i$ if $i \notin I$ and otherwise, $\tilde{m}_i = h_i = \mathsf{IBCH.Hash}(param, \mathsf{ID}, m_i, r_i)$. The set of indices $I \subseteq [1, L]$ that are sanitizable and the identity ID associated with the transaction are generated by the signer. Then, the signer signs the message $\tilde{m}$ using a conventional signature scheme. Obviously, an entity with the trapdoor associated with ID generated by the signer can modify m_i and generate a new signature on the sanitized message. Our construction of ATSS is similar to the construction proposed by Canard et al. [8], but in order to achieve accountability we use ACH in place of IBCH. In order to generate a new signature on a sanitized message, the sanitizer need to use his private key and a trapdoor information associated with the transaction identifier generated by the signer to find a collision of the ACH. The signer can then use the collision to convince a trusted third party that a message is sanitized, as nobody apart from the sanitizer has more than a negligible probability of successfully finding a second message that produces the same signing value.

Now, given a regular signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$, and an ACH scheme $\Pi = (\Pi.\mathsf{GlobalSetup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Hash}, \Pi.\mathsf{Trapdoor}, \Pi.\mathsf{Forge})$, we define the 8-tuple algorithms ($\mathsf{GlobalSetup}$, KeyGen, Sign, $\mathsf{Trapdoor}$, $\mathsf{Sanitize}$, Verify, Proof, Judge) of an ATSS scheme as follows:

$\mathsf{GlobalSetup}$ Given a security parameter λ, it first runs $param_\Pi \leftarrow \Pi.\mathsf{GlobalSetup}(\lambda)$, and chooses two cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^\lambda, H_2 : \{0,1\}^* \rightarrow \mathcal{R}$. Then, it publishes the common public parameter $param = (param_\Pi, H_1, H_2)$.

KeyGen Given a security parameter λ and the common public parameter $param$, it first runs $(pk_\Sigma, sk_\Sigma) \leftarrow \Sigma.\mathsf{KeyGen}(\lambda)$, $(pk_\Pi, sk_\Pi) \leftarrow \Pi.\mathsf{KeyGen}(\lambda, param_\Pi)$. Then, it picks a key $\kappa_{sig} \in \{0,1\}^\lambda$ for the hash function H_1, sets the public key $pk = (pk_\Sigma, pk_\Pi)$ and the private key $sk = (sk_\Sigma, sk_\Pi, \kappa_{sig})$. Finally, it publishes pk and keeps sk secret. We denote by $pk_{sig} = (pk_{sig,\Sigma}, pk_{sig,\Pi})$ and $sk_{sig} = (sk_{sig,\Sigma}, sk_{sig,\Pi}, \kappa_{sig})$ the public key and private key of the signer, and by $pk_{san} = (pk_{san,\Sigma}, pk_{san,\Pi})$ and $sk_{san} = (sk_{san,\Sigma}, sk_{san,\Pi}, \kappa_{san})$ the public key and private key of a sanitizer.

Sign Given a sanitizer's public key $pk_{san} = (pk_{san,\Sigma}, pk_{san,\Pi})$, a message $m = m_1 \| \cdots \| m_L$, a set of indices $I \subseteq [1, L]$ that are sanitizable, a transaction identifier TID and the signer's private key $sk_{sig} = (sk_{sig,\Sigma}, sk_{sig,\Pi}, \kappa_{sig})$, it proceeds as follows.

1. Compute $z = H_1(\kappa_{sig}, \mathsf{TID})$; for all $i \in [1, L] \backslash I$, set $\tilde{m}_i = m_i$.
2. For all $i \in I$, compute $r_i = H_2(z, i)$ and $h_i = \Pi.\mathsf{Hash}(param, pk_{sig,\Pi}, pk_{san,\Pi}, \mathsf{TID}, m_i, r_i)$ and set $\tilde{m}_i = h_i$. Let r be the concatenation of all r_i, $i \in I$.
3. Set $\tilde{m} = \tilde{m}_1 \| \cdots \| \tilde{m}_L$ and run $\tilde{\sigma} \leftarrow \Sigma.\mathsf{Sign}(\tilde{m}, sk_{sig,\Sigma})$.
4. Finally, set $\sigma = \tilde{\sigma} \| r$ and output the signature σ on m.

$\mathsf{Trapdoor}$ Given a message m, a set of the indices I that are sanitizable, a transaction identifier TID, a valid signature σ on $(pk_{sig}, pk_{san}, m, I, \mathsf{TID})$ and the signer's private key $sk_{sig} = (sk_{sig,\Sigma}, sk_{sig,\Pi}, \kappa_{sig})$, it runs $td_{\mathsf{TID}} \leftarrow \Pi.\mathsf{Trapdoor}(sk_{sig,\Pi}, \mathsf{TID})$, and outputs the trapdoor td_{TID} associated with TID.

Sanitize Given the signer's public key $pk_{sig} = (pk_{sig,\Sigma}, pk_{sig,\Pi})$, a message $m = m_1\|\cdots\|m_L$, a set of indices $I \subseteq [1, L]$ that are sanitizable, the transaction identifier TID, a valid signature $\sigma = \tilde{\sigma}\|r$ on $(pk_{sig}, pk_{san} = (pk_{san,\Sigma}, pk_{san,\Pi}), m, I, \mathsf{TID})$, a trapdoor td_{TID} associated with TID, the sanitizer's private key $sk_{san} = (sk_{san,\Sigma}, sk_{san,\Pi}, \kappa_{san})$ and a new message $m' = m'_1\|\cdots\|m'_L$, it proceeds as follows.

1. Let $I' = \{i \in [1, L] | m_i \neq m'_i\}$. Check whether $I' \subseteq I$. If not, output $\perp$, denoted an error.
2. Retrieve $\{r_i, i \in I\}$ from the signature $\sigma = \tilde{\sigma}\|r$.
3. For all $i \in I'$, compute $h_i \leftarrow \Pi.\mathsf{Hash}(param, pk_{sig,\Pi}, pk_{san,\Pi}, \mathsf{TID}, m_i, r_i)$ and $r'_i \leftarrow \Pi.\mathsf{Forge}(sk_{san,\Pi}, td_{\mathsf{TID}}, pk_{sig,\Pi}, pk_{san,\Pi}, \mathsf{TID}, m_i, r_i, h_i, m'_i)$.
4. For all $i \in I\backslash I'$, set $r'_i = r_i$. Let r' be the concatenation of all r'_i, $i \in I$.
5. Set $\sigma' = \tilde{\sigma}\|r'$ and output the new signature σ' on $(pk_{sig}, pk_{san}, m', I, \mathsf{TID})$.

Verify Given $param$, the signer's public key $pk_{sig} = (pk_{sig,\Sigma}, pk_{sig,\Pi})$, a sanitizer's public key $pk_{san} = (pk_{san,\Sigma}, pk_{san,\Pi})$, a message $m = m_1\|\cdots\|m_L$, a set of indices $I \subseteq [1, L]$ that are sanitizable, a transaction identifier TID and a putative signature $\sigma = \tilde{\sigma}\|r$, it proceeds as follows.

1. Retrieve $\{r_i, i \in I\}$ from the signature $\sigma = \tilde{\sigma}\|r$.
2. For all $i \in [1, L]\backslash I$, set $\tilde{m}_i = m_i$.
3. For all $i \in I$, compute $h_i = \Pi.\mathsf{Hash}(param, pk_{sig,\Pi}, pk_{san,\Pi}, \mathsf{TID}, m_i, r_i)$ and set $\tilde{m}_i = h_i$.
4. Set $\tilde{m} = \tilde{m}_1\|\cdots\|\tilde{m}_L$ and output $\Sigma.\mathsf{Verify}(pk_{sig,\Sigma}, \tilde{m}, \tilde{\sigma})$.

Proof Given $param$, the signer's private key $sk_{sig} = (sk_{sig,\Sigma}, sk_{sig,\Pi}, \kappa_{sig})$, a valid message-signature pair $((pk_{sig} = (pk_{sig,\Sigma}, pk_{sig,\Pi}), pk_{san} = (pk_{san,\Sigma}, pk_{san,\Pi}), m, I, \mathsf{TID}), \sigma)$, and a set of (polynomially many) additional message-signature pairs $MesSigS = ((pk_{sig}, pk^{(i)}_{san}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}), \sigma^{(i)})_{i=1,2,\ldots,q}$ generated originally by the signer, it first searches the set $MesSigS$ to find a tuple $((pk_{sig}, pk^{(i)}_{san}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}), \sigma^{(i)})$ such that

1. $pk_{san} = pk^{(i)}_{san}$, $I = I^{(i)}$ and $\mathsf{TID} = \mathsf{TID}^{(i)}$.
2. $I' \subseteq I$, where $I' = \{j \in [1, L] | m_j \neq m^{(i)}_j\}$. Note that, $m = m_1\|\cdots\|m_L$ and $m^{(i)} = m^{(i)}_1\|\cdots\|m^{(i)}_L$.
3. $\Pi.\mathsf{Hash}(param, pk_{sig,\Pi}, pk_{san,\Pi}, \mathsf{TID}, m_j, r_j) = \Pi.\mathsf{Hash}(param, pk_{sig,\Pi}, pk^{(i)}_{san,\Pi}, \mathsf{TID}, m^{(i)}_j, r^{(i)}_j)$ for all $j \in I'$, where $\sigma = \tilde{\sigma}\|r$, $r = \{r_j, j \in I\}$, $\sigma^{(i)} = \tilde{\sigma}\|r^{(i)}$ and $r^{(i)} = \{r^{(i)}_j, j \in I\}$.

Then, it computes $z_i = H_1(\mathsf{TID}^{(i)}, \kappa_{sig})$. Finally, it outputs the proof $\pi = (pk_{sig}, pk^{(i)}_{san}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}, \sigma^{(i)}, z_i)$.

Judge Given $param$, the signer's public key $pk_{sig} = (pk_{sig,\Sigma}, pk_{sig,\Pi})$, the sanitizer's public key $pk_{san} = (pk_{san,\Sigma}, pk_{san,\Pi})$, a valid message-signature pair $((pk_{sig}, pk_{san}, m, I, \mathsf{TID}), \sigma)$ and a proof $\pi = (pk_{sig}, pk^{(i)}_{san}, m^{(i)}, I^{(i)}, \mathsf{TID}^{(i)}, \sigma^{(i)}, z_i)$. Let $\sigma = \tilde{\sigma}\|r$ where $r = \{r_j, j \in I\}$, and $\sigma^{(i)} = \tilde{\sigma}\|r^{(i)}$ where $r^{(i)} = \{r^{(i)}_j, j \in I\}$, it first checks whether the following conditions hold:

1. $pk_{san} = pk^{(i)}_{san}$, $I = I^{(i)}$ and $\mathsf{TID} = \mathsf{TID}^{(i)}$;
2. $(m, \sigma) \neq (m^{(i)}, \sigma^{(i)})$;

3. $I' \subseteq I$, where $I' = \{j \in [1, L] | m_j \neq m_j^{(i)}\}$. Note that, $m = m_1 \| \cdots \| m_L$ and $m^{(i)} = m_1^{(i)} \| \cdots \| m_L^{(i)}$;
4. $\Pi.\mathsf{Hash}(param, pk_{sig,\Pi}, pk_{san,\Pi}, \mathsf{TID}, m_j, r_j) = \Pi.\mathsf{Hash}(param, pk_{sig,\Pi}, pk_{san,\Pi}, \mathsf{TID}, m_j^{(i)}, r_j^{(i)})$ for all $j \in I'$;
5. $r_j^{(i)} = H_2(z_i, j)$ for all $j \in I$.

If so, it outputs `San` indicating the message-signature pair $((pk_{sig}, pk_{san}, m, I, \mathsf{TID}), \sigma)$ was created by the sanitizer; else it outputs `Sig` indicating the message-signature pair $((pk_{sig}, pk_{san}, m, I, \mathsf{TID}), \sigma)$ was created by the signer.

It is obvious that the above ATSS scheme satisfies correctness. We now state the security theorems of the scheme, including *unforgeability*, *indistinguishability* and *accountability*. The proofs will be given in the full version of the paper due to the space limitation.

Theorem 3 (Unforgeability). *If the signature scheme Σ is existential unforgeable under adaptive chosen message attacks and the ACH scheme Π is resistant to collision forgery under active attacks, the above construction of ATSS is existential unforgeable under adaptive chosen message attacks.*

Theorem 4 (Indistinguishability). *If the ACH scheme Π is forgery indistinguishable, in the random oracle model, the following distributions $\mathcal{D}_{\mathsf{Sanitize}}$ and $\mathcal{D}_{\mathsf{Sign}}$ are computational indistinguishable for all sufficiently large λ, any $param \leftarrow \mathsf{GlobalSetup}$, $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, $(pk_{san}, sk_{san}) \leftarrow \mathsf{KeyGen}(\lambda, param)$, any set of indices $I \subseteq [1, L]$, any message pairs m, m' such that $m_i = m'_i$ for all $i \notin I$, and any transaction identifier TID:*

$$\begin{aligned}
\mathcal{D}_{\mathsf{Sanitize}} &= \{(m', \hat{\sigma}) | \sigma \leftarrow \mathsf{Sign}(pk_{san}, m, I, \mathsf{TID}, sk_{sig}), \\
&\quad td_{\mathsf{TID}} \leftarrow \mathsf{Trapdoor}(m, I, \mathsf{TID}, \sigma, sk_{sig}), \\
&\quad \hat{\sigma} \leftarrow \mathsf{Sanitize}(pk_{sig}, m, I, \mathsf{TID}, \sigma, m', td_{\mathsf{TID}}, sk_{san})\}_{\lambda, param, pk_{sig}, pk_{san}, I, \mathsf{TID}}, \\
\mathcal{D}_{\mathsf{Sign}} &= \{(m', \sigma') | \sigma' \leftarrow \mathsf{Sign}(pk_{san}, m', I, \mathsf{TID}, sk_{sig})\}_{\lambda, param, pk_{sig}, pk_{san}, I, \mathsf{TID}}.
\end{aligned}$$

Theorem 5 (Sanitizer-accountability). *If the signature scheme Σ is existential unforgeable under adaptive chosen message attacks, the above construction of ATSS is sanitizer-accountable.*

Theorem 6 (Signer-accountability). *If the ACH scheme Π is resistant to collision forgery under active attacks, in the random oracle model, the above construction of ATSS is signer-accountable.*

6 Conclusion and Future Work

In this paper, we motivated and introduced the notion of accountable trapdoor sanitizable signature (ATSS). As a building block of ATSS and that might be of independent interest, we also introduced the notion of accountable chameleon hash (ACH), which is an extension of chameleon hash. We defined the security requirements for ACH, and proposed a concrete construction that satisfies

the requirements based on the CDH assumption in the random oracle model. Finally, Based on ACH, we proposed a generic construction of ATSS. Instantiating the generic construction with our ACH scheme, we constructed the first ATSS scheme. An important future research problem is to construct ACH schemes (and thus accordingly, ATSS schemes) in the standard model.

Acknowledgement. The authors thank the anonymous reviewers for their helpful comments. This work is in part supported by the Office of Research, Singapore Management University. The first author is partially supported by Natural Science Foundation of China (No. 61272453).

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
3. Ateniese, G., de Medeiros, B.: On the key exposure problem in chameleon hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005)
4. Bao, F., Deng, R.H., Ding, X., Lai, J., Zhao, Y.: Hierarchical identity-based chameleon hash and its applications. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 201–219. Springer, Heidelberg (2011)
5. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The case of hashing and signing. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg (1994)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
7. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
8. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor sanitizable signatures and their application to content protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008)
9. Chen, X., Zhang, F., Kim, K.: Chameleon hashing without key exposure. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 87–98. Springer, Heidelberg (2004)
10. Chen, X., Zhang, F., Susilo, W., Tian, H., Li, J., Kim, K.: Identity-based chameleon hash scheme without key exposure. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 200–215. Springer, Heidelberg (2010)
11. Chen, X., Zhang, F., Tian, H., Wei, B., Kim, K.: Key-exposure free chameleon hashing and signatures based on discrete logarithm systems. Cryptology ePrint Archive, Report 2009/035 (2009), `http://eprint.iacr.org/`

12. Deng, R.H., Yang, Y.: A study of data authentication in proxy-enabled multimedia delivery systems: Model, schemes and application. ACM T. on Multimedia Computing, Communications and Applications 5(4), 28.1–28.20 (2009)
13. Gao, W., Li, F., Wang, X.: Chameleon hash without key exposure based on schnorr signature. Computer Standards & Interfaces 31(2), 282–285 (2009)
14. Gao, W., Wang, X., Xie, D.: Chameleon hashes without key exposure based on factoring. J. Comput. Sci. Technol. 22(1), 109–113 (2007)
15. 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)
16. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
17. 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)
18. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS (2000)
19. Micali, S., Rivest, R.L.: Transitive signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 236–243. Springer, Heidelberg (2002)
20. Miyazaki, K., Hanaoka, G., Imai, H.: Invisibly sanitizable digital signature scheme. IEICE Transactions 91-A(1), 392–402 (2008)
21. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions 88-A(1), 239–246 (2005)
22. Pöhls, H.C., Samelin, K., Posegga, J.: Sanitizable signatures in xml signature - performance, mixing properties, and revisiting the property of transparency. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 166–182. Springer, Heidelberg (2011)
23. 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)
24. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
25. Yum, D.H., Seo, J.W., Lee, P.J.: Trapdoor sanitizable signatures made easy. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 53–68. Springer, Heidelberg (2010)
26. Zhang, F., Safavi-Naini, R., Susilo, W.: Id-based chameleon hashes from bilinear pairings. Cryptology ePrint Archive, Report 2003/208 (2003), `http://eprint.iacr.org/`

A Conditional Proxy Broadcast Re-Encryption Scheme Supporting Timed-Release

Kaitai Liang[1], Qiong Huang[2,⋆], Roman Schlegel[1], Duncan S. Wong[1], and Chunming Tang[3]

[1] Department of Computer Science, City University of Hong Kong, China
kliang4@student.cityu.edu.hk, rschlegel@gmx.ch, duncan@cityu.edu.hk
[2] College of Informatics, South China Agricultural University, Guangzhou, China
csqhuang@alumni.cityu.edu.hk
[3] School of Mathematics and Information Science, Guangzhou University, China
ctang@gzhu.edu.cn

Abstract. To allow a delegator not only to delegate the keyword-controlled decryption rights of a broadcast encryption to a set of specified recipients, but also to control when the decryption rights will be delegated, in this paper, for the first time, we introduce a new notion called Timed-Release Conditional Proxy Broadcast Re-Encryption (TR-CPBRE). We also propose a concrete construction for TR-CPBRE which can be proven selective identity adaptive CCA secure under the (P, Q, f)-general decisional Diffie-Hellman exponent assumption, and chosen-time period chosen-ciphertext secure under the bilinear Diffie-Hellman assumption. When compared with the existing CPBRE and Timed-Release Proxy Re-Encryption (TR-PRE) schemes, our scheme achieves better efficiency, and enables the delegator to make a fine-grained delegation of decryption rights to multiple delegatees.

Keywords: timed-release encryption, unidirectional conditional proxy broadcast re-encryption, bilinear map.

1 Introduction

Introduced by May [24] and further elaborated by Rivest et al. [26], Timed-Release Encryption (TRE) is a kind of time-dependent encryption where even a designated recipient cannot decrypt a ciphertext before a semi-trusted time server releases a trapdoor associated with the release time of the encryptor's choice. It has been found to have many real-world applications, such as sealed-bid auctions [11] and electronic-voting. To date, there have been some papers [12,13,20,23,25] that have proposed different variants of TRE.

⋆ Q. Huang is supported by the National Natural Science Foundation of China (No. 61103232), the Research Fund for the Doctoral Program of Higher Education of China (No. 20114404120027), and the Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (No. LYM11033).

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 132–146, 2013.

The traditional TRE only supports single recipient that seems undesirable in practice as a message might be intended for several recipients simultaneously. In 2005, Cathalo et al. [10] proposed an efficient TRE scheme, in which an encryptor is allowed to encrypt a message to multi-recipient with the same release time. The scheme is applicable to many network applications. Suppose there is an international programming contest, such as ACM-ICPC[1] and Google Code Jam[2]. The participating teams that are located all over the world are managed by different universities. All teams will be granted access to the problem set in a specified time. To prevent some unfair issues incurred by the network congestion or delivery delay, the contest organizer might prefer to allow all universities and teams to receive the problem set before the beginning of the contest, but not to open the set prior to the pre-specified time.

The above problem can be solved by using [10] as follows. The organizer first specifies a release time RT for a semi-trusted time server. With knowledge of the public keys of all registered universities and teams, the organizer encrypts the problem set m (as well as RT) (e.g., $Enc(PK_A, m, RT)$, $Enc(PK_{G1}, m, RT)$), and further sends the resulting ciphertexts to each university. Upon receiving the ciphertexts from the organizer, the university keeps its own ciphertext locally, and then forwards the rest of ciphertexts to the corresponding teams (whose identities are recorded in the register list). When the release time has arrived, the time server will release a trapdoor τ (corresponding to RT) such that the universities and teams can access the problem set simultaneously (See Fig. 1).

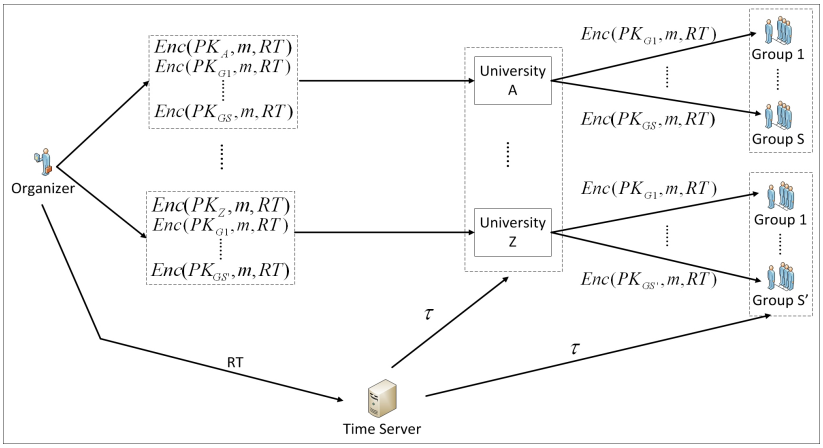


Fig. 1. Timed-Release Encryption for International Programming Contest

The above solution, however, comes at a price that the organizer has to perform $n_1 \times (n_2 + 1)$ encryptions; meanwhile, $(n_2 + 1)$ ciphertexts are needed to be sent from the organizer to each university, where n_1 is the total number of university, and n_2 is the maximum number of team supervised by each university.

[1] http://icpc.baylor.edu/
[2] http://code.google.com/codejam/

This might be undesirable in practice due to the incurred linear communication complexity and computation cost.

To reduce the complexity, we might employ some existing cryptographic primitives in the above scenario. Intuitively, Broadcast Encryption (BE) that addresses the problem of confidentially broadcasting a message to a group of recipients might be one of candidates. Despite there exist some BE schemes (e.g., [6,7,18]) in the literature, it is unknown that whether these schemes can be extended to support timed-release property or not.

Proxy Re-Encryption (PRE) proposed by Blaze, Bleumer and Stauss [3], which increases the flexibility of data sharing, allows a semi-trusted proxy to transform a ciphertext intended for Alice into another ciphertext intended for Bob. The proxy, however, can learn nothing of the plaintext. PRE is applicable to many network applications, such as secure distributed files systems [1] and email forwarding encryption [3]. Since its introduction, many classic PRE schemes (e.g., [8,22,19]) have been proposed.

To employ PRE in the context of TRE, Emura et al. [17] proposed the first Timed-Release Proxy Re-Encryption (TR-PRE) that might be another candidate to solve the linear complexity problem. In TR-PRE, the proxy is allowed to re-encrypt a ciphertext with a release time under a public key to the one with the same release time under another public key by using a re-encryption key given by the delegator. Here we use *University A* as an example. By uploading n_2 re-encryption keys (e.g., $rk_{A \to Group\ 1}, ..., rk_{A \to Group\ S}$) and its ciphertext $Enc(PK_A, m, RT)$ to the cloud (i.e. the proxy), *University A* (i.e. the delegator) can request the proxy to re-encrypt the ciphertext to the ones intended for the teams under A's control (denoted the team set as $I_A = \{Group\ 1, ..., Group\ S\}$).

Despite TR-PRE allows the proxy to fulfill the re-encryption so as to relieve the workload of the organizer (who does not need to generate $(n_1 \times n_2)$ ciphertexts), the organizer still needs to generate n_1 encryptions for universities, and each university has to construct n_2 re-encryption keys. Moreover, without supporting any keyword (conditional) control on re-encryption, once given a re-encryption key (e.g., $rk_{A \to Group\ 1}$), the proxy can re-encrypt all ciphertexts of *University A* to *Group* 1. This will incur the potential risk for access control.

Conditional Proxy Broadcast Re-Encryption (CPBRE), which was proposed by Chu et al. [14], can further reduce the cost incurred by TR-PRE. Specifically, CPBRE allows a delegator to delegate the decryption rights of a broadcast encryption to a set of delegatees, and to specify a condition to control the re-encryption power of the proxy. In CPBRE, only one (instead of n_2) re-encryption key is required to be generated by each university. Besides, the organizer only needs to generate one (instead of n_1) ciphertext for the university set (denoted as $I_U = \{University\ A, ..., University\ Z\}$). Thus CPBRE is an appropriate primitive for solving the linearly complexity problem. Nevertheless, the existing CPBRE[3] cannot be trivially extended to support timed-release property due to the limitation of its proof technique. In the security proof given in [14], the challenger

[3] The only and available CPBRE due to Chu et al. is secure against Replayable Chosen Ciphertext Attacks (RCCA) [9].

outputs a valid challenge ciphertext with the help of the challenger of an Hierarchical Identity-Coupling Broadcast Encryption (HICBE) [2]. To support timed-release, the challenge ciphertext has to be modified accordingly. This is out of the capability of the HICBE's challenger, that is, the challenger cannot output the corresponding challenge ciphertext. Therefore, a new CPBRE supporting timed-release property (i.e. Timed-Release Conditional Proxy Broadcast Re-Encryption (TR-CPBRE)) is desirable.

1.1 Our Contributions

In this paper, we formalize the definition and security models for TR-CPBRE. Specifically, a release time is required as an auxiliary input to the encryption and re-encryption key algorithms; meanwhile, this release time and its corresponding timed-release key are required in the input to the decryption algorithms. Note that a timed-release key is generated by a timed-release key generation algorithm that takes in the secret key of a semi-trusted time server and a given release time.

For security models, we consider two different aspects: one is to allow the adversary to get the timed-release key but not the secret key, that is, even if given the timed-release key, the adversary cannot decrypt a ciphertext without the appropriate secret key; the other is the inverse case, that is, even if given the secret key, the adversary cannot decrypt a corresponding ciphertext without the appropriate timed-release key. As of [14,17], we refer to the security of the former and the latter as *chosen ciphertext security* and *chosen-time-period and chosen-ciphertext security*, respectively.

Besides, we propose the first TR-CPBRE that is selective ID CCA (IND-sID-CCA) secure (under the (P, Q, f)-general decisional Diffie-Hellman exponent assumption), and is secure against chosen-time period chosen-ciphertext attacks (CTCA) (under the bilinear Diffie-Hellman assumption) in the random oracle model. In our scheme, the organizer is only needed to generate one ciphertext (with a release time RT and some condition c) for the university set I_U (i.e. $Enc(I_U, m, RT, c)$). To delegate the decryption rights of the broadcast encryption to its team set I_A, $University\ A$ first specifies to which group's broadcast encryption it would like to delegate (I_U), and next generates a re-encryption key from itself to I_A under c and RT (e.g., $rk_{A \to I_A|c,RT}$) for the proxy. The proxy then re-encrypts the ciphertext to the one ($Enc(I_A, m, RT, c)$) that can be only decrypted by the members of I_A (See Fig. 2). Note that the release time is still effective after the re-encryption. To the best of our knowledge, no TRE and PRE scheme (in general) capture timed-release and broadcast encryption properties simultaneously. TR-CPBRE is the first of its type.

Our scheme solves the limitations incurred by [17] in the sense that it only requires one re-encryption key and one re-encryption ciphertext rather than n_2 copies of them for each set of teams; meanwhile, it supports conditional control on re-encryption (i.e. conditional delegation). Thus our scheme enjoys improvement in communication compared to [17].

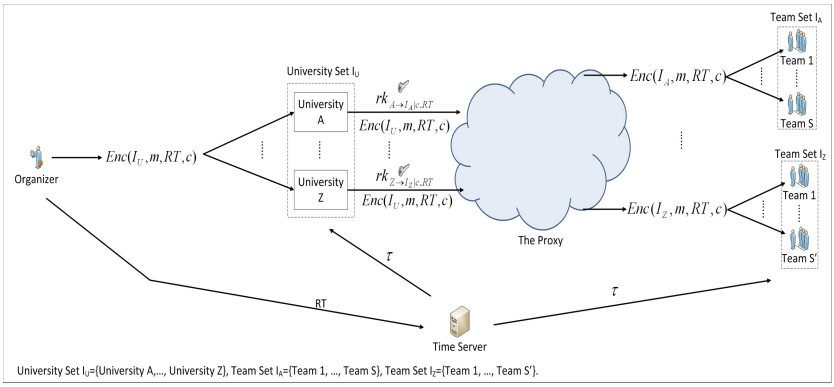


Fig. 2. Timed-Release Conditional Proxy Broadcast Re-Encryption

We argue that TR-CPBRE has many other real world applications, such as on-line learning systems (IXL[4]). For example, in an on-line learning system, the service provider can broadcast the learning materials in terms of different semesters with different release times to the universities which support on-line teaching, such that each university can accordingly open the classes in different semesters for its on-line learners.

We summarize the comparison of properties between our scheme, [14] and [17] in Table 1. While conditional delegation, broadcast re-encryption and timed-release property have been partially achieved by previous schemes, there is no CCA-secure proposal that achieves such properties simultaneously. However, this paper achieves the goal.

Table 1. Property Comparison

Schemes	Security	Selective Security	Conditional Delegation	Broadcast Re-Encryption	Timed-Release Property
CPBRE [14]	RCCA	✓	✓	✓	✗
TR-PRE [17]	RCCA	✗	✗	✗	✓
Our TR-CPBRE	**CCA**	✓	✓	✓	✓

2 Definition and Security Models

2.1 Definition of TR-CPBRE

Definition 1. *A (single-hop unidirectional) Timed-Release Conditional Proxy Broadcast Re-Encryption (TR-CPBRE) scheme consists of the following algorithms:*

[4] `http://www.ixl.com/`

1. $(param, msk, sk_{TS}) \leftarrow Setup(1^\lambda, n)$: *on input a security parameter* λ *and* n, *which indicates the maximum allowable number of receivers, output a public key* $param$, *a master secret key* msk, *a secret key* sk_{TS} *and a public key* TP *of a Time Server. Note that* TP *is regarded as one part of* $param$.
2. $sk_{ID} \leftarrow KeyGen(param, msk, ID)$: *on input* $param$, msk, *and an identity* $ID \in \{0,1\}^*$, *output a secret key* sk_{ID} *for identity* ID.
3. $\tau \leftarrow TS(sk_{TS}, RT)$: *on input* sk_{TS} *and a release time* $RT \in \{0,1\}^\lambda$, *output a timed-release key* τ.
4. $rk_{ID_i \to \overline{S}|RT,c} \leftarrow ReKeyGen(param, ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$: *on input* $param$, *an identity* ID_i *and the corresponding secret key* sk_{ID_i}, *two identity sets* S *and* $\overline{S}$, *a condition* $c \in \{0,1\}^*$ *and a release time* RT, *output a re-encryption key* $rk_{ID_i \to \overline{S}|RT,c}$, *where* $ID_i \in S$.
5. $C \leftarrow Enc(param, S, c, RT, m)$: *on input* $param$, *an identity set* S, c, RT *and a message* $m \in \{0,1\}^\lambda$, *output an original ciphertext* C.
6. $C_R \leftarrow ReEnc(param, rk_{ID_i \to \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$: *on input* $param$, *a re-encryption key* $rk_{ID_i \to \overline{S}|RT,c}$, *an identity* ID_i, *an identity set* S *such that* $ID_i \in S$, *an identity set* $\overline{S}$, c, RT *and* C, *output a re-encrypted ciphertext* C_R *or* $\perp$ *for failure.*
7. $m \leftarrow Dec(param, sk_{ID_i}, ID_i, S, c, RT, C, \tau)$: *on input* $param$, sk_{ID_i}, ID_i, S *such that* $ID_i \in S$, c, RT, *an original ciphertext* C *and* τ, *output a message* m *or* $\perp$ *for failure.*
8. $m \leftarrow Dec_R(param, sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau)$: *on input* $param$, *a delegatee's secret key* $sk_{\overline{ID}_{i'}}$, *a delegator's identity* ID_i, *a delegatee's identity* $\overline{ID}_{i'}$, *an identity set* S *such that* $ID_i \in S$, *an identity set* $\overline{S}$ *such that* $\overline{ID}_{i'} \in \overline{S}$, c, RT, *a re-encrypted ciphertext* C_R *and* τ, *output a message* m *or* $\perp$ *for failure.*

For simplicity, hereafter we omit *param* in the expression of the algorithms input.

Correctness: For any $\lambda, n \in \mathbb{N}$, any identity sets $S, \overline{S}$, any identities $ID_i, \overline{ID}_{i'} \in \{0,1\}^*$ such that $ID_i \in S$, $\overline{ID}_{i'} \in \overline{S}$, any condition $c \in \{0,1\}^*$, any release time $RT \in \{0,1\}^\lambda$ and any message $m \in \{0,1\}^\lambda$, if $(param, msk, sk_{TS}) \leftarrow Setup(1^\lambda, n)$, $\tau \leftarrow TS(sk_{TS}, RT)$, $sk_{ID} \leftarrow KeyGen(msk, ID)$, for all ID used in the system, $rk_{ID_i \to \overline{S}|RT,c} \leftarrow ReKeyGen(ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$, $C \leftarrow Enc(S, c, RT, m)$, and $C_R \leftarrow ReEnc(rk_{ID_i \to \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$, we have

$$Dec(sk_{ID_i}, ID_i, S, c, RT, C, \tau) = m;$$
$$Dec_R(sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau) = m.$$

2.2 Security Models

There are two main security requirements for TR-CPBRE: IND-sID-CCA security and CTCA security. Here we only give the security notions of original ciphertext. Note that the security notions of re-encrypted ciphertext can be defined in the same manner, we hence omit the details. We start with the formalization of IND-sID-CCA security. For capturing timed-release feature, in the

model, we require that an adversary $\mathcal{A}$ is not able to win the game even if the time server's secret key sk_{TS} is known.

Definition 2. *A (single-hop unidirectional) TR-CPBRE scheme is IND-sID-CCA-secure at original ciphertext if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can win the game below with non-negligible advantage. In the game, $\mathcal{C}$ is the game challenger, λ and n are the security parameter and the maximum allowable number of receivers, respectively.*

1. ***Initialization.*** *$\mathcal{A}$ outputs a challenge identity set $S^* = \{ID_1^*, ..., ID_s^*\}$, where $s \leq n$.*
2. ***Setup.*** *$\mathcal{C}$ runs $Setup(1^\lambda, n)$ and sends $param, sk_{TS}$ to $\mathcal{A}$.*
3. ***Phase 1.*** *$\mathcal{A}$ is given access to the following oracles.*
   (a) *$\mathcal{O}_{sk}(ID)$: on input an identity ID, output $sk_{ID} \leftarrow KeyGen(msk, ID)$. If $Extract(ID')$ is queried, we say that ID' is corrupted. One restriction is that $\mathcal{A}$ cannot query $Extract(ID)$ for any $ID \in S^*$.*
   (b) *$\mathcal{O}_{rk}(ID_i, S, \overline{S}, c, RT)$: on input an identity ID_i, two sets S and $\overline{S}$, a condition c and a release time RT, output $rk_{ID_i \rightarrow \overline{S}|RT,c} \leftarrow ReKeyGen(ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$, where $sk_{ID_i} \leftarrow KeyGen(msk, ID_i)$, $ID_i \in S$.*
   (c) *$\mathcal{O}_{re}(ID_i, S, \overline{S}, c, RT, C)$: on input an identity ID_i, two sets S and $\overline{S}$, a condition c, a release time RT, and an original ciphertext C, output $C_R \leftarrow ReEnc(rk_{ID_i \rightarrow \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$, where $rk_{ID_i \rightarrow \overline{S}|RT,c} \leftarrow ReKeyGen\ (ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$, $sk_{ID_i} \leftarrow KeyGen(msk, ID_i)$, $ID_i \in S$.*
   (d) *$\mathcal{O}_{dec}(ID_i, S, c, RT, C)$: on input an identity ID_i, an identity set S, a condition c, a release time RT, and an original ciphertext C, output $m \leftarrow Dec(sk_{ID_i}, ID_i, S, c, RT, C, \tau)$, where $sk_{ID_i} \leftarrow KeyGen(msk, ID_i)$, $\tau \leftarrow TS(sk_{TS}, RT)$, $ID_i \in S$.*
   (e) *$\mathcal{O}_{decr}(ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R)$: on input two identities ID_i and $\overline{ID}_{i'}$, two identity sets S and $\overline{S}$, a condition c, a release time RT, and a re-encrypted ciphertext C_R, output $m \leftarrow Dec_R(sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau)$, where $sk_{\overline{ID}_{i'}} \leftarrow KeyGen(msk, \overline{ID}_{i'})$, $\tau \leftarrow TS(sk_{TS}, RT)$, $\overline{ID}_{i'} \in \overline{S}$, $ID_i \in S$.*
4. ***Challenge.*** *$\mathcal{A}$ outputs two equal length messages m_0, m_1, a challenge condition c^* and a challenge release time RT^* to $\mathcal{C}$. If the queries $\mathcal{O}_{rk}(ID_i, S^*, \overline{S}, c^*, RT^*)$ and $\mathcal{O}_{sk}(\overline{ID}_{i'})$ are never made, $\mathcal{C}$ returns $C^* = Enc(S^*, c^*, RT^*, m_b)$ to $\mathcal{A}$, where $b \in_R \{0, 1\}$, $ID_i \in S^*$ and $\overline{ID}_{i'} \in \overline{S}$.*
5. ***Phase 2.*** *$\mathcal{A}$ continues making queries as in Phase 1 except the following:*
   (a) *$\mathcal{O}_{sk}(ID)$ for any $ID \in S^*$;*
   (b) *$\mathcal{O}_{rk}(ID_i, S^*, \overline{S}, c^*, RT^*)$ and $\mathcal{O}_{sk}(\overline{ID}_{i'})$ for any $ID_i \in S^*$ and $\overline{ID}_{i'} \in \overline{S}$;*
   (c) *$\mathcal{O}_{dec}(ID_i, S^*, c^*, RT^*, C^*)$ for any $ID_i \in S^*$;*
   (d) *$\mathcal{O}_{re}(ID_i, S^*, \overline{S}, c^*, RT^*, C^*)$ and $\mathcal{O}_{sk}(\overline{ID}_{i'})$ for any $ID_i \in S^*$, $\overline{ID}_{i'} \in \overline{S}$;*
   (e) *$\mathcal{O}_{decr}(ID_i, \overline{ID}_{i'}, S^*, \overline{S}, c^*, RT^*, C_R)$ for any C_R, $ID_i \in S^*$, $\overline{ID}_{i'} \in \overline{S}$, where $(\overline{S}, c^*, RT^*, C_R)$ is a derivative of (S^*, c^*, RT^*, C^*). As of [8], the derivative of (S^*, c^*, RT^*, C^*) is defined as follows.*

i. (S^*, c^*, RT^*, C^*) *is a derivative of itself.*

ii. If $\mathcal{A}$ *has issued a re-encryption key query on* $(ID_i, S^*, \overline{S}, c^*, RT^*)$ *to obtain* $rk_{ID_i \to \overline{S}|RT^*,c^*}$, *and* $C_R \leftarrow ReEnc(rk_{ID_i \to \overline{S}|RT^*,c^*}, ID_i, S^*, \overline{S}, c^*, RT^*, C^*)$, *then* $(\overline{S}, c^*, RT^*, C_R)$ *is a derivative of* (S^*, c^*, RT^*, C^*), *where* $ID_i \in S^*$.

iii. If $\mathcal{A}$ *has issued a re-encryption query on* $(ID_i, S^*, \overline{S}, c^*, RT^*, C^*)$ *and obtained* C_R, *then* $(\overline{S}, c^*, RT^*, C_R)$ *is a derivative of* (S^*, c^*, RT^*, C^*).

6. ***Guess.*** $\mathcal{A}$ *outputs a guess bit* $b' \in \{0,1\}$. *If* $b' = b$, $\mathcal{A}$ *wins.*

The advantage of $\mathcal{A}$ *is* $\epsilon = Adv^{IND-sID-CCA-Or}_{TR-CPBRE,\mathcal{A}}(1^\lambda, n) = |Pr[b' = b] - \frac{1}{2}|$.

We now proceed to the IND-CTCA security.

Definition 3. *A (single-hop unidirectional) TR-CPBRE scheme is IND-CTCA-secure at original ciphertext if the advantage* $Adv^{IND-CTCA-Or}_{TR-CPBRE,\mathcal{A}}(1^\lambda, n)$ *is negligible for any PPT adversary* $\mathcal{A}$ *in the following experiment. Set* $O = \{\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{ts}, \mathcal{O}_{re}, \mathcal{O}_{dec}, \mathcal{O}_{decr}\}$.

$$Adv^{IND-CTCA-Or}_{TR-CPBRE,\mathcal{A}}(1^\lambda, n) = |Pr[b' = b : (param, msk, sk_{TS}) \leftarrow Setup(1^\lambda, n); (m_0, m_1, S^*, RT^*, c^*, State) \leftarrow A^O(param); b \in_R \{0,1\}; C^* \leftarrow Enc(S^*, c^*, RT^*, m_b); b' \leftarrow A^O(C^*, State)] - \frac{1}{2}|,$$

where State is the state information, $\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}_{dec}, \mathcal{O}_{decr}$ *are the oracles defined in Definition 2.* $\mathcal{O}_{ts}$ *is the timed-release key extraction oracle that takes as input a release time RT (except for the challenge release time* RT^**) and outputs a timed-release key* τ. *The constraints for* $\mathcal{O}_{dec}$ *and* $\mathcal{O}_{decr}$ *remain the same as those in Definition 2, while there is no restriction for* $\mathcal{O}_{sk}$ *and* $\mathcal{O}_{rk}$. *Besides,* $\mathcal{O}_{re}$ *outputs* $\perp$ *if it is queried on a re-encrypted ciphertext.*

3 Preliminaries

Bilinear Maps. Let $BSetup$ be an algorithm that on input the security parameter λ, outputs the parameters of a bilinear map as $(q, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbb{G}_{T'}, e, \bar{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and $\mathbb{G}_{T'}$ are multiplicative cyclic groups of prime order q, where $|q| = \lambda$, and g, h are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The mappings $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and $\bar{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_{T'}$ have three properties: (1) *Bilinearity*: for all $a, b \in_R \mathbb{Z}_q^*$, $e(g^a, h^b) = e(g, h)^{ab}$, $e(g^a, g^b) = e(g, g)^{ab}$; (2) *Non-degeneracy*: $e(g, h) \neq 1_{\mathbb{G}_T}$, $e(g, g) \neq 1_{\mathbb{G}_{T'}}$, where $1_{\mathbb{G}_T}$ and $1_{\mathbb{G}_{T'}}$ are the unit of $\mathbb{G}_T$ and $\mathbb{G}_{T'}$; (3) *Computability*: e and $\bar{e}$ can be efficiently computed.

The General Decisional Diffie-Hellman Exponent Assumption. We review the general decisional Diffie-Hellman exponent problem in the symmetric case so that $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ as in [16]. Let $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow BSetup(1^\lambda)$, and set $g_1 = e(g, g) \in \mathbb{G}_T$, where $\mathbb{G}, \mathbb{G}_T$ are two multiplicative cyclic groups

with prime order q and $g \in \mathbb{G}$ is a generator. Let s, n be positive integers and $P, Q \in \mathbb{F}_q[X_1, ..., X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_q$. We write $P = (p_1, ..., p_s)$, $Q = (q_1, ..., q_s)$ and set $p_1 = q_1 = 1$. For any function $h : \mathbb{F}_q \rightarrow \Omega$ and vector $(x_1, ..., x_n) \in \mathbb{F}_q^n$, $h(P(x_1, ..., x_n))$ denotes $(h(p_1(x_1, ..., x_n)), ..., h(p_s(x_1, ..., x_n))) \in \Omega^s$. Note that a similar notation can be used for Q. Let $f \in \mathbb{F}_q[X_1, ..., X_n]$. If there exists the following linear decomposition: $f = \sum_{1 \leq i,j \leq s} a_{i,j} p_i p_j + \sum_{1 \leq i \leq s} b_i q_i$, where $a_{i,j}, b_i \in \mathbb{Z}_q$. We say that f depends on (P, Q), i.e. $f \in \langle P, Q \rangle$. The (P, Q, f)-General Decisional Diffie-Hellman Exponent ((P, Q, f)-GDDHE) problem [16] is defined as follows. Note that we let P, Q be as above and $f \in \mathbb{F}_q[X_1, ..., X_n]$.

Definition 4. ***(P,Q,f)-GDDHE Assumption.*** *Given the tuple $H(x_1, ..., x_n) = (g^{P(x_1,...,x_n)}, g_1^{Q(x_1,...,x_n)}) \in \mathbb{G}^s \times \mathbb{G}_T^s$ and $T \in_R \mathbb{G}_T$, the (P,Q,f)-GDDHE problem is to decide whether $T = g_1^{f(x_1,...,x_n)}$. Define $Adv_{\mathcal{A}}^{(P,Q,f)-GDDHE} = |Pr[\mathcal{A}(H(x_1, ..., x_n), g_1^{f(x_1,...,x_n)}) = 0] - Pr[\mathcal{A}(H(x_1, ..., x_n), T) = 0]|$ as the advantage of $\mathcal{A}$ in wining the (P,Q,f)-GDDHE problem. We say that the (P, Q, f)-GDDHE assumption holds in $\mathbb{G}$ if no PPT algorithm has non-negligible advantage.*

Definition 5. ***Bilinear Diffie-Hellman (BDH) Assumption [5].*** *Given the tuple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, the BDH problem is to compute $e(g, g)^{abc}$, where $a, b, c \in \mathbb{Z}_q^*$. Define $Adv_{\mathcal{A}}^{BDH} = Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}]$ as the advantage of $\mathcal{A}$ in winning the BDH problem. We say that the BDH assumption holds in $\mathbb{G}$ if no PPT algorithm has non-negligible advantage.*

Target Collision Resistant Hash Function. Target Collision Resistant (TCR) hash function was introduced by Cramer and Shoup [15]. A TCR hash function H guarantees that given a random element x which is from the valid domain of H, a PPT adversary $\mathcal{A}$ cannot find $y \neq x$ such that $H(x) = H(y)$. We let $Adv_{H,\mathcal{A}}^{TCR} = Pr[(x, y) \leftarrow \mathcal{A}(1^\lambda) : H(x) = H(y), x \neq y, x, y \in DH]$ be the advantage of $\mathcal{A}$ in successfully finding collisions from a TCR hash function H, where DH is the valid input domain of H. If a hash function is chosen from a TCR hash function family, $Adv_{H,\mathcal{A}}^{TCR}$ is negligible.

4 A New TR-CPBRE Scheme

In this section, we start with a new CPBRE scheme which is considered as a basic scheme for constructing a TR-CPBRE. The new scheme has two building blocks: an IBBE [16] and a TCR hash function [15] where we employ the CHK technique [4] using a TCR hash function to make a "signature" on the ciphertext and including a "verifying key" in the ciphertext. To extend the new scheme to a TR-CPBRE, we employ Boneh and Franklin (BF) IBE scheme [5], and regard a release time as an identity and a timed-release key as the secret key corresponding to the identity. We then use IBE to re-encrypt the ciphertext output such that the decryption of the underlying plaintext requires two pieces

of secret information: one is the secret key of the delegatee generated in our CPBRE, the other is the secret key of an identity generated in the IBE scheme.

Our technique is different from that of [14]. We begin with an IBBE scheme which is used to construct a new CCA-secure CPBRE, then we propose a TR-CPBRE by combining the new CPBRE with an IBE. In [14], Chu et al. started with an HICBE [2] and extended the HICBE to a RCCA-secure CPBRE. In the following, we provide the details of our construction.

1. $Setup(1^\lambda, n)$. Let $c \in \{0,1\}^*$ be a condition and $RT \in \{0,1\}^\lambda$ be a release time. Choose $\gamma, \bar{r} \in_R \mathbb{Z}_q^*$, three generators $g, g' \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ and hash functions: $H_0 : \{0,1\}^{2\lambda} \to \mathbb{Z}_q^*$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_T \to \{0,1\}^{2\lambda}$, $H_3 : \{0,1\}^* \to \mathbb{G}_1$, $H_4 : \{0,1\}^* \to \mathbb{G}_1$, $H_5 : \{0,1\}^\lambda \to \mathbb{Z}_q^*$, $H_6 : \{0,1\}^* \to \mathbb{G}_1$, $H_7 : \{0,1\}^\lambda \to \mathbb{G}_1$, $H_8 : \mathbb{G}_{T'} \to \{0,1\}^{2\lambda}$, $H_9 : \{0,1\}^{4\lambda} \to \mathbb{Z}_q^*$, $H_{10} : \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$. The master secret key is $msk = (g', \gamma)$, the public key is $param = (g, h, w, v, h^\gamma, \ldots, h^{\gamma^n}, H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8, H_9, H_{10}, TP)$, and the secret key of time server is $sk_{TS} = \bar{r}$, where $w = g'^\gamma$, $v = e(g', h)$ and $TP = g^{\bar{r}}$. Hereafter let s and s' be two maximum numbers of receivers in two identity sets S and $\overline{S}$, respectively, where $s \le n$, $s' \le n$.
2. $KeyGen(msk, ID)$. Given $msk = (g', \gamma)$ and an identity ID, output the secret key $sk_{ID} = g'^{\frac{1}{\gamma + H_1(ID)}}$.
3. $TS(sk_{TS}, RT)$. Given sk_{TS} and a release time RT, output a timed-release key $\tau = H_7(RT)^{\bar{r}}$.
4. $Enc(S, c, RT, m)$. Choose $\alpha \in_R \{0,1\}^\lambda, \sigma \in_R \{0,1\}^{2\lambda}$, compute $k = H_0(m,\alpha)$, set $C_1 = w^{-k}$, $C_2 = h^{k \cdot \prod_{i=1}^{s} (\gamma + H_1(ID_i))}$[5], $C_3 = (m||\alpha) \oplus H_2(e(g', h)^k)$, $\bar{C}_3 = C_3 \oplus H_{10}(\sigma)$, $C_4 = H_3(c, S, RT)^k$, $C_5 = H_4(C_1, \bar{C}_3, C_4, C_6, C_7)^k$, $C_6 = g^{\bar{k}}$, $C_7 = \sigma \oplus H_8(\bar{e}(H_7(RT), TP)^{\bar{k}})$, and output $C = (RT, C_1, C_2, \bar{C}_3, C_4, C_5, C_6, C_7)$, where $\bar{k} = H_9(\sigma, C_3)$, $ID_i \in S$, $m \in \{0,1\}^\lambda$.
5. $ReKeyGen(ID_i, sk_{ID_i}, S, \overline{S}, c, RT)$. Choose $\rho \in_R \mathbb{Z}_q^*$, $\{\theta, \alpha'\} \in_R \{0,1\}^\lambda$, compute $k' = H_0(\theta, \alpha')$, $rk_0 = sk_{ID_i}^{H_5(\theta)} \cdot (H_3(c, S, RT)^\rho)$, $rk_1 = w^{-k'}$, $rk_2 = h^{k' \cdot \prod_{i'=1}^{s'} (\gamma + H_1(\overline{ID}_{i'}))}$, $rk_3 = (\theta||\alpha') \oplus H_2(e(g', h)^{k'})$, $rk_4 = H_6(RT, c, rk_1, rk_2, rk_3)^{k'}$, $rk_5 = h^{\rho \cdot \prod_{i=1}^{s} (\gamma + H_1(ID_i))}$, and output the re-encryption key $rk_{ID_i \to \overline{S}|RT,c} = (rk_0, rk_1, rk_2, rk_3, rk_4, rk_5)$, where $ID_i \in S$, $\overline{ID}_{i'} \in \overline{S}$.
6. $ReEnc(rk_{ID_i \to \overline{S}|RT,c}, ID_i, S, \overline{S}, c, RT, C)$.
   (1) Verify the validity of original ciphertext C

$$\begin{aligned} &e(w^{-1}, C_2) \stackrel{?}{=} e(C_1, h^{\prod_{i=1}^{s} (\gamma + H_1(ID_i))}), \\ &\bar{e}(w^{-1}, C_4) \stackrel{?}{=} \bar{e}(C_1, H_3(c, S, RT)), \\ &\bar{e}(w^{-1}, C_5) \stackrel{?}{=} \bar{e}(C_1, H_4(C_1, \bar{C}_3, C_4, C_6, C_7)), ID_i \stackrel{?}{\in} S. \end{aligned} \tag{1}$$

   If Eq. (1) does not hold, output $\perp$. Otherwise, proceed.

[5] The encryptor can compute C_2 with knowledge of $param$, k and S.

(2) Compute $C_2' = e(rk_0, C_2)/e(C_4, rk_5)$, output $C_R = (RT,\ C_1,\ C_2',\ \bar{C}_3,\ C_4,\ C_6,\ C_7,\ rk_1,\ rk_2,\ rk_3,\ rk_4)$.

7. $Dec(sk_{ID_i}, ID_i, S, c, RT, C, \tau)$.
   (1) Verify Eq. (1). If the equation does not hold, output $\perp$. Otherwise, proceed.
   (2) Compute $\sigma = C_7 \oplus H_8(\bar{e}(\tau, C_6))$, $C_3 = \bar{C}_3 \oplus H_{10}(\sigma)$, $e(g', h)^k = (e(C_1, h^{B_{i,s(\gamma)}})e(sk_{ID_i}, C_2))^{\beta}$, and $m \| \alpha = C_3 \oplus H_2(e(g', h)^k)$, where $\beta = \frac{1}{\prod_{j=1, j\neq i}^{s} H_1(ID_j)}$, and $B_{i,s(\gamma)} = \frac{1}{\gamma} \cdot (\prod_{j=1,j\neq i}^{s}(\gamma + H_1(ID_j)) - \prod_{j=1,j\neq i}^{s} H_1(ID_j))$[6]. If $C_6 = g^{H_9(\sigma, C_3)}$ and $C_1 = w^{-H_0(m,\alpha)}$, output m. Otherwise, output $\perp$.

8. $Dec_R(sk_{\overline{ID}_{i'}}, ID_i, \overline{ID}_{i'}, S, \overline{S}, c, RT, C_R, \tau)$.
   (1) Compute $e(g', h)^{k'} = (e(rk_1, h^{B_{i',s'(\gamma)}})e(sk_{\overline{ID}_{i'}}, rk_2))^{\beta'}$, and $\theta \| \alpha' = rk_3 \oplus H_2(e(g', h)^{k'})$, where $\beta' = \frac{1}{\prod_{j'=1, j'\neq i'}^{s'} H_1(\overline{ID}_{j'})}$, and $B_{i',s'(\gamma)} = \frac{1}{\gamma} \cdot (\prod_{j'=1,j'\neq i'}^{s'}(\gamma + H_1(\overline{ID}_{j'})) - \prod_{j'=1,j'\neq i'}^{s'} H_1(\overline{ID}_{j'}))$.
   (2) Verify

$$\begin{aligned} rk_1 &\stackrel{?}{=} w^{-H_0(\theta,\alpha')}, rk_2 \stackrel{?}{=} h^{H_0(\theta,\alpha') \cdot \prod_{i'=1}^{s'}(\gamma + H_1(\overline{ID}_{i'}))}, \\ rk_4 &\stackrel{?}{=} H_6(RT, c, rk_1, rk_2, rk_3)^{H_0(\theta,\alpha')}, \overline{ID}_{i'} \stackrel{?}{\in} \overline{S}. \end{aligned} \tag{2}$$

   If Eq. (2) does not hold, output $\perp$. Otherwise, proceed.
   (3) Compute $\sigma = C_7 \oplus H_8(\bar{e}(\tau, C_6))$ and $C_3 = \bar{C}_3 \oplus H_{10}(\sigma)$. If $ID_i \in S$, compute $M = (e(C_1, h^{B_{i,s(\gamma)}})(C_2')^{H_5(\theta)^{-1}})^{\beta}, m || \alpha = C_3 \oplus H_2(M)$, where $\beta = \frac{1}{\prod_{j=1, j\neq i}^{s} H_1(ID_j)}$, $B_{i,s(\gamma)} = \frac{1}{\gamma} \cdot (\prod_{j=1,j\neq i}^{s}(\gamma + H_1(ID_j)) - \prod_{j=1,j\neq i}^{s} H_1(ID_j))$. If $C_1 = w^{-H_0(m,\alpha)}, C_4 = H_3(c, S, RT)^{H_0(m,\alpha)}$, and $C_6 = g^{H_9(\sigma, C_3)}$, output m. Otherwise, output $\perp$.

Correctness: It is easy to verify that the underlying plaintexts of the original and re-encrypted ciphertexts can be recovered correctly if the ciphertexts are computed via the description above. We hence skip the details.

Theorem 1. *Suppose (P, Q, f)-GDDHE assumption holds, our TR-CPBRE scheme for n receivers is IND-sID-CCA-secure at original ciphertext in the random oracle model. If there is a PPT adversary $\mathcal{A}$, who issues at most q_{H_i} queries to H_i and breaks the $(\bar{t}, q_{sk}, q_{rk}, q_{re}, q_{d2}, q_{d1}, \epsilon)$-IND-sID-CCA-Or security of our TR-CPBRE scheme, then we can construct a PPT adversary $\mathcal{C}$ to solve the (t', ϵ')-(P, Q, f)-GDDHE problem with*

[6] With knowledge of *param*, ID_i and S, the decryptor is able to compute $h^{B_{i,s(\gamma)}}$. More details can be found in [16].

$$\epsilon' \geq \frac{1}{q_{H_2}}(\frac{\epsilon}{2\dot{e}(1+q_{rk})} - \frac{q_{H_0}+(q_{H_0}+q_{H_2})(q_{d_1}+q_{d_2})}{2^{2\lambda}} - \frac{2(q_{d_1}+q_{d_2})+q_{re}}{q}),$$

$$\begin{aligned} t' \leq\ & \bar{t} + O(1)(q_{H_i} + q_{sk} + q_{rk} + q_{re} + q_{d2} + q_{d1}) + t_e(q_{sk} + q_{H_0}(2q_{re} + 2q_{d2} \\ & + 8q_{d1}) + (2n+9)q_{rk} + (3n+8)q_{re} + (n+2)q_{d2} + (2n+4)q_{d1}) \\ & + t_p(7q_{re} + 7q_{d2} + 2q_{d1}), \end{aligned}$$

where $\dot{e}$ denotes the base of the natural logarithm, t_e denotes the running time of an exponentiation, t_p denotes the running time of a pairing, q_{sk}, q_{rk}, q_{re}, q_{d2}, q_{d1} denote the total number of secret key extraction queries, re-encryption key extraction queries, re-encryption queries, the original and re-encrypted ciphertexts decryption queries, respectively, $i \in \{0,1,2,3,4,5,6,7,8,9,10\}$.

Due to limited space, the proof of Theorem 1 is provided in the full paper [21].

Theorem 2. *Suppose the BDH assumption holds and H_7, H_8, H_9, H_{10} are TCR hash functions, our TR-CPBRE scheme for n receivers is IND-CTCA-secure at original ciphertext.*

Please refer to the full paper [21] for the proof of Theorem 2.

5 Comparison

We compare our scheme with [14] and [17] in terms of property and efficiency. We first define the notations and parameters used in the Tables. Let t_p, t_e, t_v, t_s denote the computation cost of a bilinear pairing, an exponentiation, one verification and one signature of a one-time signature, respectively. Suppose [14], [17] and our scheme share the same number (N) of delegatees for each delegation.

Table 1 generally shows that our scheme achieves all properties with CCA security under the (P, Q, f)-GDDHE assumption. Specifically, [14] is RCCA secure under the n-$BDHE$ assumption [6], whereas our scheme is CCA secure and additionally supports timed-release property. Compared to [17], which is secure against RCCA under the 3-$QDBDH$ assumption [22], our scheme provides conditional delegation and broadcast re-encryption without losing CCA security. In conclusion, our scheme enables the delegator to implement a more fine-grained delegation of decryption rights in cloud storage systems where CCA security is required, but its security relies on random oracles. The problem of proposing a TR-CPBRE scheme with CCA security in the standard model remains open.

From Table 2, we see that [14] suffers from the largest number of pairings and [17] suffers from $O(N)$ complexity in pairings. Compared with [14], our TR-CPBRE requires one additional t_p in Enc and Dec, and $2t_p$ in $ReEnc$, respectively, but significantly reduces the number of pairings ($16t_p$) in Dec_R without requiring more pairings with regard to $ReKey$. As opposed to [17], our scheme achieves a constant cost of parings in $ReEnc$, and has the same number of pairings in the sum of cost of other metrics (except for $ReKey$). In conclusion, our scheme requires less number of pairings when compared with [14] and [17].

Table 2. Computation Cost Comparison

Schemes	Computation Cost				
	Enc	$ReEnc$	Dec	Dec_R	$ReKey$
CPBRE [14]	$(N+6)t_e + t_p$	$(N+3)t_e + 6t_p$	$2Nt_e + 8t_p$	$(4N-1)t_e + 20t_p$	$(N+7)t_e + t_p$
TR-PRE [17]	$t_s + 10t_e + 4t_p$	$(4t_e + t_v)N + 2Nt_p$	$5t_e + t_v + 5t_p$	$5t_e + t_v + 7t_p$	Nt_e
Our scheme	$(N+8)t_e + 2t_p$	$(N+1)t_e + 8t_p$	$(2N+4)t_e + 9t_p$	$(3N+9)t_e + 4t_p$	$(2N+9)t_e + t_p$

Table 3. Communication Cost Comparison

Schemes	Ciphertexts and Re-Encryption Key Length (Group of elements)		
	Re-Encrypted Ciphertext	Original Ciphertext	ReKey
CPBRE [14]	13 groups	5 groups	6 groups
TR-PRE [17]	11 groups	9 groups	1 group
Our scheme	11 groups	8 groups	6 groups

Table 3 shows that [14] has the smallest number of group elements in original ciphertext, while it suffers from the largest number of group elements in re-encrypted ciphertext. When compared with [14], our scheme reduces 2 group elements in re-encrypted ciphertext. Besides, the number of group elements of our scheme in the original ciphertext is 1 group less than that of [17]; meanwhile, both of schemes share the same number of group elements in re-encrypted ciphertext. However, in $ReKey$ our TR-CPBRE and [14] need 5 more groups compared to [17]. It is worth mentioning that, when compared with [17], [14] and our scheme enjoy better efficiency in communication as N increases.

6 Concluding Remarks

In this paper, we introduced a new variant of PRE, named TR-CPBRE, which achieves conditional delegation, broadcast re-encryption and timed-release property simultaneously. To the best of our knowledge, our TR-CPBRE is the first of its kind. We also showed that our scheme can be proved IND-sID-CCA secure in the random oracle model under the (P, Q, f)-GDDHE assumption. Moreover, when compared with the existing CPBRE and TR-PRE schemes, our scheme not only requires less number of pairings and achieves better efficiency in communication, but also enables the delegator to make a fine-grained delegation of decryption rights to multiple delegatees without losing CCA security.

This paper also motivates some interesting open problems, for example, how to construct a CCA-secure TR-CPBRE scheme in the adaptive identity model, i.e. achieving IND-aID-CCA security.

References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. 9(1), 1–30 (2006)

2. Attrapadung, N., Furukawa, J., Imai, H.: Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 161–177. Springer, Heidelberg (2006)
3. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
4. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. 36(5), 1301–1328 (2007)
5. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)
6. 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), http://dx.doi.org/10.1007/11535218_16
7. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 211–220. ACM, New York (2006), http://doi.acm.org/10.1145/1180405.1180432
8. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 185–194. ACM, New York (2007), http://doi.acm.org/10.1145/1315245.1315269
9. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
10. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and non-interactive timed-release encryption. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005), http://dx.doi.org/10.1007/11602897_25
11. Chalkias, K., Hristu-Varsakelis, D., Stephanides, G.: Improved anonymous timed-release encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 311–326. Springer, Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-74835-9_21
12. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Provably secure timed-release public key encryption. ACM Trans. Inf. Syst. Secur. 11(2) (2008)
13. Chow, S.S.M., Yiu, S.M.: Timed-release encryption revisited. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 38–51. Springer, Heidelberg (2008)
14. Chu, C.-K., Weng, J., Chow, S.S.M., Zhou, J., Deng, R.H.: Conditional proxy broadcast re-encryption. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 327–342. Springer, Heidelberg (2009)
15. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. IACR Cryptology ePrint Archive 2001, 108 (2001)
16. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007)
17. Emura, K., Miyaji, A., Omote, K.: A timed-release proxy re-encryption scheme. IEICE Transactions 94-A(8), 1682–1695 (2011)
18. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)

19. Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic construction of chosen ciphertext secure proxy re-encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 349–364. Springer, Heidelberg (2012)
20. Kikuchi, R., Fujioka, A., Okamoto, Y., Saito, T.: Strong security notions for timed-release public-key encryption revisited. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 88–108. Springer, Heidelberg (2012)
21. Liang, K., Huang, Q., Schlegel, R., Wong, D.S., Tang, C.: A conditional proxy broadcast re-encryption scheme supporting timed-release (full paper). Cryptology ePrint Archive, http://eprint.iacr.org/
22. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
23. Matsuda, T., Nakai, Y., Matsuura, K.: Efficient generic constructions of timed-release encryption with pre-open capability. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 225–245. Springer, Heidelberg (2010)
24. May, T.: Timed-release cryptography (February 1993), http://www.hks.net.cpunks/cpunks-0/1560.html (unpublished manuscript)
25. Nakai, Y., Matsuda, T., Kitada, W., Matsuura, K.: A generic construction of timed-release encryption with pre-open capability. In: Takagi, T., Mambo, M. (eds.) IWSEC 2009. LNCS, vol. 5824, pp. 53–70. Springer, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-04846-3_5
26. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Tech. rep., Cambridge, MA, USA (1996)

About Hash into Montgomery Form Elliptic Curves*

Wei Yu[1,2], Kunpeng Wang[2], Bao Li[2], and Song Tian[2]

[1] Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, 230027, China
yuwei_1_yw@163.com
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093

Abstract. Montgomery-form elliptic curves are widely used for efficient arithmetic calculations and immunity from timing attacks. Constructing hash function to hash messages into Montgomery-form elliptic curves is important, and this paper proposes four deterministic encoding algorithms to perform this transformation. One is based on finding a cube root, whereas the other three are based on finding square roots. We prove that the four algorithms are all hash functions. Moreover, we provide new functions indifferentiable from a random oracle based on our deterministic encodings.

Keywords: Hash, Random Oracle, Montgomery Form Elliptic Curves.

1 Introduction

Many algebraic curve cryptosystems require messages to be hashed into an algebraic curve. Boneh-Franklin's identity-based encryption scheme [1] proposes a one-to-one mapping f from the base field $\mathbb{F}_p$ to a particular supersingular elliptic curve. This enables us to construct a hash function $f(h(m))$ where $m \in \{0,1\}^*$ is a message and h a classical hash function. There are many other identity-based schemes that need messages hashed into an algebraic curve, such as encryption schemes [2,3,4], signature schemes [5,6,7,8], signcryption schemes [9,10], and Lindell's recent universally composable commitment scheme [11].

Password-based authentication protocols also require hashing messages into elliptic curves. The simple password exponential key exchange [12] (SPEKE) and the password authenticated key exchange [13] (PAK) protocols both require a hash algorithm to map the password into a point of the curve. Being probabilistic, the algorithms are susceptible to timing attacks [14]. Therefore, being able to hash into an elliptic curve within a constant number of operations would be beneficial.

* Supported in part by National Basic Research Program of China(973) under Grant No.2013CB338002, in part by National Research Foundation of China under Grant No. 60970153 and 61070171, and in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 147–159, 2013.

The algorithm in [1], which maps an element of $\mathbb{F}_{p^n}$ to an ordinary elliptic curve, is probabilistic and fails to return a point for a fraction 2^{-k} of the inputs, where k is a predetermined bound. One drawback of the algorithm is that the number of steps of the algorithm depends on the input u. Thus, the number of operations is not constant. If in practice the input u has to be secret, this circumstance may lead to a timing attack.

Various algorithms mapping $\mathbb{F}_{p^n}$ to an elliptic curve in deterministic polynomial time were reported by Shallue and Woestijne [15] in ANTS 2006 and T. Icart [16] in Crypto 2009. Shallue and Woestijne's algorithm is based on Skalba's equality [17] and uses a modification of the Tonelli-Shanks algorithm for computing square roots. This algorithm runs in time $\mathcal{O}(\log^3 q)$ when $q \equiv 3 \mod 4$ and in time $\mathcal{O}(\log^4 q)$ otherwise. T. Icart's algorithm is based on computing a cube root, which can be implemented in time $\mathcal{O}(\log^3 q)$ and in a constant number of operations over $\mathbb{F}_q$, when $q = p^n \equiv 2 \mod 3$. The algorithms encode an element of a finite field into Weierstrass-form elliptic curves. Later, hashing into twisted Edwards-form elliptic curves [18] and Hessian curves [19] were proposed.

There are some methods hashing into hyperelliptic curves such as [20,21]. The main idea is to compute square roots. A hash function from plaintext to C_{34}-curves by finding a cube root is constructed in [22].

Montgomery-form elliptic curves are widely used for efficient arithmetics and immunity of the timing attacks. Constructing hash functions that hash into Montgomery-form elliptic curves is a very urgent issue that falls within the context of finding general methods for hashing into elliptic curves of other forms according to birational equivalence.

In this paper, we describe four deterministic encoding algorithms to hash into Montgomery-form elliptic curves. The first deterministic algorithm is based on finding a cube root called the cube-root algorithm, which among the four algorithms that we proposed is very efficient. The second algorithm, called Legendre algorithm, is based on computing Legendre symbols; the third, called Shallue-Woestijne-Ulas (SWU) algorithm, is based on Skalba's equality [17]; and the fourth, the SWU algorithm, is the simplified SWU algorithm. We prove that the four algorithms are all hash functions. Finally, based on our deterministic encodings, we provide new functions indifferentiable from a random oracle.

2 Montgomery Form

In [23], Montgomery considered elliptic curves, defined over $\mathbb{F}_q$ where $q = p^n, p$ is a prime, $p > 3$, that can be written in what has since become known as Montgomery form:

$$E_{a,b} : by^2 = x^3 + ax^2 + x, \tag{1}$$

where $a, b \in \mathbb{F}_q$, b is the non-quadratic residue.

Montgomery originally introduced this type of curve for speeding up the Pollard and elliptic curve methods ECM for integer factorization. However, not all elliptic curves have the Montgomery-form, because the order of any elliptic curve

of Montgomery-form is divisible by 4. Recent ECC-standards [24] recommend that for cryptographic applications the cofactor of the elliptic curve should be no greater than four. The elliptic curve cryptosystems based on the Montgomery-form ECM : $by^2 = x^3 + ax^2 + x$ are immune to timing attacks using a randomized projective coordinate technique of K. Okeya, H. Kurumatani and K. Sakurai [25], where they also present an efficient algorithm for generating Montgomery-form elliptic curves with cofactor 4.

3 Cube Root Method

In this section, a deterministic encoding f_1 is constructed by finding a cube root. We call this method the cube-root algorithm. When $q = p^n \equiv 2 \mod 3$, the function

$$x \mapsto x^3$$

is a bijection with inverse function

$$x \mapsto x^{\frac{1}{3}} = x^{\frac{2p^n-1}{3}} = x^{\frac{2q-1}{3}},$$

which enables the construction of f_1 from $\mathbb{F}_{p^n}$ to a subset of elliptic curve $E_{a,b}$.

That is, $f_1(u) = (x, y)$ where

$$\begin{aligned} x =& \frac{\left(27bv^2 + (a - bu^2)^3\right)^{1/3} - a + bu^2}{3}, \\ y =& ux + v. \end{aligned}$$

$$\text{where } v = \frac{3 - (a - bu^2)^2}{6bu}. \tag{2}$$

It is easy to check that (x, y) satisfies the Montgomery equation $by^2 = x^3 + ax^2 + x$.

3.1 Properties of Cube Root Algorithm

Lemma 1. *Let $P(x, y)$ be a point on the curve $E_{a,b}$. The solutions u_s of $f_{a,b}(u) = P$ are the solutions of the polynomial equation:*

$$H_{a,b}(x, y) : b^2u^4 - (2ab + 6bx)u^2 + 6byu + a^2 - 3 = 0.$$

Proof: *In equation (2), $v = \frac{3-(a-bu^2)^2}{6bu}$. From the definition of f_1, we have*

$$\begin{cases} by^2 = x^3 + ax^2 + x \\ H_{a,b}(x, y) = 0 \end{cases} \Leftrightarrow \begin{cases} by^2 = x^3 + ax^2 + x \\ 6buy = 6bu^2x + 3 - (a - bu^2)^2 \end{cases}$$

$$\Leftrightarrow \begin{cases} by^2 = x^3 + ax^2 + x \\ y = ux + v \end{cases} \Leftrightarrow \begin{cases} b(u^2x^2 + 2uvx + v^2) = x^3 + ax^2 + x \\ y = ux + v \end{cases}$$

$$\Leftrightarrow \begin{cases} 3x^3 + 3x^2(a - bu^2) + x(a - bu^2)^2 = 3bv^2 \\ y = ux + v \end{cases}$$

$$\Leftrightarrow \begin{cases} (3x + a - bu^2)^3 = 27bv^2 + (a - bu^2)^3 \\ y = ux + v \end{cases} \quad \Leftrightarrow \begin{cases} x = \frac{\left(27bv^2 + (a - bu^2)^3\right)^{1/3} - a + bu^2}{3} \\ y = ux + v \end{cases}$$

■

The discriminant of $H_{a,b}(x, y)$: $\Delta_{a,b} = 144b^6(-27a^2x^4 - 27x^6 - 54ax^5 - 54x^4 + 18ax^3 + 117x^2 - 48 + 16a^2 + 24ax - 24a^2x^2 - 16a^3x^3)$ is not a square. Let D be the number of affine points $(x, y) \in E(\mathbb{F}_q)$ with $\Delta_{(a,b)}(x, y) = 0$. It is easy to check $D \leqslant 12$. The splitting field $\mathbb{L}_{a,b}$ of the irreducible separable polynomial $H_{a,b}$ is a Galois extension of $\mathbb{F}_q(E_{a,b})$. Let $g_{a,b}$ be the genus of the function field $\mathbb{L}_{a,b}$. From the Hurwitz formula, $g_{a,b} = 7$ (see appendix). Let $R_{a,b}(x, y, u)$ be the resolvent cubic of the quartic polynomial $H_{a,b}$. R_{ab} is $b^2u^3 + 4b(a + 3x)u^2 + 12(3x^2 + 2ax + 1)u + 36b^4y^2$.

The discriminant of R_{ab}: $\Delta_R = 144b^2(-48+16a^2-192ax-288x^2+648ab^3x^2+216a^2b^3x-432x^4-64a^3b^4+216b^3x^6+216b^4x^3-96a^2x^2-486b^7x^3-64a^3b^3x^3-64a^4b^3x+216ab^3x^5+216a^2b^3x^3+216ab^4x^2-144a^2b^3x^4-144a^3b^3x^2-144a^2b^4x+216ab^3x^4-486ab^6x^4-243a^2b^6x^2-486ab^7x-243b^8-243b^6x^6-576ax^3+64a^3x+144a^2x^4+64a^4x^2+192a^3x^3+216ab^3x^3+648b^3x^4+648b^4x+216ab^4)$ is not a square. Let C be a smooth projective curve whose function field is the quartic extension $\mathbb{L}_{a,b}$. Then, the Galois group of the quartic extension $\mathbb{F}_q(C)/\mathbb{F}_q(E)$ is S_4.

Using Theorem 3 in [26], for any nontrivial character χ of $E(\mathbb{F}_q)$, the character sum $S_f(\chi) \leqslant (2 \times 7 - 2)\sqrt{q} + 3$, which is shown in Appendix. u has three poles on C. Thus, f_{ab} is a well-distributed encoding.

3.2 One-Way

We prove the construction $H_1(m) = f_1(h(m))$ is one-way if h is one-way. We define one-way as follows:

Definition 1. *A hash function h is (t, ϵ) one-way, if any algorithm running in time t, when given a random $y \in im(h)$ as input, outputs m such that $h(m) = y$ with probability at most ϵ. If ϵ is negligible for any polynomial time t in the security parameter, then the hash function h is one-way.*

Proposition 1. *If h is a (t, ϵ) one-way function, then H_1 is (t', ϵ') one-way where $H_1(m) = f_{\cdot}(h(m))$ and $\epsilon' = 16\epsilon$.*

Proof: *In the proof of lemma 5 in [16], we take $L = 4$, then we have our proposition. If ϵ is negligible, then $\epsilon' = 16\epsilon$ is also negligible. Then if h is one-way, H_1 is one-way.* ■

3.3 Collision-Resistant

We prove that given a classical hash function h the construction $H_1(m) = f_1(h(m))$ is collision-resistant if h is collision-resistant. The definition of collision-resistant is:

Definition 2. *A hash function h is (t, ϵ) collision-resistant, if any algorithm running in time t, when given random messages m, m', returns $h(m) = h(m')$ with probability at most ϵ. If ϵ is negligible for any polynomial time t in the security parameter, then the hash function $h(m)$ is collision-resistant.*

Proposition 2. *If $h : \{0,1\}^* \mapsto \{0,1\}^k$ is a (t, ϵ) collision-resistant, then H' is a (t', ϵ') collision-resistant hash function where $H_1'(m) = f_1(c \circ h(m) + d)$ for $c, d \in \mathbb{F}_q$ selected randomly and*

$$\epsilon' = \epsilon + \frac{2^{2k+2}}{q}$$

Proof: *We use Theorem 3 [16] setting L is 4. We then have our proposition. In practice, ϵ is about 2^{-k}, then ϵ' is approximately 2^{-k+2} if the size of q is at least $5k/2$ bits. If ϵ is negligible, then $\epsilon' = 64\epsilon$ is also negligible. Then if h is collision-resistant, H_1' is collision-resistant.* ■

Let $c = 1, d = 1$, then $H_1(m)$ is collision-resistant.

4 Legendre Method

Let f be an odd monic polynomial over a finite field $\mathbb{F}_q$ with $q \equiv 3 \mod 4$, which has a simple root in $\overline{\mathbb{F}}_q$,

$$y^2 = f(x) = \frac{x^3 + x}{b}.$$

Let

$$f_2 : \mathbb{F}_q \to E_{0,b}.$$

$$u \mapsto \left(-\epsilon(u)u, \epsilon(u)\sqrt{\frac{\epsilon(u)f(u)}{-b}} \right), \ \epsilon(u) \neq 0$$

where $\epsilon(u) = \left(\frac{f(u)}{q}\right)$, and by $\left(\frac{\cdot}{q}\right)$ the Legendre symbol over $\mathbb{F}_q$. If $\epsilon(u) = 0$, $u \mapsto (u, 0)$.

Let $W \subset E_{0,b}$ be the set of $\mathbb{F}_q$-rational Weierstrass points on $E_{0,b}$, and $T \subset \mathbb{F}_q$ the set of roots of f_2.

Lemma 2. *[20] The function f_2 is well defined, maps all points in T to $(u, 0) \in W$, and induces a bijection $\mathbb{F}_q \backslash T \to E_{0,b} \backslash W$.*

Proof: *The proof of bijection $\mathbb{F}_q \backslash T \to E_{0,b} \backslash W$ is in [20]. If $\epsilon(u) = 0$, $u \mapsto (u, 0)$ is also a bijection and $(u, 0) \in W$, then F is a bijection $\mathbb{F}_q \to E_{0,b} \backslash \infty$.* ■

Lemma 3. *The cardinality of $E_{0,b}$ is $q+1$.*

Proof: *From the above, we obtain $\#E_{0,b} = \#(\mathbb{F}_q \backslash T) + \#W = q - \#T + \#W$. However, W consists of the point at infinity on $E_{0,b}$, and all points of the form $(x,0), x \in T$. Thus, $\#W = \#T + 1$, and $E_{0,b} = q+1$.* ■

It is easy to check that the function $f_2(h(m))$ is i) one way when h is one way, and ii) collision-resistant when h is collision-resistant.

5 SWU and Brief SWU

5.1 Equivalence with Weierstrass Form Elliptic Curves

Any elliptic curve is homogeneous with a Weierstrass-form elliptic curve.

The elliptic curve in Montgomery form is written

$$by^2 = x^3 + ax^2 + x.$$

Replacing $s = \frac{x}{b} + \frac{a}{3b}, t = \frac{y}{b}$, we obtain

$$t^2 = s^3 + \frac{3-a^2}{3b^2}s + \frac{2a^3 - 9a}{27b^3},$$

with the reverse map $(t,s) \mapsto (x,y)$:

$$x = bs - \frac{a}{3}, y = bt.$$

Next, we construct the deterministic encoding $u \mapsto (t,s), u \in \mathbb{F}_q$, and use the reverse map. Then, $f_3 : u \mapsto (x,y)$, called the SWU algorithm.

5.2 SWU Algorithm

Let

$$\begin{aligned} g(s) &= s^3 + \frac{3-a^2}{3b^2}s + \frac{2a^3-9a}{27b^3}. \\ X_1(u,r) &= r, \\ X_2(u,r) &= -\frac{b}{a}\left(1 + \frac{1}{u^4 g(r)^2 + u^2 g(r)}\right), \\ X_3(u,r) &= u^2 g(r) X_2(u,r), \\ U(u,r) &= u^3 g(r)^2 g(X_2(u,r)). \end{aligned}$$

Thus

$$U(u,r)^2 = g(X_1(u,r))g(X_2(u,r))g(X_3(u,r)). \tag{3}$$

From equation (3), at least one of $g(X_1(u,r)), g(X_2(u,r)), g(X_3(u,r))$ is a quadratic residue. Then, one of $X_1(u,r), X_2(u,r), X_3(u,r)$ is the abscissa of a point on the curve $t^2 = g(s)$.

If $g(X_1(u,r))$ is a residue, then $s = X_1(u,r), t = \sqrt{g(X_1(u,r))}$, else if $g(X_2(u,r))$ is a residue, then $s = X_2(u,r), t = \sqrt{g(X_2(u,r))}$, else $s = X_3(u,r), t = \sqrt{g(X_3(u,r))}$. Then

$$x = bs - \frac{a}{3},\ y = bt.$$

If $q \equiv 3 \mod 4$, $\sqrt{x}$ is simply an exponentiation $x^{\frac{1}{2}} = \pm x^{\frac{q+1}{4}}$. This map forms the basis of the modified SWU method in which the main idea is to compute square roots (see [15,21]). Then, our map is called an SWU map.

This map $u \mapsto (x,y)$ is denoted f_3. We simplify f_3 in the following.

5.3 Brief SWU

Note that knowing the value of r is not required in computing X_2, X_3 and U; indeed, these only depend on $g(r)$. For this reason, r does not have to be explicitly computed and we can take $g(r) = -1$, where -1 is a quadratic non-residue because $q \equiv 3 \mod 4$. Even if the value of r does not necessarily exist in $\mathbb{F}_q$, it exists in $\mathbb{F}_{q^3}$. With that value of r, the Ulas' formulae are still correct. Rewriting the SWU maps as a single variable with $g(r) = -1$ gives the following maps. Let

$$g(s) = s^3 + \frac{3-a^2}{3b^2}s + \frac{2a^3 - 9a}{27b^3},$$

where $ab \neq 0$, and

$$\begin{aligned} X_2(u) &= -\frac{b}{a}\left(1 + \frac{1}{u^4 - u^2}\right), \\ X_3(u) &= -u^2 X_2(u), \\ U(u) &= u^3 g(X_2(u)). \end{aligned}$$

Thus

$$U(u)^2 = -g(X_2(u))g(X_3(u)). \tag{4}$$

Therefore either $g(X_2(u))$ or $g(X_3(u))$ must be a quartic residue. This leads to the simplified SWU algorithm.

If $g(X_2(u))$ is a residue, then $s = X_2(u), t = \sqrt{g(X_2(u))}$ else $s = X_3(u), t = \sqrt{g(X_3(u))}$. Then

$$x = bs - \frac{a}{3},\ y = bt.$$

These maps are denoted by $f_4 : u \mapsto (x,y)$. Because $\deg X_2(t) = 4$ and $\deg X_3(t) = 4$, each equation then has at most four solutions; therefore a point has at most eight pre-images that can be efficiently computed.

5.4 One-Way

We prove that the construction $H_4(m) = f_4(h(m))$ is one-way if h is one-way.

Proposition 3. *If h is a (t,ϵ) one-way function, then H_4 is (t',ϵ') one-way where $H_4(m) = f_4(h(m))$ and $\epsilon' = 64\epsilon$.*

Proof: *In the proof of Lemma 5 in [16], we take* $L = 8$*, then we have our proposition. If* ϵ *is negligible, then* $\epsilon' = 64\epsilon$ *also can be negligible. Then if* h *is one-way,* H_4 *is one-way.* ■

5.5 Collision-Resistant

We prove that the construction $H_4(m) = f_4(h(m))$ is collision-resistant if h is collision-resistant.

Proposition 4. *If* $h : \{0,1\}^* \to \{0,1\}^k$ *is a* (t, ϵ) *collision-resistant, then* H_4' *is a* (t', ϵ') *collision-resistant hash function where* $H_4'(m) = f_4(c \circ h(m) + d)$ *for* $c, d \in \mathbb{F}_q$ *selected randomly and*

$$\epsilon' = \epsilon + \frac{2^{2k+2}}{q}$$

Proof: *We use Theorem 3 [16] for* L *is 8 and our proposition then follows. In practice,* ϵ *is about* 2^{-k}*, then* ϵ' *is approximately* 2^{-k+3} *when the size of* q *is at least* $5k/2$ *bits. If* ϵ *is negligible, then* $\epsilon' = 64\epsilon$ *also is negligible. Then if* h *is collision-resistant,* H_4' *is collision-resistant.* ■

Let $c = 1, d = 1$, the $H_4(m)$ is collision-resistant.

6 Indifferentiable from Random Oracle

6.1 First Construction

As a consequence, if $f : S \to \mathbb{G}$ is any weak encoding [27] to a cyclic group $\mathbb{G}$ with generator G, then the hash function $H_R : \{0,1\}^* \to G$ is defined by

$$H_R(m) = f(h_1(m)) + h_2(m)G,$$

where $h_1 : \{0,1\}^* \to \mathbb{F}_p$ and $h_2 : \{0,1\}^* \to \mathbb{Z}_N$ are two hash functions. $H_R(m)$ is indifferentiable from a random oracle in the random oracle model for h_1 and h_2.

We only need to prove f_1, f_2, f_4 are all weak encodings. It is easy to show: f_1 is an α_1-weak encoding from $\mathbb{F}_q$ to $E_{a,b}(\mathbb{F}_q)$, with $\alpha_1 = 4N/q$, where N is the order of $E_{a,b}(\mathbb{F}_q)$. Similarly, f_2 is an α_2-weak encoding from $\mathbb{F}_q$ to $E_{a,b}(\mathbb{F}_q)$, with $\alpha_2 = N/q$, and f_4 is an α_4-weak encoding from $\mathbb{F}_q$ to $E_{a,b}(\mathbb{F}_q)$, with $\alpha_4 = 8N/q$. As $\alpha_1, \alpha_2, \alpha_4$ are polynomial function of the security parameter, f_1, f_2, f_4 are all weak encodings. Thus, $H_{Ri}(m) = f_i(h_1(m)) + h_2(m)G, i = 1, 2, 4$ are all indifferentiable from a random oracle in the random oracle model for h_1 and h_2.

6.2 Second Construction

The second construction is:

$$H'_R(m) = f_1(h_1(m)) + f_1(h_2(m)).$$

Using Corollary 2 of [26]: f_1 is a well-distributed encoding, then $H'_R(m)$ is well behaved i.e. $H'_R(m)$ is indifferentiable from a random oracle.

In Section 3.1, we have proved that f_1 is a well-distributed encoding; hence, $H'_R(m)$ is indifferentiable from a random oracle, where h_1 and h_2 are regarded as independent random oracles with values in $\mathbb{F}_q$.

7 Time Complexity

7.1 Theoretical Analysis of Hash

Let M denote field multiplication, S field squaring, M_A multiplication by a constant number, I field inversion, E_C the cube root, E_S the square root, D a determination of the square residue, and K the security parameter. We make the usual assumptions that $E_C = E_B = \frac{K}{2}M, S = M_A = M, I = 10M$. f_1 needs $I + E_C + 2M + 3S + 3M_A = (\frac{K}{2} + 18)M$; f_2 needs $E_S + M_A + M + S + D = (\frac{K}{2} + 3)M + D$; f_4 needs $I + E_S + 4M_A + 2M + 2S + 1D = (\frac{K}{2} + 8)M + 1D$; Using Icart's function [16] with birational equivalence requires $[2S + 4M_A] + [I + 4M_A + 2M + 3S + E_C] + [2M + M_A] = I + E_C + 4M + 5S + 9M_A = (\frac{K}{2} + 28)M$. The cost of f_1 is $10M$ less than Icart's function with birational equivalence. f_4 is slower than f_1 because $|D|$ is greater than $15M$. If K is 196, f_1 is $\frac{10}{196/2+28} = 8\%$ faster than Icart's function.

7.2 Theoretical Analysis of Random Oracle

The construction: $H(m) = f(h_1(m)) + h_2(m)G$ needs one hash and one scalar multiplication. The second construction $H(m) = f(h_1(m)) + f(h_2(m))$ needs two hash times. The second method is more efficient.

7.3 Practical Implementations

The running time of various encodings into Montgomery-form elliptic curves is discussed. The implementation has been done with the NIST-recommended 192-bit prime $P192$ and the 384-bit prime $P384$ (Table 1).

Hash Time. We give the experimental results for the different hash methods f_1, f_2, f_4, and Icart's function with birational equivalence. We have run all items 1000 times and obtained the average values listed in Table 2. From Table 2, f_1 runs fastest, being in particular $\frac{0.181}{2.296} = 7.9\%$ faster than Icart's function.

Table 1. NIST

Prime	value	residue mod 3	residue mod 4
$P192$	$2^{192} - 2^{64} - 1$	2	3
$P384$	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$	2	3

Table 2. Time cost of different method on NIST

Prime	$P192$	$P384$
f_1	2.115	5.226
f_2	2.410	5.885
f_4	2.496	5.984
Icart's function	2.296	5.453

Random Oracle Time. The calculation time kP was 60 ms for $P192$ and 284ms for $P384$. The cost for method $H_{Ri}(m) = f_i(h(m)) + h_2(m)G, i = 1, 2, 4$ was about 60 ms for $P192$ and 290 ms for $P384$. The cost for method $H'_R(m) = f_1(h_1(m)) + f_1(h_2(m))$ was about 4 ms in $P192$ and 10 ms for $P384$. The second method, $H'_R(m)$, is clearly more efficient than the first, $H_{Ri}(m)$, $i = 1, 2, 4$.

8 Conclusion

This paper described four deterministic encoding algorithms $f_i, (i = 1, 2, 3, 4)$ for hashing into Montgomery-form elliptic curves. We also gave proof that the four algorithms determine hashing functions. Moreover, we provided new functions indifferentiable from a random oracle based on our deterministic encodings.

References

1. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
2. Baek, J., Zheng, Y.: Identity-based threshold decryption. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 262–276. Springer, Heidelberg (2004)
3. Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
4. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
5. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)

6. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
7. Cha, J.C., Cheon, J.H.: An identity-based signature from gap-Diffie-Hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2002)
8. Zhang, F., Kim, K.: Id-based blind signature and ring signature from pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002)
9. Boyen, X.: Multipurpose identity-based signcryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 383–399. Springer, Heidelberg (2003)
10. Libert, B., Quisquater, J.-J.: Efficient signcryption with key privacy from gap Diffie-Hellman groups. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 187–200. Springer, Heidelberg (2004)
11. Lindell, Y.: Highly-efficient universally-composable commitments based on the DDH assumption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 446–466. Springer, Heidelberg (2011)
12. Jablon, D.P.: Strong password-only authenticated key exchange. SIGCOMM Comput. Commun. Rev. 26(5), 5–26 (1996)
13. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
14. Boyd, C., Montague, P., Nguyen, K.: Elliptic curve based password authenticated key exchange protocols. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 487–501. Springer, Heidelberg (2001)
15. Shallue, A., van de Woestijne, C.E.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 510–524. Springer, Heidelberg (2006)
16. Icart, T.: How to hash into elliptic curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Heidelberg (2009)
17. Skalba, M.: Points on elliptic curves over finite fields. Acta Arith. 117, 293–301 (2005)
18. Yu, W., Wang, K.: How to Hash into Twisted Edwards Form Elliptic Curves. In: Lai, X. (ed.) Information Security and Cryptology, Inscrypt 2010, pp. 35–43 (June 1, 2011)
19. Farashahi, R.R.: Hashing into Hessian curves. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 278–289. Springer, Heidelberg (2011)
20. Fouque, P.-A., Tibouchi, M.: Deterministic encoding and hashing to odd hyperelliptic curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 265–277. Springer, Heidelberg (2010)
21. Ulas, M.: Rational points on certain hyperelliptic curves over finite fields. Bull. Polish Acad. Sci. Math. 55, impan, 97–104 (2007)
22. Yu, W., Wang, K., Li, B., Tian, S.: Construct Hash Function from Plaintext to C_{34} Curves. Chinese Journal of Computers 35(9), 1868–1873 (2012)
23. Montgomery, P.L.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Math. Comp. 48, 243–264 (1987)
24. Standards for Efficient Cryptography, Elliptic Curve Cryptography Ver.0.5 (1999), `http://www.secg.org/drafts.htm`
25. Okeya, K., Kurumatani, H., Sakurai, K.: Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 238–257. Springer, Heidelberg (2000)

26. Farashahi, R.R., Fouque, P.-A., Shparlinski, I.E., Tibouchi, M., Voloch, J.F.: Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. Math. Comp. 82, 491–512 (2013)
27. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010)

Appendix

Let $E_{a,b} : by^2 = x^3 + ax^2 + x$ be an elliptic curve of Montgomery form defined over finite field F_q, where $a, b \in F_q$, $b(a^2 - 4) \neq 0$. We extend $f_{a,b}$ to P^1, the projective line over F_q, by setting $f_{a,b}(\infty) = \infty$. Then the graph of $f_{a,b}$

$$G_{f_{a,b}} = \{(u, w) \in P^1 \times E_{a,b} | f_{a,b}(u) = w\}$$

is a closed subscheme of $P^1 \times E_{a,b}$ defined by

$$b^2u^4 - (2ab + 6bx)u^2 + 6byu + a^2 - 3 = 0.$$

We assume that char$F_q \neq 2, 3$. The geometry of $G_{f_{a,b}}$ can be described in the following lemma.

Lemma 4. *If $a^2 \neq 3$, $16a^6 - 144a^4 + 351a^2 - 108 \neq 0$, and $4a^4 - 28a^2 + 63 \neq 0$, the subscheme $G_{f_{a,b}}$ is a geometrically integral curve on $P^1 \times E_{a,b}$ with no singular point except the triple point at infinity. Its normalization C is a smooth, geometrically integral curve of genus 7. The natural map $\varphi : C \to G_{f_{a,b}} \xrightarrow{pr_2} E_{a,b}$ is a morphism of degree 4 ramified at twelve distinct finite points of $E_{a,b}(\bar{F}_q)$, with ramification index 2.*

Proof: *The affine patch of $G_{f_{a,b}}$ given by $u \neq \infty$ and $w \neq \infty$ can be represented as the algebraic set $\{(u, x, y) \in A^3 | b^2y - (x^3 + ax^2 + x) = 0, b^2u^4 - (2ab + 6bx)u^2 + 6byu + a^2 - 3 = 0\}$. Because $E_{a,b}$ is smooth, the Jacobian matrix*

$$\begin{pmatrix} 0 & -3x^2 - 2ax - 1 & 2by \\ 4b^2u^3 - 4b(a + 3x)u + 6by & -6bu^2 & 6bu \end{pmatrix}$$

is of rank less than 2 if and only if

$$\begin{cases} b^2y - (x^3 + ax^2 + x) = 0, \\ b^2u^4 - (2ab + 6bx)u^2 + 6byu + a^2 - 3 = 0, \\ 4b^2u^3 - 4b(a + 3x)u + 6by = 0, \\ 6bu(-3x^2 - 2ax - 1 + 2byu) = 0. \end{cases}$$

As $a^2 \neq 3$, $u \neq 0$ and $2byu = 3x^2 + 2ax + 1$. We substitute this into the second equation, then $bu^2 = a + 3x$. Thus, $x^3 + ax^2 + x = 3x^2 + 2ax + 1 = 0$, which is impossible. Therefore, there is no singular point in this affine patch.

Let v be the local coordinate $\frac{1}{u}$ on P^1 in a neighborhood of ∞ and $(z, t) = (\frac{1}{y}, \frac{x}{y})$ the local coordinate on $E_{a,b}$ in a neighborhood of ∞.

As $v = 0$ is not a root of equation $(a^2-3)v^4 + 6byv^3 - 2b(a+3x)v^2 + b^2 = 0$, there is no point (∞, w) with $w \neq \infty$. Next, we compute points of the form(u, ∞) with $u \neq \infty$. The point$(0,0,0)$ is the only one in the algebraic set $\{(u,z,t) \in A^3 | t^3 + at^2z + tz^2 - bz = 0, b^2u^4z - (2abz + 6bt)u^2 + 6bu + (a^2-3)z = 0\}$ of form $(u,0,0)$. The Jacobian matrix at $(0,0,0)$ is of rank 2, then $(0,\infty)$ is regular.

Let $\mathcal{O} = \frac{F_q[v,z,t]}{(t^3+at^2z+tz^2-bz)}_{(v,z,t)}$ be the local ring of $P^1 \times E_{a,b}$ at (∞,∞), $\mathfrak{m} = (v,z,t)$ the maximal ideal of $\mathcal{O}$. The local equation of $G_{f_{a,b}}$ in the neighborhood of (∞,∞) is given by $b^2z - 2b(az+3t)v^2 + 6bv^3 + (a^2-3)zv^4$, which is in $\mathfrak{m}^3 \backslash \mathfrak{m}^4$. Therefore, the multiplicity of (∞,∞) is 3.

From the above, the normalization C of $G_{f_{a,b}}$ is a smooth, geometrically integral curve. The morphism $C \to G_{f_{a,b}}$ is an isomorphism outside (∞,∞). The fiber over (∞,∞) consists of 3 points. Because $deg\varphi = 4 = \#\varphi^{-1}(\infty)$, φ is unramified at $\infty \in E_{a,b}$. The possible ramification points are those finite points of $E_{a,b}$, where $\triangle_{a,b} = 0$. The discriminant of $\triangle_{a,b}$ is $\triangle = 2^{53}3^{33}(a^2-4)^2(16a^6 - 144a^4 + 351a^2 - 108)$. We assume that $\triangle \neq 0$, then $\triangle_{a,b}$ has six simple roots. If $4a^4 - 28a^2 + 63 \neq 0$, the system $\{\triangle_{a,b} = 0, x^3 + ax^2 + x = 0\}$ has no solutions. Hence φ has twelve distinct ramification points on $E_{a,b}(\bar{F}_q)$. $H_{a,b}$ has nonzero second derivatives which leads to no triple point. It follows that all ramification points have ramification index 2. Therefore, the genus of C is $g(C) = 7$ by the Hurwitz formula. ■

Joint Triple-Base Number System for Multi-Scalar Multiplication*

Wei Yu[1,2], Kunpeng Wang[2], Bao Li[2], and Song Tian[2]

[1] Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, 230027, China
yuwei_1_yw@163.com
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093

Abstract. At present, the joint sparse form and the joint binary-ternary method are the most efficient representation systems for calculating multi-scalar multiplications $[k]P+[l]Q$, where k, l are scalars and P, Q are points on the same elliptic curve. We introduce the concept of a joint triple-base chain. Our algorithm, named the joint binary-ternary-quintuple method, is able to find a shorter joint triple-base chain for the sparseness of triple-base number systems. With respect to the joint sparse form, this algorithm saves 32% of the additions, saving 13% even compared with the joint binary-ternary method. The joint binary-ternary-quintuple method is the fastest method among the existing algorithms, which speeds up the signature verification of the elliptic curve digital signature algorithm. It is very suitable for software implementation.

Keywords: Elliptic Curve Cryptography, Multi-Scalar Multiplication, Hamming Weight, Joint Triple-Base Chain.

1 Introduction

Since there are no general-purpose sub-exponential algorithms known for the elliptic curve discrete logarithm problem, more and more attention has been focused on the use of elliptic curves in public key cryptography. In particular, increasing the speed of multi-scalar multiplication is significant as it is required in the signature verification of the elliptic curve digital signature algorithm (ECDSA).

Shamir's trick, which was actually introduced by Straus [1], speeds up the computation of the product of powers of two elements in the same group. His trick was designed for finite field arithmetic and is based on binary expansions of integers. This simple and powerful trick also works for speeding up the operation of multi-scalar multiplication, $[k]P + [l]Q$, on elliptic curves. If k and l are n-bit long integers, computing $[k]P$, $[l]Q$ separately and adding them together costs

* Supported in part by National Basic Research Program of China(973) under Grant No.2013CB338002, in part by National Research Foundation of China under Grant No. 60970153 and 61070171, and in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 160–173, 2013.

$2n$ doublings and n additions on average, whereas using Shamir's trick with one pre-computed point $P+Q$ costs only n doublings and $\frac{3}{4}n$ additions.

In 2001, Solinas [2] introduced the joint sparse form (JSF), which reduces the number of doublings and additions to n doublings and $\frac{n}{2}$ additions using two pre-computed points $P+Q$ and $P-Q$. Later, Adikari, Dimitrov and Imbert [3] gave a hybrid method for computing multi-scalar multiplication with a density of 0.3209 using 14 pre-computed points, where density is defined as the Hamming weight divided by n.

The double-base number system (DBNS) was first introduced by Dimitrov, Imbert and Mishra [4]. Doche, Kohel and Sica [5] described a joint double-base number system (JDBNS), a generalization of DBNS, for computing multi-scalar multiplication using the so-called joint binary-ternary method (JBT). They also introduced the concept of a joint double-base chain. The density of JBT using two pre-computed points $P+Q$ and $P-Q$ is approximately 0.3945. The density of generalized JBT with 10 pre-computed points is 0.3120, which is even lower than the density of the hybrid method with 14 pre-computed points.

The triple-base number system was first introduced by Mishra and Dimitrov [6], who used bases 2, 3 and 5 to compute scalar multiplication efficiently. Longa [7] used a multi-base non-adjacent form (mbNAF) to calculate scalar multiplication over prime fields in his master thesis. Although the Hamming weight of mbNAF is not as low, the method is very efficient. Recently, Purohit and Rawat [8] have calculated scalar multiplication efficiently using bases 2, 3 and 7.

In this paper, we introduce the joint triple-base number system, which is a generalization of the triple-base number system, to calculate multi-scalar multiplication by exploiting the sparseness and high redundancy of the representation integer pairs. The joint triple-base number system is able to represent integer pairs using fewer items than the JDBNS . We first introduce the concept of a joint triple-base chain (JTBC). As a special case, we use the 2,3,5-base with the joint binary-ternary-quintuple method (JBTQ), which calculates multi-scalar multiplication with a density less than 0.35. The Hamming weight is one of the most important factors influencing the speed of multi-scalar multiplication. The Hamming weight of Shamir's trick with one pre-computed point is $0.75n$, JSF with two pre-computed points is $0.5n$, JBT with two pre-computed points is approximately $0.3945n$, JBTQ with two pre-computed points is less than $0.35n$, the hybrid method with 14 pre-computed points is $0.3209n$, generalized JBT with 10 pre-computed points is approximately $0.3120n$ and generalized JBTQ with 10 pre-computed points is about $0.27n$. It is obvious that JBTQ has the lowest Hamming weight among JSF, JBT, JBTQ when using same number of pre-computed points. The Hamming weight of JBTQ with 10 pre-computed points is even lower than the hybrid method with 14 pre-computed points. Compared with JSF, this algorithm with two pre-computed points saves 32% of the additions and gains about 12% in overall multi-scalar multiplication performance using Jacobian coordinates on standard elliptic curves. Compared with JBT, this algorithm saves about 13% of the additions and gains about 5% in overall performance when both algorithms use two pre-computed points. JBTQ is a

simple and natural generalization of JBT, but we prove that it is more efficient and has almost the same cost as JTBC. We give efficient quintupling formulae for two types elliptic curves, and believe that there is room for improvement in these formulae. To our knowledge, the cost of JBTQ is less than that of existing algorithms.

2 Preliminary

2.1 Triple-Base Chain

Mishra and Dimitrov [6] introduced the triple-base number system motivated by its sparseness and the efficient computation of scalar multiplication. An integer z can be represented as a triple-base chain:

$$z = \sum_{i=1}^{m} s_i b_1^{e_{1i}} b_2^{e_{2i}} b_3^{e_{3i}},$$

where $s_i \in \{\pm 1\}$, $e_{1m} \geqslant e_{1(m-1)} \geqslant \ldots \geqslant e_{12} \geqslant e_{11} \geqslant 0$, $e_{2m} \geqslant e_{2(m-1)} \geqslant \ldots \geqslant e_{22} \geqslant e_{21} \geqslant 0$, $e_{3m} \geqslant e_{3(m-1)} \geqslant \ldots \geqslant e_{32} \geqslant e_{31} \geqslant 0$.

More specifically, using the 2,3,5-base, z is represented as

$$z = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i} 5^{q_i},$$

where $s_i \in \{\pm 1\}$, $b_m \geqslant b_{m-1} \geqslant \ldots \geqslant b_2 \geqslant b_1 \geqslant 0$, $t_m \geqslant t_{m-1} \geqslant \ldots \geqslant t_2 \geqslant t_1 \geqslant 0$, $q_m \geqslant q_{m-1} \geqslant \ldots \geqslant q_2 \geqslant q_1 \geqslant 0$.

2.2 Multi-Scalar Multiplication

Multi-scalar multiplications $[k]P + [l]Q$ are mainly required in signature verification for ECDSA. We use the same integers, $k = 542788 = (10000100100001000100)_2$, $l = 462444 = (1110000111001101100)_2$, as in [5]. To use Shamir's trick, k, l are written as

$$\binom{k}{l} = \binom{10000100100001000100}{01110000111001101100}.$$

Computing $[k]P$ and $[l]Q$ separately and adding them together costs 15 additions and 39 doublings, whereas 12 additions and 20 doublings are required using Shamir's trick.

Let $\bar{1}$ denote -1. The JSF [2] representation of k, l is

$$\binom{k}{l} = \binom{10000100100001000100}{10\bar{1}0001001\bar{0}1010\bar{1}0\bar{1}00}_{JSF}.$$

The JSF has the smallest density of all joint signed-binary representations for (k, l), and the computation costs 20 doublings and 9 additions.

The JBT representation [5] of k, l is

$$\begin{aligned}\binom{k}{l} = &\binom{1}{1}2^{11}3^5 + \binom{1}{\bar{1}}2^9 3^4 + \binom{0}{1}2^7 3^4 + \binom{1}{\bar{1}}2^7 3^3 \\ &+ \binom{0}{\bar{1}}2^5 3^3 + \binom{1}{1}2^5 3^2 - \binom{1}{1}2^5 3 \\ &+ \binom{0}{1}2^4 + \binom{1}{\bar{1}}2^2,\end{aligned}$$

and requires 8 additions, 11 doublings and 5 triplings.

2.3 Joint Triple-Base Chain

The joint triple-base number system (JTBNS) represents an integer pair (k, l) as

$$\binom{k}{l} = \sum_{i=1}^{m} \binom{d_i}{e_i} 2^{b_i} 3^{t_i} 5^{q_i}, \text{where } d_i, e_i \in \{0, \pm 1\}.$$

We define the density of a JTBNS expansion as the number of items in the expansion divided by the binary length of $\max(k, l)$. The exponents in the joint triple-base chain must satisfy $b_m \geqslant b_{m-1} \geqslant \ldots \geqslant b_2 \geqslant b_1 \geqslant 0$, $t_m \geqslant t_{m-1} \geqslant \ldots \geqslant t_2 \geqslant t_1 \geqslant 0$, $q_m \geqslant q_{m-1} \geqslant \ldots \geqslant q_2 \geqslant q_1 \geqslant 0$. The pair k, l above can be represented as:

$$\begin{aligned}\binom{k}{l} = &\binom{1}{1}2^{12}3^3 5^1 + \binom{0}{\bar{1}}2^{11}3^2 5^1 + \binom{\bar{1}}{0}2^8 3^2 5^1 \\ &+ \binom{1}{1}2^5 3^2 5^1 + \binom{0}{1}2^5 3^2 5^0 - \binom{1}{1}2^5 3^1 5^0 \\ &+ \binom{0}{1}2^4 3^0 5^0 + \binom{1}{\bar{1}}2^2 3^0 5^0,\end{aligned}$$

which requires 7 additions, 12 doublings, 3 triplings and 1 quintupling. In the following subsection, we present the cost of these elliptic curve point operations.

2.4 Cost of Elliptic Curve Point Operations

An elliptic curve E over a prime field $\mathbb{F}_p$, denoted by $E(\mathbb{F}_p)$ is defined by the Weierstrass equation [9]:

$$y^2 = x^3 + ax + b,$$

where $a, b \in \mathbb{F}_q$ and $\Delta = 4a^3 + 27b^2 \neq 0$.

We assume that $a = -3$, which is the parameter of the NIST standard elliptic curve, and take Jacobian coordinates. The point representation (x, y) is known as affine coordinates, while the form $(X : Y : Z)$ is known as projective coordinates and is inversion-free. Jacobian coordinates are a special case of projective

coordinates that have very efficient point operations. The equivalence class of a Jacobian projective point $(X : Y : Z)$ is

$$(X : Y : Z) = \{(\lambda^2 X, \lambda^3 Y, \lambda Z) : \lambda \in \mathbb{F}_p^*\},$$

and $(X : Y : Z)$ corresponds to the affine point $(X/Z^2, Y/Z^3)$.

Jacobi quartic curves are another form elliptic curve defined by the projective curve

$$Y^2 = X^4 + 2aX^2Z^2 + Z^4,$$

where $a \in \mathbb{F}_q$ and $a^2 \neq 1$. The projective point $(X : Y : Z)$ corresponds to the affine point $(X/Z, Y/Z^2)$.

Table 1 shows the cost of elliptic curve point operations with no stored values, as summarized by Longa and Gebotys [10], on standard elliptic curves using Jacobian coordinates (Jacobian) and on Jacobi quartic curves using an extended coordinate system of the form $(X : Y : Z : X^2 : Z^2)$ (JQuartic). Finding the remainder, doubling (2P), tripling (3P), quintupling (5P) and mixed addition (P+Q) are denoted by D, T, Q and A respectively, where mixed addition means that one of the addends is given in affine coordinates [11]. Formulae for these point operations for Jacobian are shown in Appendix A, and can also be found in [10], while those for JQuartic can be found in [10] and [12]. Costs are expressed in terms of field multiplications (M) and field squarings (S). We make the usual assumption that 1S = 0.8M, and disregard field additions/subtractions and discard multiplications/divisions by small constants for simplification purposes.

Table 1. Cost of elliptic curve point operations

computation	Jacobian cost/M	JQuartic cost/M
A	7M+4S/10.2	6M+3S/8.4
D	3M+5S/7	2M+5S/6
T	7M+7S/12.6	8M+4S/11.2
Q	10M+12S/19.6	14M+4S/17.2

The new quintupling formulae are very efficient for both Jacobian and JQuartic. We can also compute $5P = 2(2P) + P$ with cost $2 \times 7 + 10.2 = 24.2M$ for Jacobian and $2 \times 6 + 8.4 = 20.4$ for JQuartic, or $5P = 3P + 2P$ with cost $7 + 12.6 + 15 = 34.6M$ for Jacobian and $6 + 11.2 + 10.2 = 27.4M$ for JQuartic. In the computation of $5P = 3P + 2P$, the addition is not a mixed addition but rather a full addition. The cost of a full addition is $11M + 5S$ for Jacobian and $7M + 4S$ for JQuartic. Thus, these quintupling formulae are efficient for both Jacobian and JQuartic.

In the next section, we give our main algorithm for computing a joint triple-base chain using bases 2,3 and 5.

3 Joint Triple Base Algorithm and Generalizations

Before giving the main algorithm, we introduce some notation. For a positive prime p, let $v_p(x)$ denote the p-adic evaluation of x and let $v_p(x, y)$ denote $\min(v_p(x), v_p(y))$ for an integer pair (x, y). We define $v_p(0) = \infty$.

Definition 1. *The gather $\mathcal{C}$ is the set of all pairs of positive integers (x, y) such that $v_2(x, y) = v_3(x, y) = v_5(x, y) = 0$. $(2, 3)$ is an element of $\mathcal{C}$.*

The JBTQ to find a JTBC is shown in Table 3. In JBTQ, we take two positive integers k and l as the input. The pair (k, l) is divided by $2^{v_2(k,l)}\ 3^{v_3(k,l)}5^{v_5(k,l)}$ to obtain $(x, y) \in \mathcal{C}$. We then use the function gain to find the coefficients d and e that maximize the factor g that has the form $2^b3^t5^q$ where b, t, q are non-negative integers. $(x - d_i, y - e_i)$ is divided by the gain g to obtain a new pair in $\mathcal{C}$. Thus, we can iterate the process until $x \leqslant 1$ and $y \leqslant 1$. The pair (x, y) is the coefficients of the last item in the JTBC representation.

The function gain is shown in Table 2, where we choose $(d, e) \in \{-1, 0, 1\}^2 \backslash (0, 0)$. In the function gain, we need to compute the 2,3,5-part of the gcd (greatest common divisor) 8 times, returning the largest value and the relevant coefficients. Computing the 2,3,5-part of the gcd does not require computing the gcd, and can be done as follows. Check whether both $x - i$ and $y - j$ are divisible by 2. If they are, then divide by 2 and repeat; else check whether both $x - i$ and $y - j$ are divisible by 3. If they are, then divide by 3 and repeat; else check whether both $x - i$ and $y - j$ are divisible by 5. If they are, then divide by 5 and repeat. The cost of gain will be considered in the cost of algorithm JBTQ.

We define the 2,3,5-part of gcd(0,0) to be ∞. Because JBTQ calls the function gain only when $x > 1$ or $y > 1$, the pair (0,0) will not appear in the call of JBTQ.

The cost of JBTQ, which also is the recoding cost, contains of two main parts: calling the function gain and the divisions in lines 3 and 6 of JBTQ. The product of all divisors is approximately equal to the larger of k and l. The division is not in fact needed, that is, the result of line 6 in JBTQ is contained in the computation of the 2,3,5-part of the gcd in the function gain. Thus the main cost of JBTQ is the cost of calling the function gain. In the function gain, we compute the 2,3,5-part of the gcd eight times, and select the largest as g. The product of $(2^{b_1}3^{t_1}5^{q_1})$ and all g is approximately equal to the larger of k and l. In total, there are at most $8n$ divisions with the divisor 2, 3 or 5. Thus the total cost of the recoding is about 8 binary expansions. That is, the recoding cost of JBTQ is trivial relative to multi-scalar multiplication.

A simple example of JBTQ is shown in Section 2.3 where $k = 542788$, $l = 462444$.

The generalizations of JBTQ are similar to the generalizations of JBT. One generalization is to choose $(d, e) \in \{0, \pm 1\ \pm 7\}^2 \backslash (0, 0)$. In this case, JBTQ requires 10 pre-computed points including $P \pm Q, 7P, 7Q, 7P \pm Q, P \pm 7Q, 7P \pm 7Q$. In the next section, we will give the density of this generalization, as well as analyses of the function gain and of the algorithm JBTQ for computing multi-scalar multiplication.

Table 2. Function gain

Input: Two integers x and y satisfying $(x, y) \in \mathcal{C}$
Output: Coefficients d and e, and the max gain g
1. $g = 0$, $d = 0$, $e = 0$
2. for i=-1 to 1
3. for j=-1 to 1
4. find z, b, t and q such that z is the 2,3,5-part of $\gcd(x - i, y - j)$ (except the case $i = j = 0$)
5. if $z > g$
6. $g = z$,$d = i$, $e = j$
7. return (g, d, e)

Table 3. Joint Binary-Ternary-Quintuple Algorithm

Input: Two integers k and l such that $k > 1$ or $l > 1$
Output: A joint 2,3,5 chain computing k and l simultaneously
1. $i = 1$
2. $b_1 = v_2(k, l)$,$t_1 = v_3(k, l)$,$q_1 = v_5(k, l)$
3. $x = k/(2^{b_1}3^{t_1}5^{q_1})$,$y = l/(2^{b_1}3^{t_1}5^{q_1})$
4. while $x > 1$ or $y > 1$ do
5. $(g, d_i, e_i) =$gain(x, y)
6. $x = (x - d_i)/g, y = (y - e_i)/g$
7. $i = i + 1$
8. $b_i = b_{i-1} + v_2(g)$,$t_i = t_{i-1} + v_3(g)$,$q_i = q_{i-1} + v_5(g)$
9. $d_i = x$,$e_i = y$
10. return $\left(\binom{d_i}{e_i}2^{b_i}3^{t_i}5^{q_i}\right)$, which is a triple-base chain

4 Analysis and Comparison

We analyze the complexity of the algorithm JBTQ for computing multi-scalar multiplication and compare this with other algorithms using the same number of pre-computed points.

4.1 Complexity Analysis

Let $S_{\alpha,\beta,\gamma}$ denote the set $[1, 2^\alpha 3^\beta 5^\gamma]^2$, where $[a, b]$ denote all integers from a to b. Let gain(x, y) denote the value g returned by the function gain. We determine the probability $p_{\alpha,\beta,\gamma}$ that gain$(x, y) = 2^\alpha 3^\beta 5^\gamma$ in the following part.

Lemma 1. *Given three integers α, β, γ, the cardinality of $\mathcal{C} \cap S_{\alpha,\beta,\gamma}$ is $2^{2\alpha+4}3^{2\beta}5^{2\gamma-2}$.*

Proof: *The cardinality of $S_{\alpha,\beta,\gamma}$ is $2^{2\alpha}3^{2\beta}5^{2\gamma}$. The cardinality of $\mathcal{C} \cap S_{\alpha,\beta,\gamma}$ is equal to $2^{2\alpha}3^{2\beta}5^{2\gamma} \times (1 - \frac{1}{4}) \times (1 - \frac{1}{9}) \times (1 - \frac{1}{25}) = 2^{2\alpha+4}3^{2\beta}5^{2\gamma-2}$.* ■

Lemma 2. *Let α, β and γ be three nonnegative integers. Suppose that* $\text{gain}(x, y) = 2^{\alpha}3^{\beta}5^{\gamma}$, *and that μ satisfies $2^{\mu} > 2^{\alpha}3^{\beta}5^{\gamma}$, ν satisfies $3^{\nu} > 2^{\alpha}3^{\beta}5^{\gamma}$ and ω satisfies $5^{\omega} > 2^{\alpha}3^{\beta}5^{\gamma}$. We have*

$$\text{gain}(x + i2^{\mu}3^{\nu}5^{\omega}, y + j2^{\mu}3^{\nu}5^{\omega}) = \text{gain}(x, y),\ \forall i, j \in \mathbb{Z}.$$

Proof: *It is easy to show that* $\text{gain}(x + i2^{\mu}3^{\nu}5^{\omega}, y + j2^{\mu}3^{\nu}5^{\omega}) \geqslant \text{gain}(x, y)$. *We will show that* $\text{gain}(x + i2^{\mu}3^{\nu}5^{\omega}, y + j2^{\mu}3^{\nu}5^{\omega}) > \text{gain}(x, y)$ *is impossible. Assume that* $\text{gain}(x + i2^{\mu}3^{\nu}5^{\omega}, y + j2^{\mu}3^{\nu}5^{\omega}) = 2^{\alpha_1}3^{\beta_1}5^{\gamma_1} > \text{gain}(x, y) = 2^{\alpha}3^{\beta}5^{\gamma}$. *Then there exist c, d satisfying*

$$\begin{aligned}
&v_2(x - c + i2^{\mu}3^{\nu}5^{\omega}) \geqslant \alpha_1 \text{ and } v_2(y - d + j2^{\mu}3^{\nu}5^{\omega}) \geqslant \alpha_1,\\
&v_3(x - c + i2^{\mu}3^{\nu}5^{\omega}) \geqslant \beta_1 \text{ and } v_3(y - d + j2^{\mu}3^{\nu}5^{\omega}) \geqslant \beta_1,\\
&v_5(x - c + i2^{\mu}3^{\nu}5^{\omega}) \geqslant \gamma_1 \text{ and } v_5(y - d + j2^{\mu}3^{\nu}5^{\omega}) \geqslant \gamma_1.
\end{aligned}$$

If $v_2(x - c) \neq v_2(i2^{\mu}3^{\nu}5^{\omega})$, then $v_2(x - c) \geqslant \alpha_1$. If $v_2(x - c) = v_2(i2^{\mu}3^{\nu}5^{\omega})$, then $v_2(x - c) \geqslant \mu$, and thus $v_2(x - c) = \min(\alpha_1, \mu)$. In the same way, we have $v_2(y - d) = \min(\alpha_1, \mu)$, and so $v_2(x - c, y - d) = \min(\alpha_1, \mu)$. We can also obtain $v_3(x - c, y - d) = \min(\beta_1, \nu)$ and $v_5(x - c, y - d) = \min(\gamma_1, \omega)$. Because μ satisfies $2^{\mu} > 2^{\alpha}3^{\beta}5^{\gamma}$, ν satisfies $3^{\nu} > 2^{\alpha}3^{\beta}5^{\gamma}$, ω satisfies $5^{\omega} > 2^{\alpha}3^{\beta}5^{\gamma}$ and $2^{\alpha_1}3^{\beta_1}5^{\gamma_1} > 2^{\alpha}3^{\beta}5^{\gamma}$, we have $\text{gain}(x, y) \geqslant 2^{\min(\alpha_1,\mu)}3^{\min(\beta_1,\nu)}5^{\min(\gamma_1,\omega)} > 2^{\alpha}3^{\beta}5^{\gamma}$, which is impossible. Thus, $\text{gain}(x + i2^{\mu}3^{\nu}5^{\omega}, y + j2^{\mu}3^{\nu}5^{\omega}) = \text{gain}(x, y)$. ■

Lemma 3. *The probability $p_{\alpha,\beta,\gamma}$ is bounded above by $\frac{1}{2^{2\alpha+4}3^{2\beta-2}5^{2\gamma-2}}$ for any nonnegative integers α, β, γ.*

Proof: *There are 30 integers in the interval $[1, 2^{\alpha+1}3^{\beta+1}5^{\gamma+1}]$ that are divisible by $2^{\alpha}3^{\beta}5^{\gamma}$. There are a total of 90 elements x_0 such that one of x_0, $x_0 - 1$ and $x_0 + 1$ is divisible by $2^{\alpha}3^{\beta}5^{\gamma}$. In the square $S_{\mu,\nu,\omega}$, the pairs having a gain equal to $2^{\alpha}3^{\beta}5^{\gamma}$ are of the form $(x_0 + i2^{\alpha+1}3^{\beta+1}5^{\gamma+1}, y_0 + j2^{\alpha+1}3^{\beta+1}5^{\gamma+1})$, where x_0, y_0 is one of the 90 elements above and $(i, j) \in [0, 2^{\mu-\alpha-1}3^{\nu-\beta-1}5^{\omega-\gamma-1} - 1]^2$. There are $2^{2(\mu-\alpha)}3^{2(\nu-\beta+1)}\ 5^{2(\omega-\gamma)}$ pairs at most, which we divide by the cardinality of $\mathcal{C} \cap S_{\mu,\nu,\omega} = 2^{2\mu+4}3^{2\nu}5^{2\omega-2}$. Thus $p_{\alpha,\beta,\gamma} \leqslant \frac{1}{2^{2\alpha+4}3^{2\beta-2}5^{2\gamma-2}}$.* ■

In Lemma 3, $\text{gain}(1, 1)$ is one of the few situations that the gain is not $2^{\alpha}3^{\beta}5^{\gamma}$ for all α, β, γ. The number of these items is negligible compared with $2^{2(\mu-\alpha)}$ $3^{2(\nu-\beta+1)}5^{2(\omega-\gamma)}$, and so equality in Lemma 3 cannot be guaranteed.

Theorem 1. *Let k, l be two integers with $\gcd(k, l)$ coprime to 30. The average density of JBTQ returned by Algorithm JBTQ is in the interval $[0.3374, 0.3494]$. The average values of the largest power of 2, 3 and 5 in the corresponding chain are approximately equal to $0.42 \log_2 k$, $0.23 \log_2 k$ and $0.09 \log_2 k$, respectively.*

Proof: *We use Lemmas 1 and 2 to determine the first probabilities $p_{\alpha,\beta,\gamma}$ and Lemma 2 to calculate the μ, ν, ω. As shown in Lemma 3, we need to investigate $2^{2(\mu-\alpha)}3^{2(\nu-\beta+1)}5^{2(\omega-\gamma)}$ pairs. Take $p_{1,1,1}$ for example. Using Lemma 2 to obtain $\mu = 5, \nu = 4, \omega = 3$, we need to investigate $2^8 3^8 5^4 = 1.04976 \times 10^9$ pairs.*

Let $\bar{\alpha}$, $\bar{\beta}$ and $\bar{\gamma}$ denote the average value of the largest power of 2, 3 and 5 respectively in the JBTQ expansion. To analyze the density of JBTQ and deduce

$2^{\bar{\alpha}}3^{\bar{\beta}}5^{\bar{\gamma}}$, *we give in Table 4 the probability* $p_{\alpha,\beta,\gamma}$ *that* $z = 2^{\alpha}3^{\beta}5^{\gamma}$ *is returned by the function* gain *for integers* $z < 32$. *For* $z \geqslant 32$, *the upper bound follows from Lemma 3.*

In fact, $p_{z\geqslant 32}$ *is equal to 1 minus the sum of the probabilities listed in Table 4.*

Table 4. Probability $p_{\alpha,\beta,\gamma}$ for integers $z = 2^{\alpha}3^{\beta}5^{\gamma} < 32$

number/probability	number/probability	number/probability
$1{:}p_{0,0,0} = 0$	$2{:}p_{1,0,0} = 0$	$3{:}p_{0,1,0} = \frac{4}{3^3}$
$4{:}p_{2,0,0} = \frac{5}{3^3}$	$5{:}p_{0,0,1} = \frac{41}{2^1 3^2 5^2}$	$6{:}p_{1,1,0} = \frac{653}{2^1 3^4 5^2}$
$8{:}p_{3,0,0} = \frac{11}{2^1 3^3 5^1}$	$9{:}p_{0,2,0} = \frac{521}{2^3 3^5 5^1}$	$10{:}p_{1,0,1} = \frac{953}{2^3 3^4 5^2}$
$12{:}p_{2,1,0} = \frac{71}{2^1 3^3 5^2}$	$15{:}p_{0,1,1} = \frac{41}{2^3 3^2 5^2}$	$16{:}p_{4,0,0} = \frac{4759}{2^5 3^5 5^2}$
$18{:}p_{1,2,0} = \frac{1639}{2^2 3^6 5^2}$	$20{:}p_{2,0,1} = \frac{7}{3^4 5^1}$	$24{:}p_{3,1,0} = \frac{71}{2^3 3^3 5^2}$
$25{:}p_{0,0,2} = \frac{73}{2^0 3^4 5^3}$	$27{:}p_{0,3,0} = \frac{300383}{2^5 3^7 5^4}$	$30{:}p_{1,1,1} = \frac{19}{2^4 3^3 5^1}$

$p_{z\geqslant 32} = 0.0855128$.

Let K *denote the average number of bits eliminated at each step of JBTQ. Then*

$$K = \sum_{\alpha=0}^{\infty}\sum_{\beta=0}^{\infty}\sum_{\gamma=0}^{\infty} p_{\alpha,\beta,\gamma}(\alpha + \beta\log_2 3 + \gamma\log_2 5).$$

Let $z = 2^{\alpha}3^{\beta}5^{\gamma}$. *Defining*

$$K_{z\geqslant 32} = \sum_{z=32}^{\infty} p_{\alpha,\beta,\gamma}(\alpha + \beta\log_2 3 + \gamma\log_2 5),$$

we obtain

$$\begin{aligned} K =& \sum_{z=1}^{31} p_{\alpha,\beta,\gamma}(\alpha + \beta\log_2 3 + \gamma\log_2 5) + K_{z\geqslant 32} \\ =& (1.02105 + 0.579467\log_2 3 + 0.213216\log_2 5) + K_{z\geqslant 32} \\ =& 2.43456 + K_{z\geqslant 32}. \end{aligned} \tag{1}$$

It is obvious that

$$K_{z\geqslant 32} \geqslant p_{z\geqslant 32}\log_2 32 = 0.427564.$$

Thus, we obtain $K \geqslant 2.86212$.

Using Lemma 3, we determine an upper bound for $K_{z\geqslant 32}$ *of*

$$\begin{aligned} K_{z\geqslant 32} \leqslant& \sum_{z=32}^{\infty} \frac{1}{2^{2\alpha+4}3^{2\beta-2}5^{2\gamma-2}}(\alpha + \beta\log_2 3 + \gamma\log_2 5) \\ =& 0.24273 + 0.101364\log_2 3 + 0.0541211\log_2 5 \\ =& 0.529053. \end{aligned} \tag{2}$$

This gives $K \leqslant 2.96361$, *and thus* $2.86212 \leqslant K \leqslant 2.96361$. *The average density is the inverse of* K, *which is in the interval [0.3374,0.3494]. Thus, the*

Hamming weight of JBTQ is 0.3374n to 0.3494n for n-bit long integers k, l. Using the results above, it is easy to show that $\bar{\alpha} \in [1.02105, 1.26378]$, $\bar{\beta} \in [0.579467, 0.680831]$ and $\bar{\gamma} \in [0.213216, 0.2673371]$. The average of the largest power of 2 is equal to $\bar{\alpha}$ multiplied by the average length of the expansion, which is about $0.42 \log_2 k$. The same is true for the largest powers of 3 and 5, which come to about $0.23 \log_2 k$ and $0.09 \log_2 k$, respectively.

■

We can use more pre-computed points in JBTQ. As mentioned in Section 3, if we allow $d, e \in \{0, \pm 1, \pm 7\}$ as the coefficients, JBTQ requires 10 pre-computed points. Following the ideas of Theorem 1, the density of the modified JBTQ with 10 pre-computed points is approximately equal to 0.27, which is smaller than the density of JBT with 10 pre-computed points and of the hybrid method [3] with 14 pre-computed points, which are equal to 0.3120 and 0.3209 respectively.

4.2 Experimental Results

To compare JBTQ with other methods such as JSF and JBT, we present some experimental results. We make the comparison with JSF and JBT for integers ranging in size from 256 bits to 512 bits, using two or four pre-computed points. Even with one more pre-computed point, the performance of computing multi-scalar multiplication is improved greatly. We show $(d, e) \in \{-1, 0, 1\}^2 \backslash (0, 0)$ in Table 2. When we allow $(d, e) \in \left\{\{-1, 0, 1\}^2 \backslash (0, 0)\right\} \cup \{(7, 0), (-7, 0), (0, -7), (0, 7)\}$ in function gain, the modified JBTQ with four pre-computed points $P + Q, P - Q, 7P, 7Q$ is denoted by JBTQ_7. The costs of different point operations on elliptic curves are shown in Table 1. We refer to [10] for the explicit formulae for Jacobian and refer to [12,13] for the explicit formulae for JQuartic.

Assume that k, l are n-bit numbers. We run every item using the same 1000 random integer pairs for different methods, including JSF, JBT, JBTQ, Tree-JBT_5 [5] and JBTQ_7, and derive the average shown in Tables 5, 6 and 7 for different sizes of k, l. JBTQ_7 is an extension of JBTQ, and Tree-JBT_5 is an extension of JBT. The main difference between them is that JBTQ_7 uses one more base 5 than Tree-JBT_5. 5 is one of the bases of JBTQ, and we choose $7P$ and $7Q$ as the pre-computed points in JBTQ_7.

The notation $\#P$ is the number of pre-computed points, $\#A$ is the number of mixed additions, which is the average Hamming weight, $\#D$ is the number of doublings, $\#T$ is the number of triplings, and $\#Q$ is the number of quintuplings.

The data in Tables 5, 6 and 7 show that JBTQ saves 32% of the additions compared with JSF, and 13% compared with JBT. It is about 11% faster than JSF and 4% faster than JBT when each uses Jacobian coordinates on standard elliptic curves. When Tree-JBT_5 and JBTQ_7 use four pre-computed points, JBTQ_7 is about 3% faster than Tree-JBT_5.

In Tables 5, 6 and 7, setting $n = \#D + \#T \log_2 3 + \#Q \log_2 5$, the density $\frac{\#A}{n}$ is very significant to the performance of multi-scalar multiplication. When we use Jacobian coordinates, the cost of doubling is only about 7M which is very cheap, but the cost of quintupling is $19.6M > 7 \log_2 5M$. If the cost of

Table 5. Cost of multi-scalar multiplication for different methods at 256 bits

method	$\#P$	$\#A$	$\#D$	$\#T$	$\#Q$	Jacobian	JQuartic(M)
JSF	2	128.98	255.88	0	0	3106.76	2618.71
JBT	2	101.26	138.81	73.76	0	2933.9	2509.56
JBTQ	2	87.58	114.45	59.08	20.57	2842.05	2437.87
Tree-JBT$_5$	4	85.25	134.09	76.7	0	2774.6	2379.68
JBTQ$_7$	4	76.09	104.69	55.03	27.46	2740.54	2355.94

Table 6. Cost of multi-scalar multiplication for different methods at 384 bits

method	$\#P$	$\#A$	$\#D$	$\#T$	$\#Q$	Jacobian	JQuartic(M)
JSF	2	196.93	383.89	0	0	4695.92	3957.55
JBT	2	152.88	208.09	110.80	0	4412.09	3773.69
JBTQ	2	130.91	170.95	88.64	31.14	4259.14	3653.72
Tree-JBT$_5$	4	127.93	200.88	115.33	0	4164.2	3571.59
JBTQ$_7$	4	115.93	156.77	83.07	41.03	4130.75	3550.53

Table 7. Cost of multi-scalar multiplication for different methods at 521 bits

method	$\#P$	$\#A$	$\#D$	$\#T$	$\#Q$	Jacobian	JQuartic(M)
JSF	2	260.53	520.88	0	0	6303.57	5313.73
JBT	2	207.48	281.57	150.88	0	5988.37	5122.11
JBTQ	2	180.05	232.31	119.48	42.65	5804.07	4978.04
Tree-JBT$_5$	4	175.37	271.4	157.3	0	5670.55	4863.27
JBTQ$_7$	4	157.16	205.77	112.72	54.82	5538.17	4760.13

quintupling is $\log_2 5$ that of doubling and the cost of tripling is $\log_2 3$ that of doubling, then the total cost of JSF is $(15 \times 0.5 + 7)n = 14.5n$, that of JBT is $(15 \times 0.4 + 7)n = 13n$ and that of JBTQ is $(15 \times 0.35 + 7)n = 12.2n$. JBTQ gains even more: $1 - \frac{12.2n}{14.5n} = 16\%$ compared with JSF and $1 - \frac{12.2n}{13n} = 6\%$ compared with JBT.

5 Conclusion

In this paper, we introduced a new algorithm for computing multi-scalar multiplication called JBTQ. This algorithm uses two pre-computed points, saving 32% of the additions compared with JSF and 13% compared with JBT. It is about 12% faster than JSF and 5% faster than JBT when using Jacobian coordinates on standard elliptic curves. When both Tree-JBT$_5$ and JBTQ$_7$ use four pre-computed points, JBTQ$_7$ is about 3% faster than Tree-JBT$_5$. The algorithm JBTQ can greatly increase the speed of signature verification for ECDSA.

References

1. Straus, E.G.: Addition chains of vectors (problem 5125). American Mathematical Monthly 70, 806–808 (1964)
2. Solinas, J.A.: Low-weight binary representations for pairs of integers. Combinatorics and Optimization Research Report CORR 2001-41, University of Waterloo (2001)
3. Adikari, J., Dimitrov, V.S., Imbert, L.: Hybrid binary ternary number system for elliptic curve cryptosystems. IEEE Transactions on Computers 60, 254–265 (2011)
4. Dimitrov, V., Imbert, L., Mishra, P.K.: Efficient and Secure Elliptic Curve Point Multiplication using Double-Base Chains. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 59–78. Springer, Heidelberg (2005)
5. Doche, C., Kohel, D.R., Sica, F.: Double Base Number System for multi scalar multiplications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 502–517. Springer, Heidelberg (2009)
6. Mishra, P.K., Dimitrov, V.S.: Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication Using Multibase Number Representation. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 390–406. Springer, Heidelberg (2007)
7. Longa, P.: Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields, Master Thesis, University of Ottawa (2007)
8. Purohit, G.N., Rawat, A.S.: Fast Scalar Multiplication in ECC Using The Multi base Number System, `http://eprint.iacr.org/2011/044.pdf`
9. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer (2004)
10. Longa, P., Gebotys, C.: Fast multibase methods and other several optimizations for elliptic curve scalar multiplication. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 443–462. Springer, Heidelberg (2009)
11. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (1998)
12. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Faster group operations on elliptic curves. In: Australasian Information Security Conference (AISC 2009), Wellington, New Zealand. Conferences in Research and Practice in Information Technology (CRPIT), vol. 98, pp. 7–19 (January 2009)
13. Hisil, H., Wong, K., Carter, G., Dawson, E.: An Intersection Form for Jacobi-Quartic Curves. Personal Communication (2008)

Appendix A: The Formulae for Jacobian over Prime Fields

Elliptic curve formula:

$$Y^2 = X^3 + aXZ^4 + bZ^6, \text{ where } a = -3.$$

The following formulae can also be found in [10].

Doubling

$2(X_1, Y_1, Z_1) = (X_3, Y_3, Z_3)$ can be computed as follows:
$X_3 = A^2 - 2B, Y_3 = A(B - X_3) - 8Y_1^4, Z_3 = C$,
where $A = 3X_1^2 + aZ^4, B = 4X_1Y_1^2, C = 2Y_1Z_1$.
In standard elliptic curves, a=-3, so $A = 3(X_1 - Z^2)(X_1 + Z^2)$, $B = 4X_1Y_1^2$ and $C = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2$.
The total cost is 3M+5S.

Addition

$(X_1, Y_1, Z_1) + (X_2, Y_2, Z_2) = (X_3, Y_3, Z_3)$ is given by:
$X_3 = A^2 - 4B^3 - 8Z_2^2X_1B^2, Y_3 = A(4Z_2^2X_1B^2 - X_3) - 8Z_2^3Y_1B^3, Z_3 = BC$,
where $A = 2(Z_1^3Y_2 - Z_2^3Y_1), B = Z_1^2X_2 - Z_2^2X_1, C = 2Z_1Z_2 = (Z_1 + Z_2)^2 - Z_1^2 - Z_2^2$.
The total cost is 11M+5S.

Mixed Addition

$(X_1, Y_1, Z_1) + (X_2, Y_2) = (X_3, Y_3, Z_3)$, $Z_2 = 1$ is given by:
$X_3 = A^2 - 4B^3 - 8X_1B^2, Y_3 = A(4X_1B^2 - X_3) - 8Y_1B^3, Z_3 = BC$, where $A = 2Z_1^3Y_2 - Y_1, B = Z_1^2X_2 - X_1, C = 2Z_1B = (Z_1 + B)^2 - Z_1^2 - B^2$.
The total cost is 7M+4S.

Tripling

$(X_3, Y_3, Z_3) = 3(X_1, Y_1, Z_1)$ is given by:

$$\begin{aligned} X_3 =& 16Y_1^2(2B - 2A) + 4X_1D^2, \\ Y_3 =& 8Y_1[(2A - 2B)(4B - 2A) - D^3], \\ Z_3 =& 2Z_1D = (Z_1 + D)^2 - Z_1^2 - D^2, \end{aligned}$$

where

$$\begin{aligned} 2A =& 2CD = (C + D)^2 - C^2 - D^2, \\ 2B =& 16Y_1^4, \\ C =& 3X_1^2 + aZ_1^4 = 3(X_1 - Z_1^2)(X_1 + Z_1^2), \\ D =& 12X_1Y_1^2 - C^2 = 6[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4] - C^2. \end{aligned}$$

The total cost is 7M+7S.

Quintupling

$(X_5, Y_5, Z_5) = 5(X_1, Y_1, Z_1)$ can be computed as follows:

$$\begin{aligned} X_5 =& D^2 - 4E^3 - 8X_2'E^2, \\ Y_5 =& D(4X_2'E^2 - X_5) - 8Y_2'E^3, \\ Z_5 =& 2Z_2[(B+E)^2 - B^2 - E^2], \end{aligned}$$

where

$$\begin{aligned} A =& 3(X_1 + Z_1^2)(X_1 - Z_1^2), \\ B =& X_1' - X_2, \\ C =& 2Y_1' - 2Y_2, \\ D =& C^2 + E^2 - (C+E)^2 - 4Y_2', \\ E =& C^2 - 4B^3 - 3X_2', \\ X_1' =& 4X_1Y_1^2, \\ Y_1' =& 8Y_1^4, \\ X_2 =& A^2 - 2X_1', \\ 2Y_2 =& 2AB - 2Y_1' = (A+B)^2 - A^2 - B^2 - 2Y_1', \\ Z_2 =& 2Y_1Z_1 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2, \\ X_2' =& 4X_2B^2, \\ Y_2' =& 8Y_2B^3. \end{aligned}$$

The total cost is 10M+12S.

Anonymous Authentication of Visitors for Mobile Crowd Sensing at Amusement Parks

Divyan Munirathnam Konidala[1], Robert H. Deng[1], Yingjiu Li[1], Hoong Chuin Lau[1], and Stephen E. Fienberg[2]

[1] School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902
{divyanmk,robertdeng,yjli,hclau}@smu.edu.sg
[2] Department of Statistics, Machin Learning Department, Heinz College, and Cylab Carnegie Mellon University, Pittsburgh, PA, 15213-3890 USA
fienberg@stat.cmu.edu

Abstract. In this paper we focus on authentication and privacy aspects of an application scenario that utilizes mobile crowd sensing for the benefit of amusement park operators and their visitors. The scenario involves a mobile app that gathers visitors' demographic details, preferences, and current location coordinates, and sends them to the park's sever for various analyses. These analyses assist the park operators to efficiently deploy their resources, estimate waiting times and queue lengths, and understand the behavior of individual visitors and groups. The app server also offers visitors optimal recommendations on routes and attractions for an improved dynamic experience and minimized wait times. We propose a practical usable solution we call an anonymous authentication of visitors protocol that protects the privacy of visitors even while collecting their details, preferences and location coordinates; deters adversaries outside the park from sending in huge amounts of false data, which lead to erroneous analyses and recommendations and bring down the app server. We utilize queuing theory to analyze the performance of a typical app server receiving numerous simultaneous requests from visitors to process a core function of our protocol.

Keywords: Mobile crowd sensing, Amusement park, Anonymous authentication, False data, Partially blind signature scheme.

1 Introduction

Smart mobile devices allow users to download, install, and run mobile (software) applications (apps) that allow users to receive locations based services. For example, mobile apps use the Global Positioning System (GPS) sensor extensively to provide information specific to the users' current location. For mobile crowd sensing [12], [20], a mobile app periodically collects a device's sensor data from a large group of people and transmits them to the app server. Analyses at the server can provide location specific information directly to individuals and groups.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 174–188, 2013.

Amusement and theme parks, such as Disneyland, Universal Studios, and LEGOLAND, offer a variety of attractions, including rides, shows, dining, and other forms of entertainment. Since they attract a large number of visitors, the major inconvenience faced by visitors is long wait times, and park operators are always exploring and implementing ways and means to minimize/control queue lengths and enhance visitor experiences.

1.1 Current Situation

Park operators have several options to minimize visitors' wait times and to improve their overall experience. In the following, we briefly describe three most common approaches, where visitors' involvement is required.

Special/Express Pass Approach. Parks can sell special admission tickets, at a premium which allow purchasers to bypass the regular lines and gain priority entrance [22]. This approach may create a feeling of frustration when the special pass holders bypass those visitors who are waiting in the regular lines for a long time.

Timed Ticket Approach. The visitor inserts his/her admission ticket into a machine (located near the attraction) that issues a timed ticket with the return time window printed on it [9]. Now the visitor is free to enjoy the rest of the park and need to reach the attraction within the return time window. Since visitors cannot purchase such timed tickets, this approach gives any visitor inside the park equal opportunity to pick up a timed ticket. Typically timed tickets are issued in limited number; visitors must be able to physically reach the machine before they are all issued; otherwise they need wait in the regular lines. At any point of time, a visitor cannot possess more than one valid timed ticket and if the visitor fails to reach the attraction around his/her return time window, the timed ticket will be wasted. As a result this approach is neither scalable, flexible nor dynamic with respect to the total number of visitors and their movements in the park.

Mobile App Approach. Some park operators deploy mobile apps [23], [18], which display the map of the park, offer directions, provide information about promotions, shopping and dining options, and most importantly information about various attractions and their wait times. Such apps push notifications from the app server to the visitors' smart mobile devices. They do not generally aggregate the mobile crowd sensed data and visitors' personal preferences, nor analyze them in order to recommend optimal routes and attractions, for an enhanced dynamic visitor experience and to minimize wait times.

1.2 Application Scenario

Building upon the mobile app approach, we envision an application scenario, which is based on the mobile crowd sensing and would eliminate the drawbacks of the previously described approaches and offers other great benefits to both the park operators and their visitors.

Visitor Alice downloads, installs and runs a mobile app developed by the park operator. The mobile app prompts Alice to enter demographic details such as her age, gender, nationality, height (used to determine access to certain attractions), dietary restrictions and other health issues, as well as preferences for rides; must go and must skip attractions, etc. The app would not ask for Personal Identifying Information (PII) such as name, social security number, and address. The app then sends these details and preferences to the app server managed by the park operator. Henceforth, the app would also periodically collect Alice's current GPS location coordinates and send them to the app server, and Alice would receive communications from the server.

Benefits to the Visitors. Based on Alice's current location and her demographic details and preferences, the server would periodically and dynamically calculate an updated personalized itinerary for Alice to visit various attractions that minimize her wait times. Alice can follow the itinerary to visit an attraction during the recommended time slot, and tap her smart device or an RFID enabled device at the entrance/exit of the attraction for validation. Incentives (such as points that can be redeemed for gifts) can be used to encourage Alice to follow the recommended route.

With this approach, visitors need not physically visit a machine to obtain a return timed ticket. All visitors using the mobile app have equal opportunity to receive personalized time slots and recommended routes. If the server analyzes that a particular visitor, based on his/her current location cannot reach the attraction within his/her recommended time slot, the server would dynamically recalibrate new time slots for that visitor. Therefore this scenario is scalable, dynamic, and certainly not wastage prone.

Benefits to the Park Operators. By dynamically analyzing the visitors' crowd-sensed GPS data, the app server assists the park operators to manage and deploy their resources efficiently, manage traffic flows and congestion, analyze various key performance indices such as the queue information, and average/maximum wait times at each attraction, etc. Similarly by analyzing visitors' demographic details, preferences and activities, the app server also assists park operators to gain insight into the behavior of groups and individual visitors based common preferences, and background characteristics. This then allows for new approaches to meet visitor needs as well as dynamic optimal recommendations and routes to improve their overall experience in the park.

2 Threats and Security Requirements

From our application scenario, we identified the following threats to privacy and certain security requirements to alleviate these threats.

2.1 Threats

Visitor Privacy Violation. In our application scenario the mobile app gathers visitors' demographic details, preferences and current location coordinates;

therefore, we have to make sure that all these details do not reveal and cannot be linked to the true identities of the visitors; otherwise the app server or a hacker who has hacked into the app server, can generate detailed profiles of the visitors, their buying interests, track all their activities and current locations and carry out malicious acts such as identity fraud, stalking, and sending spam adverts.

False Data. Adversaries outside the park, for various malicious reasons (extortion, blackmail) can attempt to emulate the mobile app in a computer and create an unlimited number of fake virtual visitors with their location coordinates inside the park. The app sever might unsuspectingly consider all these fake virtual visitors to be actual visitors inside the park. Sending huge amounts of such false data to the app server would result in erroneous analyses and recommendations that could confuse and frustrate the visitors and also overwhelm, degrade and eventually bring down the server.

Greedy Visitors. Greedy park visitors could attempt to tamper with their smart mobile devices [15], to send false location coordinates in order to cheat the process, obtain unfair preferential treatment, rewards, and earlier time slots for the attractions of their choice.

Man-In-Middle-Attacks. The channel between the mobile app and the app server is potentially prone to eavesdropping, data capture, and data corruption by hackers. Since the channel is carrying potentially sensitive data, such attacks would violate visitors' privacy and lead to erroneous analyses at server.

2.2 Security Requirements

Use of Pseudonyms. Visitors must interact with the app server using pseudonyms to decouple visitors' data from their true identities.

Visitor Authentication. To prevent adversaries outside the park from supplying large volume of false data, the app server needs to verify whether the data it receives is indeed from a visitor inside the park. To accomplish this, the app server must authenticate the visitors inside the park and receive data from only such authenticated visitors.

Data Auditing. The app server needs to audit the data it receives in order to detect any anomalies or false data from greedy visitors.

Secure Communication Channel. The channel between the mobile app and the app server must be secure enough to provide app server authentication to the mobile app, and data protection and integrity for communications.

3 Proposed Anonymous Authentication of Visitors (AAV) Protocol

Nothing would stop an adversary outside the park from supplying a large volume of false data. Therefore, we need to authenticate the information reported by

a visitor inside the park without divulging the visitor' identity. In this paper we apply the cryptographic notion of anonymous authentication: authenticating the visitor without revealing his/her identity. The fundamental idea here is to interact with the visitor using a pseudonym instead of his/her true identity.

3.1 Naive Approaches

One naive approach would be to use the admission ticket ID as the visitor's pseudonym. The mobile app would send to the app server, the visitors' demographic details, preferences, and current location coordinates referring the ticket ID. Linking these details to the ticket ID proves that the visitor is genuine and is indeed inside the park. The adversary must purchase a sufficiently large number of valid admission tickets to launch a successful attack, but this would not be economical as the tickets are very expensive. This naive approach would greatly limit the adversary's power; however it does not protect the privacy of the visitor. A vast majority of the visitors buy their tickets using their credit cards. In which case the issued ticket IDs are recorded and linked to the credit card, which is in turn linked to the true identity of the visitor. Therefore, this naive approach does not truly address the requirement of anonymous authentication.

A second naive approach is for the app server to accept communications that come only through the Wi-Fi network of the park. Once again here, without authentication, nothing would prevent the adversary to use the park's Wi-Fi network to create unlimited number of fake virtual visitors. Furthermore, most of the visitors may hesitate using an unsecured open Wi-Fi network and instead prefer to use their own 3G/4G network.

3.2 Background

Our proposed AAV protocol utilizes pseudonyms and partially blind signature scheme.

Blind Signature. In 1982, David Chaum proposed a new cryptographic primitive called the blind signature [5], which could be used as a primer tool to design electronic payment and electronic voting schemes with user privacy-protection in mind. Blind signature is a special kind of digital signature [17], which allows users to get signatures on their messages from authorized entities/signature issuers (e.g. banks, trusted third parties) without revealing the message contents to the authorized entity. Furthermore, if malicious signature issuers and verifiers (e.g. service providers, merchants) collude, they cannot discover the real identity of the user who actually holds the signatures.

Partially Blind Signature. Blind signatures provide total privacy for users by fully hiding messages (to be signed) from the signer. This property is not desired from the signer's point of view, because he is responsible for his signatures and he needs to know what he would be signing on. To achieve a solution acceptable for both the signer and users, Abe and Fujisaki proposed the idea of partially

blind signature [1], which was later formally proved by Abe and Okamato [2]. A partially blind signature scheme has two portions: one portion consists of the message that is hidden by the user (as in blind signature scheme) and in the other portion, the signer can explicitly embed necessary information such as issuing date, expiry date, signer's identity etc. Users should be made aware of the information that the signer wishes to embed into the signature. Users must also be able to verify that only the agreed-upon information has been embedded by the signer; otherwise the signer may secretly embed undisclosed information into the signature that could reveal the true identity of the users at a later stage.

3.3 AAV Protocol Description

The AAV protocol is executed in two phases: "Certified Pseudonym Issuing Phase", followed by the "Subsequent Interaction Phase".

In the "Certified Pseudonym Issuing Phase", Alice's mobile app generates a pseudonym P and utilizes the partially blind signature scheme to hide P in a blinded message B. The mobile app then sends the ticket ID along with B to the app server. The app server verifies the validity of ticket ID, and inputs an expiry date while digitally signing B. As a result the app server has no clue about Alice's pseudonym and cannot link the future communications from this pseudonym to Alice. The mobile app would unblind the signature on B, in order to recover the signature S to the bare pseudonym P, thus making S the certified pseudonym by the app server.

In the "Subsequent Interaction Phase", the mobile app no longer uses the ticket ID, instead it uses the P and the S to send Alice's demographic details, preferences, current location coordinates. Since the signature on B from the app server has been unblinded to the bare pseudonym P, the app server can easily verify whether the pseudonym P sent by the mobile app matches with the pseudonym signed in the signature S and also whether it is within the expiry date. As a result the app server can make sure that it is communicating with a certified pseudonym/visitor inside the park.

Setup. We construct our protocol using the RSA-based partially blind signature scheme proposed by Abe and Fujisaki [1]. The mobile app and the app server share a secure one-way hash function $h(.)$ whose length is k bits. The app server executes RSA function as follows: N is a product of two large primes p and q. N satisfies $S_i \nmid \lambda$ for all prime $S_i (3 \leq S_i \leq 2^k - 1)$, where λ is the LCM of $(p-1)$ and $(q-1)$. The prime e is an RSA public component, which is larger than or equal to $2^k - 1$. The corresponding private key is d given by $ed = 1 mod \lambda$. The mobile app has the knowledge of e and N.

It is a known fact that a one day admission ticket would expire by the end of that day the visitor enters the park and a two day admission ticket would expire by the end of the second day of the visitor's visit to the park. Therefore, we assume that both the mobile app and the app server have the knowledge of the expiry date of an admission ticket. Let x be the expiry date of the admission ticket, whose length is $k-2$ bits and both the mobile app and the app server are

capable of calculating: $\tau(x) = 2^{k-1} + 2h(x) + 1$. $\tau(x)$ is a formatting function designed to keep its domain in $2^{k-1} < \tau(x) < 2^k$ so that $\tau(x_i)$ does not divide $\tau(x_j)$ where $i \neq j$. Also, it is designed to produce odd numbers only so that it becomes relatively prime with λ.

The mobile app can also generate pseudonyms of length k bits and the communication channel between the mobile app and the app server is secured via the standard HTTPS (Hypertext Transfer Protocol Secure) protocol [16].

Certified Pseudonym Issuing Phase depicted in the Fig.1 is self explanatory; however, we elaborate some of the steps here. Step 3 executes the blinding procedure of the partially blind signature scheme, which hides P in B. No one else other than the mobile app knows the value of P, i.e., the blinded message B is statistically or perfectly indistinguishable from P as long as blinding factor R is not revealed. Step 6 validates whether *tktid* is a valid unused ticket and has not been previously used to generate a partially blind signature. In Step 3, $e\tau(x)$ has become the public key that contains the common information between the mobile app and the app server, i.e., the expiry date. Therefore, in step 7, the app server calculates the corresponding private key d_x. Step 8 executes the

Alice's Mobile App	Secure Channel HTTPS	Park's App Server
Hash function: $h(.)$		Hash function: $h(.)$
		Large primes: p, q; $N = pq$
		$\lambda = \text{LCM}(p-1, q-1)$
		$ed = 1 mod \lambda$
Server's public key: e, N		Private key: d, N
Ticket's expiry date: x		Ticket's expiry date: x
$\tau(x) = 2^{k-1} + 2h(x) + 1$		$\tau(x) = 2^{k-1} + 2h(x) + 1$
1. Scan ticket barcode: *tktid*		
2. Generate pseudonym: P		
3. Randomly choose a blind factor: $R \in Z_N^*$		
4. blind(P): $B = h(P)R^{e\tau(x)} mod N$		
	5. $(tktid, B) \longrightarrow$	
		6. Validate: *tktid*
		7. Calculate private key: $d_x = 1/e\tau(x) mod \lambda(N)$
		8. ParBlindSign(B): $\Phi = B^{d_x} mod N$
	$\longleftarrow$ 9. (Φ)	
10. unblind(Φ): $S = \Phi/R \equiv P^{d_x} mod N$		
11. Verify: S using $e\tau(x)$		

Fig. 1. Certified Pseudonym Issuing Phase of AAV Protocol

signing procedure of the partially blind signature scheme to generate the blinded signature Φ. Step 10 executes the unblinding procedure of the partially blind signature scheme on Φ, which unblinds B to reveal P in the signature S. From here on S certifies the pseudonym P.

Subsequent Interaction Phase (Fig.2). In this phase, the mobile app sends the visitor's demographic details and preferences ($DetPre$), and current GPS coordinates (Gps) using the S, P, x. The app server computes $h(P)$ and verifies the signature on S using $x, h(P), d_x$, this validates that P has been indeed certified in S. With P being the reference index in the database, the app server records and analyzes the $DetPre$, and the periodic ($Time : Gps$) data, and $Rewards$ calculations. The server would now keep track and communicate with the mobile app using this P. The mobile app would receive the optimal route ($Route$), and the dynamically calibrated personalized time slots ($TSlots$) for various attractions ($Attrs$) in the park, as well as $Rewards$.

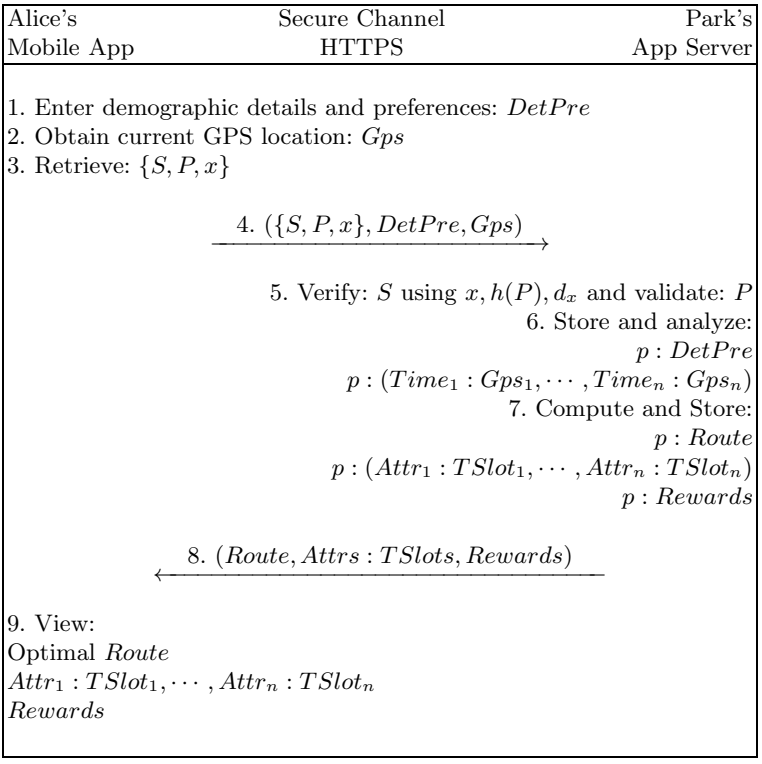


Fig. 2. Subsequent Interaction Phase of AAV Protocol

3.4 Anonymous Authentication of a Group

In our application scenario, the app server would assist the park operators to understand the behavior of groups moving together by constantly analyzing

the visitors' demographic details, preferences and activities; however, our AAV protocol is only applicable to individual visitors. Therefore, we extend the AAV protocol to accommodate anonymous authentication of a group. The basic idea here is that the pseudonyms of individual members of a group would all be linked to a single common group pseudonym. With this approach the app server can carry out behavioral analysis of individual members of a group based on their individual unique pseudonyms and also the behavioral analysis of the entire group based on their single common group pseudonym.

We can slightly modify the "Certified Pseudonym Issuing Phase", so that the head/leader of the group's mobile app would generate two pseudonyms; one representing the group pseudonym (GP) and the second representing his/her own individual pseudonym (P). Both the GP and P, would be hidden in the blinded message B. The head of the group would then inform the rest of the group members about the group pseudonym, e.g., via email. The rest of the group members' mobile apps would then include this GP along with their individual pseudonyms during their "Certified Pseudonym Issuing Phase". Finally, during the "Subsequent Interaction Phase", the app server would record both the group pseudonym and the individual pseudonyms.

4 Security Analysis

This section provides security analysis of our AAV protocol with respect to the threats described in section 2.

4.1 Use of Pseudonyms to Protect Visitors' Privacy

Our proposed AAV protocol successfully utilizes pseudonyms to decouple visitors' data from their true identities. The app server has no role in generating the pseudonyms for the visitors. The blinding procedure of the partially blind signature scheme does not reveal the visitor's pseudonym to the app server, yet the scheme allows us to obtain the signature of the app server on the pseudonym. The app server cannot link the pseudonym to neither the ticket ID nor the credit card used to purchase the ticket. Both the mobile app and the app server can independently produce $x, \tau(x)$, and $e\tau(x)$; therefore the mobile app can precisely verify (step 11 of the "Certified Pseudonym Issuing Phase") that apart from the expiry date x, the app server has not included any hidden message to distinguish the transaction later.

Restricted Privacy. Our AAV protocol provides only restricted privacy, but not complete visitor anonymity and unlinkability. Our application scenario requires that the visitor be tracked with a particular pseudonym, so that his/her preferences and current GPS location data can be gathered, analysed, and used to recommend optimal route, award rewards, and send dynamically calibrated personalized time windows for various attractions in the park.

Physical Layer Anonymity. Even though we use the AAV protocol, visitors may be tracked based on their smart device's MAC (Media Access Control) address, which is a unique fixed identifier assigned to network interfaces for communications on the physical network. However, if the visitor is using the device's 3G/4G network to communicate with the app server, the operator of the 3G/4G network would assign a different IP address each time a connection is made and the MAC address is made known only to the operator. The app server would only know the dynamic IP address, which cannot be used to track the device.

On the other hand if the visitor is using the free Wi-Fi network provided by the park operator, there are chances that the app server may retrieve and store the MAC addresses off the Wi-Fi access points. This situation is very rare and would require considerable amount of resources on the part of the app server to record the MAC address of every communication. However, there are ways to circumvent this problem; the visitor may choose to communicate with the app server through anonymity networks like the mix network [11], and Tor onion routing network [21]. Such networks direct user's internet traffic through a worldwide volunteer network of servers to conceal a user's location or guard against network surveillance or traffic analysis. There exists an open source client for the Tor network on Android mobile devices called the "Orbot" [19].

4.2 Visitor Authentication to Deter False Data from Adversaries Outside the Park

Our AAV protocol achieves anonymous authentication, whereby the app server accepts data only from pseudonyms that have been certified during the "Certified Pseudonym Issuing Phase". The function $\tau(x)$ prevents a visitor to obtain multiple signatures on the same pseudonym with different expiry dates [1]. The $h(P)$ in the step 4: blind(P): $B = h(P)R^{e\tau(x)} mod N$, prevents two visitors with valid certified pseudonyms to collude and forge a new valid certified pseudonym.

Uniqueness of Ticket ID. In our "Certified Pseudonym Issuing Phase" we depend on the ticket ID to be unique and non-sequential. The app server verifies whether the ticket ID is un-used, un-expired, and was not previously used to obtain the partially blind signature. But, if the park operators do not issue unique and non-sequential tickets, any one could produce and sell fake tickets and can also misuse our protocol. It is a problematic situation for both the park operators and for our protocol. In cases where the ticket ID is already on the ticket and not uniquely generated and printed at the time of issuing, we can expect sequential IDs. We suggest that the park operators generate and print another unique number (for example the current date and time) on every ticket at the time of issuing. In such a scenario, the mobile app would prompt the visitor to type in that unique number and send it along with the $(tktid, B)$, step 5 of "Certified Pseudonym Issuing Phase". This approach would ensure that we are dealing with unique and non-sequential ticket IDs.

4.3 Data Auditing: Heuristics, Thresholds, and Revocation to Deter Greedy Visitors

Following the approach of [15], which used heuristics to detect fake-location attacks against location-based services, we suggest that the app server would formulate and put in place certain heuristics such as calculating the time elapsed between the visitors' previous location and the current location. If this time matches with the average time taken by other visitors to commute between the same two locations, then the data is considered legitimate. Minimum threshold values must also be put in place, to detect false data. Whenever the app server identifies a particular pseudonym sending in data that does not match these heuristics and threshold values, it could be a greedy visitor, in which case the app server can immediately black list that particular certified pseudonym and deny all future communications.

4.4 Secure Channel to Counter Man-in-the-middle Attacks

In our AAV protocol, the communication channel between the mobile app and app server is secured using the standard HTTPS protocol [16]. The HTTPS authenticates the app server, and guarantees confidentiality and integrity for the data communicated between the mobile app and app server. The developers of the mobile app must carefully implement the HTTPS protocol; recent works [10], [13], have shown that improper implementations and over looking of various critical settings of HTTPS have resulted in complete breakdown of certificate verification, which can lead to successful man-in-the-middle attacks.

5 Related Work

There exist other cryptographic solutions such as the group signature schemes [7], and anonymous credential schemes [6], [8]. These schemes allow a member of a group to sign on a message on behalf of the group. The verifier of the signed message can prove that the message has come from the group, but cannot deduce the true identity of the group member who signed the message. At the outset these schemes seem suitable, but for the following reasons they are not practical for our application scenario. These schemes require all the visitors in the park to have a public and private key pair. They also require a group manager to add members into the group, issue group public key and to certify the group members' credentials. The group manager, if the need arises, can also trace and identify the member who signed a particular message. In our application scenario, the visitors enter the park in huge numbers in an unpredictable manner; we cannot expect the visitors to generate public and private key pairs; it is impossible for the visitors to establish a group manager among themselves and to execute complex operations of these signature schemes. The park operator cannot be a group manager, in which case the true identities of the visitors are revealed.

There also exist anonymous e-token schemes [4], where an user is initially issued a certain number of certified pseudonyms or anonymous e-tokens by the

server. At a later phase, every time the user communicates with the server, he/she uses a different anonymous e-token. This scheme does not reveal the true identity of the user, yet the server can confirm if the data has come from an authorized user possessing certified anonymous e-token. As mentioned in our security analysis section—Restricted Privacy—this scheme is also not practical for our application scenario because it provides complete anonymity and unlinkability of pseudonyms.

6 App Server Performance Results

In our proposed AAV protocol, the app server executes two core cryptographic procedures; the "partially blind signing procedure" (during the Certified Pseudonym Issuing Phase) and the "signature verification procedure" (during the Subsequent Communication Phase). Similar to the typical RSA-based signing and signature verification procedures [17]; both the RSA-based partially blind signing and signature verification procedures [1] include one hash operation and one exponentiation operation. We relied on [24] for the speed benchmarks of some of the most commonly used cryptographic algorithms. A 1024 bit RSA-based signing procedure and signature verification procedures take 0.67 milliseconds (ms) and 0.04 ms respectively, when executed on an AMD Opteron 8354, 2.2 GHz processor under Linux. We can assume that a 1024 RSA-based partially blind signing procedure and signature verification procedures would not take more than 0.67 ms and 0.04 ms respectively, when executed on an AMD Opteron 8354, 2.2 GHz processor under Linux.

Amusement parks attract large numbers of visitors. For example, the Magic Kingdom at Walt Disney World Resort, Florida, USA is the largest amusement park worldwide in order of annual attendance [3]; 17 million visitors in the year 2011, averaging 46,000 visitors a day. As a result, thousands of visitors' mobile apps would concurrently access the app server. Since a 1024 bit RSA-based partially blind signing procedure requires longer time (0.67 ms), when compared to the signature verification procedure (0.04 ms), we are particularly interested in the number of simultaneous partially blind signing requests that could be handled by an app server. Therefore we applied queuing theory [14] to analyze the performance of the app server by predicting its response times while executing the 1024 bit RSA-based partially blind signing procedure.

In queuing theory, a system consists of a single queue of jobs submitted to one or more servers. We used the stochastic $M/M/C$ queuing model, where the M represents the Markov or memory less or exponential nature of the job arrival and job service rates, and C is the number of servers attached to the queue. Initially we considered 1 app server, i.e., $M/M/1$ queuing model. A single server can process approximately 1,492 partially blind signing requests per second at the rate of 0.67 ms per request. We subjected this $M/M/1$ queuing model with increasing number of requests per second and calculated the respective server's average response time. The average response time is the sum of the average amount of time that it takes a server to process such a request, and the average amount of time a request spends in the queue.

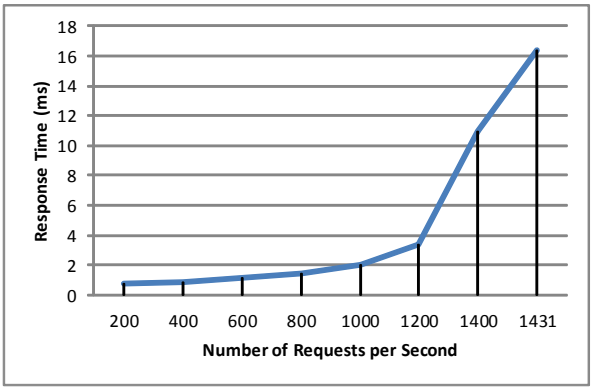


Fig. 3. Average Response Time of App Server with $M/M/1$ Queuing Model

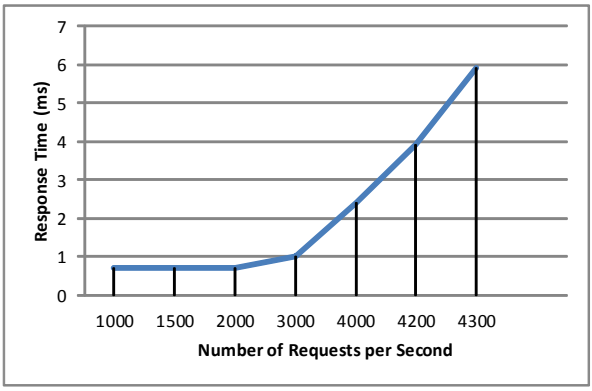


Fig. 4. Average Response Time of App Server with $M/M/3$ Queuing Model

Fig.3 depicts the chart of server's average response time with respect to the increasing number of requests per second. It shows that as the number of requests per second gets close to 1,492, the app sever becomes unstable, exponentially reaching: maximum utilization capacity and maximum number of requests that can be processed and are waiting in the queue, leading to a dead lock, and finally the average response time reaching one second per request, frustrating the visitors of the park. Fig.3 also points to the fact that the arrival rate of 1,000 requests per second allows the app server to be stable and process all the requests efficiently with an average response time of just 2 ms. However, considering the huge number of visitors in the park, we estimate that at least 3,000 (partially blind signing) requests per second must be processed by the app server. Therefore, we must increase the number of app servers, i.e., the value of

C in the $M/M/C$ queuing model. Our calculations as depicted via the chart in Fig.4 show that the $M/M/3$ queuing model, where a single queue of requests is now handled by 3 app servers, can efficiently process 3,000 requests per second, with an average response time of just 1 ms.

7 Conclusion

In this paper, we analyzed the authentication and privacy aspects of an application scenario that utilizes mobile crowd sensing for the benefit of amusement park operators and their visitors. We proposed a simple and practical anonymous authentication of visitors protocol that utilizes pseudonyms and partially blind signature scheme. The protocol protects the privacy of the visitors while authenticating them to the park's server and also prevents adversaries outside the park from bombarding the server with huge amounts of false data. We have utilized $M/M/C$ queuing model to analyze the server performance while receiving a large number of simultaneous partially blind signing requests (per second) from the visitors and recommended a minimum of 3 servers to handle 3,000 such requests per second for an optimal server response time. We offered several security discussions that need to be considered while deploying the application scenario. In fact, the contributions of this paper would be applicable to other types of similar application scenarios that are based on mobile crowd sensing and incentivizing the visitors.

Acknowledgement. This research/project is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

1. Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)
2. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (2000)
3. AECOM, TEA-AECOM 2011 Theme Index The Global Attractions Attendance Report, Themed Entertainment Association (TEA) (2011)
4. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clone wars: Efficient periodic n-times anonymous authentication. In: CCS 2006, pp. 201–210 (2006)
5. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO 1982, pp. 199–203 (1982)
6. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
7. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

8. Damgård, I.B.: Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 328–335. Springer, Heidelberg (1990)
9. Disney's FASTPASS Service, http://disneyworld.disney.go.com/guest-services/fast-pass/
10. Fahl, S., Harbach, M., Muders, T., Smith, M., Baumgartner, L., Freisleben, B.: Why Eve and Mallory love Android: An analysis of Android SSL (In)security. In: CCS 2012, pp. 50–61 (2012)
11. Berthold, O., Federrath, H., Köpsell, S.: Web mixes: A system for anonymous and unobservable internet access. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 115–129. Springer, Heidelberg (2001)
12. Ganti, R., Ye, F., Lei, H.: Mobile crowdsensing: Current state and future challenges. IEEE Communications Magazine 49(11), 32–39 (2011)
13. Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., Shmatikov, V.: The most dangerous code in the world: Validating SSL certificates in non-browser software. In: CCS 2012, pp. 38–49 (2012)
14. Gross, D., Shortle, J.F., Thompson, J.M., Harris, C.M.: Fundamentals of Queueing Theory. Wiley (2008)
15. He, W., Liu, X., Ren, M.: Location cheating: A security challenge to location-based social network services. In: ICDCS 2011, pp. 740–749 (2011)
16. Internet Engineering Task Force (IETF), Network Working Group, HTTP Over TLS, RFC2818 (2000), http://tools.ietf.org/html/rfc2818
17. Menezes, A.J., vaz Oorschot, P.C., Vanstone, S.A.: Digital Signatures. In: Handbook of Applied Cryptography, ch.11. CRC Press (1997)
18. Merlin Entertainments iTunes App., LEGOLAND California (2012), https://itunes.apple.com/us/app/legoland-california-official/id452395530
19. Orbot: Tor on Android, The Tor Project (2012), https://guardianproject.info/apps/orbot/
20. Sherchan, W., Jayaraman, P.P., Krishnaswamy, S., Zaslavsky, A.B., Loke, S.W., Sinha, A.: Using on-the-move mining for Mobile crowdsensing. In: MDM 2012, pp. 115–124 (2012)
21. Tor, Anonymity Online, https://www.torproject.org/
22. Universal Express Passes, Universal Orlando Resort, http://www.universalorlando.com/Theme-Park-Tickets/Universal-Express/Express-Passes.aspx
23. Walt Disney iTunes App., Disney Mobile Magic (2012), https://itunes.apple.com/us/app/disney-mobile-magic/id500000336
24. Dai, W.: Speed Comparison of Popular Crypto Algorithms, http://www.cryptopp.com/benchmarks.html

Secure RFID Ownership Transfer Protocols*

Nan Li[1], Yi Mu[1], Willy Susilo[1,**], and Vijay Varadharajan[2]

[1] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Wollongong, Australia
{nl864,ymu,wsusilo}@uow.edu.au
[2] Information and Networked Systems Security Research
Department of Computing, Faculty of Science
Macquarie University, Sydney, Australia
vijay.varadharajan@mq.edu.au

Abstract. An RFID tag could change hands many times during its lifetime. In a retail chain, the ownership of the tag is instituted by the supplier who initially owns the tag. In the view of a buyer, the validity of the current tag ownership and the originality of supplier are most important. In typical RFID ownership transfer protocols, the knowledge of the tag's authentication key proves the ownership. However, it is insufficient against an active attacker, since tags are usually lack of tamper-proof protections. Ownership transfer relies on a successful verification of tag's supplier and current ownership. In this paper, we formally define the security model of ownership transfer protocols and propose a secure ownership transfer protocol. In our scheme, current owner provides a new owner with the evidence of transfer and a proof of tag origin. Key management becomes easy in our system, since the one asymmetric verification key of the owner can be used to verify multiple tags that belong to the owner.

1 Introduction

A basic RFID system comprises three components: RFID reader, RFID tag, and backend database. RFID has exhibited many practical applications such as serving as identity of an object in supply chains, supermarkets and hospitals. A tag attached to a product has a unique identifier stored in its backend database. In practice, a product (with a tag) is owned by a user. Often, the product needs to change hands due to selling or buying. This process is referred to as ownership transfer.

In the lifetime of a tag, its ownership is likely to be transferred from one owner to another. An ownership transfer protocol runs between the current owner and the new owner. Generally speaking, the protocol is considered in two phases,

* This work is supported by the Australian Research Council Discovery Project DP110101951.
** This work is supported by ARC Future Fellowship FT0991397.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 189–203, 2013.

namely ownership verification and ownership transfer. A new owner firstly verifies the current ownership of the tag. If the current ownership is confirmed, he can request the ownership transfer. After a successful ownership transfer, the current owner who becomes the previous owner of the tag can no longer access the tag and the new owner who becomes the current owner of the tag can prove the ownership of the tag. According to the current ownership of the tag, a user can be a previous owner, current owner or new owner of the tag.

The security of RFID ownership transfer protocols is considered in threefold: the secure ownership, exclusive ownership and secure ownership transfer [3]. The first two are related to the phase of the ownership verification. Informally, they guarantee that the actual owner always has the ownership of a tag and no others can simultaneously obtain the ownership. The criteria of secure ownership transfer evaluates the phase of the tag ownership transfer. A new user who is unauthorized by the current owner cannot gain the ownership of the tag. A secure ownership transfer protocol should satisfy all these requirements.

The traditional RFID ownership transfer protocols are based on the lightweight symmetric key authentication schemes. The backend server and a tag share a predefined symmetric key and the tag's identity. The tag's ownership is checked by implementing the authentication protocol. However, most (passive) tags are not tamper-resistant, so that adversaries can launch active attacks. It is possible to physically corrupt or clone a tag and obtain the internal state. Once the internal state is leaked, the adversary can control the tag as the real owner. Therefore, it can prove the ownership and even transfer the tag to others.

1.1 Motivation

The aim of this paper is to propose a secure RFID ownership transfer protocol. In most previous RFID ownership transfer protocols, the proof of ownership relies on the knowledge of the tag's authentication key. If the user can provide a valid secret key, the verifier accepts its ownership of the tag. While it is insufficient against the attacker who compromises the tag. In practice, we call the party who currently owns the tag as a seller and the party who receives the ownership as a buyer. The symmetric authentication key shared between the seller and the tag provides no identity information of the seller. Anyone who has the key is able to prove the ownership and transfer it to other parties. It may injure the rights of seller and buyer. As a buyer, it usually concerns the origin of the product and the validity of the seller. He expects to check them during a purchase. The key management in large RFID system is also an issue. A tag normally requires a unique key for proving the ownership. The buyer has to obtain a large number of keys to check ownerships of tags. It not only requires a secure channel in communication, but also hard to maintain the records of these transactions. It would be desirable that one verification key can do the job. With this key, anyone can verify the ownership of tags that belong to the owner.

We look into an RFID system, where a supplier obtains products from a manufacturer. The supplier authorizes the manufacturer, via a warrant, to make specific products. After the products are ready for the supplier, the manufacturer

setups RFID tags and attaches them to products, respectively. When a buyer purchases the product from the supplier, the ownership transfer is required. The buyer checks the information of supplier and the product prior to making a payment. Once the deal is complete, the buyer owns the tag and supplier can no longer claim the ownership. Meanwhile, the seller provides the undeniable transfer proof which includes the information of seller, buyer and tag. The buyer can also resell the product in the future. One aim in this paper is to construct an ownership transfer scheme in this scenario.

Symmetric key based protocols are insufficient to reach a strong security level for ownership transfer protocols. It is a challenge to resist an active attack. We assume that the tag authentication can be done by using a traditional RFID authentication protocol, while we only focus on the ownership transfer protocol. An owner is usually a powerful entity which can perform public key cryptographic algorithms for ownership transfer, which does not rely on the computation power of tag.

1.2 Our Contributions

In this paper, we enhance the security of ownership transfer protocols by considering some strong attacks, such as the replacement attacks. In our model, the ability of an adversary is assumed by allowing more oracle queries. A formal definition of security model is given in this paper. An RFID ownership transfer protocol which is secure against the presented model is proposed. We consider a chain of the ownership transfers. It guarantees the actual ownership even if the internal state of the tag is disclosed. The protocol prevents an unauthorized owner from transferring the ownership to another. In other words, the validity of the current owner is verified during the ownership transfer. As a feature of our protocol, instead of using different authentication key to check the ownership of each tag, a buyer can use the seller's public key to check the all the tags. We analyse the security of proposed ownership transfer protocol and provide a formal security proof.

2 Related Work

Saito, Imamoto and Sakurai [13] introduced an ownership transfer protocol using two approaches. Both provide the privacy and security protection of the current owner and the new owner. One is based on the three-party model and the other is on two-party model. Since the schemes are based on symmetric key cryptographic algorithms, the secret key of tag is pre-shared with the owner. In the three-party model, the second key is shared between the trusted third party (TTP) and the tag. In ownership transfer, the TTP helps the new owner to update the tag's new secret. While the online TTP is required during the ownership transfer. Once the tag is compromised, the shared secret key between the tag and the TTP is also disclosed.

Independently, Molnar, Soppera and Wagner [9] proposed an ownership transfer protocol of RFID tags. The protocol addresses the privacy problems of ownership transfer through the *pseudonym*. The proposed scheme employs a tree based key structure to enable the *time-limited delegation* for temporarily ownership transfer. It is that the current owner can temporarily delegate the ownership of the tag to another party. After a period of time, the ownership is returned to the original owner without the agreement of the delegatee. However, the scheme needs a counter which is in the non-volatile memory to count the number of authentications. A *Trusted Center* (TC) who controls all the secret of tags assists the readers to authenticate the tag. Unfortunately, most trusted third party based ownership transfer protocols [6,8,11,9] suffer from the similar issues as in [13].

Several security properties of ownership transfer protocols were introduced by Ng, Susilo, Mu and Safavi-Naini [10], where they introduced four new properties: tag assurance, current ownership proof, undeniable ownership transfer and owner initiation. The proposed scheme satisfies most security properties of ownership transfer while only some hash calculations are required on the tag. Elkhiyaoui, Blass and Molva [4] presented the problem of issuer verification during the ownership transfer. In this paper, the privacy and security of ownership transfer protocols are formally defined and the proposed scheme achieves the constant time authentication. The scheme prevents the attacker from injecting fake tags in the supply chains. The origin of the tag is verified prior to the transfer. Abyaneh [2] shows that the forward and backward privacy are broken if the attacker was an owner of the tag. Additionally, the definition of the security model does not allow the adversary to rewrite the tag's content. It may be vulnerable against some active attacks.

A scalable authentication protocol which supports the ownership transfer was proposed in [5]. The protocol provides the controlled delegation without using the non-volatile memory to store a counter. The feature of desynchronization engages the protocol runs without the TTP. It employs a table which consists of two hash chains to identify a tag. While, the cost of storage on the server is questionable when the maximum size of the hash chains increased. Meanwhile, it also suffers form the denial-of-service attack.

Deursen, Mauw, Radomirović and Vullers [3] introduced a formal definition of secure ownership transfer in RFID systems. They described two roles: the *tag owner* and the *tag holder*. Basically, both of them can pass the ownership test but only the owner is engaged to transfer the ownership. It was claimed that the tag owner and holder are coincide in the notion of secure ownership. However, the holder of the tag may not be the owner in decentralized systems. Since the security of ownership is based on the authentication of the tag, most symmetric-key ownership transfer protocols [14,13,9,12] assume that the tag is incorruptible. In [10] and [4], a tag is allowed to be compromised. Nevertheless, the content of the tag cannot be rewrite after the adversary disclosed the key.

3 System Model

In this section, we formally define the ownership transfer protocols using the retail chain as an instance.

3.1 Entities

- **Tag T_i:** An object is attached by one tag T. The tag has a small memory which stores the current state s_i of the tag. T_i is a low-cost device which can at most calculate the hash function F.
- **Manufacturer M_i:** The manufacturer is the one who makes the products for suppliers. One manufacturer can cooperate with many different suppliers while the product must be authorized by the specific supplier.
- **Supplier S_i:** The supplier is the one who sales the products to customers. It handles the first ownership transfer of the tag. The supplier authorizes the manufacturer to produce expected number of products meanwhile S provides a unique warrant for each product.
- **Previous Owner $O_{(t_i,k-1)}$:** The previous owner $O_{(t_i,k-1)}$ is the one who previously owns the tag T_i at the time $k-1$. It provides the proof of transfer $\Sigma_{(t,k-1,k)}$ to the current owner.
- **Current Owner $O_{(t_i,k)}$:** The current owner $O_{(t_i,k)}$ is the one who currently owns the tag T_i at the time k. It maintains a database which stores the states of tags and authenticates tags though a reader R_k. The current owner can prove the current ownership $\sigma_{(t_i,k)}$ of the tag and show the valid transfer obtained from the previous owner. $O_{(t_i,k)}$ is allowed to transfer the current ownership of T_i to the new owner.
- **New Owner $O_{(t_i,k+1)}$:** The new owner $O_{(t_i,k+1)}$ is the one who is a potential owner of the tag T_i. Prior to accepting the ownership of tag T_i, the new owner verifies the tag's supplier S, the previous transfer proof $\Sigma_{(t,k-1,k)}$ and the current ownership $\sigma_{(t_i,k)}$. It provides an evidence of the acceptance once the transfer is completed.

Remark 1. The supplier can be considered as a special owner of tag and the manufacturer is an agent of particular supplier. The previous owner, current owner and new owner are roles which are changeable in different periods of the tag ownership. That means the new owner becomes a current owner or previous owner once he receives or transfers the tag ownership, respectively.

3.2 RFID Ownership Transfer Systems

In our system model, we do not employ the centralized server which is normally a trusted third party. Instead, we adopt the two-party mode that each party maintains an isolated database and readers. A party who engages in the ownership transfer is an owner of a tag. From now on, we refer to an owner as an entity which is supported by RFID readers and a backend database. While one owner has a public/private key pair where the public key is known to anyone. In the

model, we only need the secure communication channel during the authentication key exchange. Since the proposed scheme applies symmetric-key based tag authentication, it is impossible to securely update the key with shared secret [7]. The key update of the protocol should be performed outside the control range of the previous owner.

The ownership transfer system is described in Fig. 1. Different from the previous models, we consider the ownership of the tag as a chain. To handle an ownership transfer, the information of tag's supplier, previous owner, current owner and new owner are all required. Nevertheless, only the current owner needs to provide its secret.

In the model, the ownership transfer stems from the supplier. Let one owner be a level. Level 0 is the supplier of the tag. The manufacturer generates the proof of ownership under the supplier's warrant and stores it on the tag. Anyone who has the supplier's public key can verify the ownership of the product. In this level, the supplier simultaneously plays the role of the previous owner since the product is brand new. Then, it transfers the ownership to a new owner who is in level 1. Owner 1 accepts the ownership from the supplier and takes the role of the current owner. At this time, the supplier transferred the current ownership but remains the role of supplier and previous owner of the tag. Following the process, the ownership of the tag is generally in the k-th level.

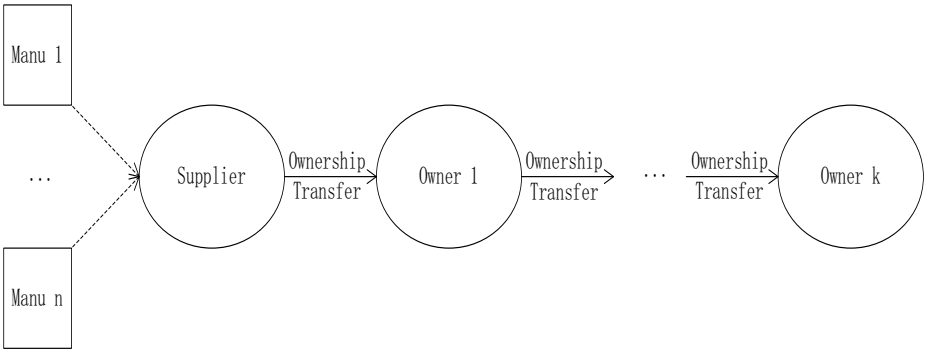


Fig. 1. Ownership transfer systems

A complete ownership transfer process has two phases: ownership verification and ownership transfer. In the ownership verification phase, the buyer checks the supplier of the tag, previous authenticated transfer proof and the validity of current ownership. Only if all the verifications are successful, two owners play the game of the ownership transfer. In the completion of an ownership transfer, the seller outputs a new authenticated transfer proof and the buyer outputs a proof of new ownership.

3.3 Ownership Transfer Protocols

An RFID ownership transfer protocol consists of seven algorithms: system setup (Setup), key generation (KeyGen), tag initiation (TagInit), authentication (Auth), ownership transfer (Transfer), ownership prove (OwnerProve) and ownership verification (OwnerVerify). The seven algorithms in RFID ownership transfer protocols are defined as follows.

- $params \leftarrow \mathsf{Setup}(\lambda)$: Taking as input a security parameter λ, outputs a set of public parameters $params$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(params)$: Taking as input the system parameters $params$, outputs a pair of public and private keys (pk, sk).
- $(c, \sigma_{(t,0)}) \leftarrow \mathsf{TagInit}(T, pk_s, sk_s, pk_m, sk_m)$: Taking as input a tag T, a pair (pk_s, sk_s) of supplier's public/private keys and a pair (pk_m, sk_m) of manufacturer's public/private keys, outputs the tag's initial state c and ownership proof $\sigma_{(t,0)}$. It runs between a manufacturer and a supplier.
- $Info \leftarrow \mathsf{Auth}(T, O_{(t,k)})$: Taking as input a tag T and the current owner $O_{(t,k)}$, outputs a set of information $Info$ of tag. It runs between the current owner and the tag.
- $\Sigma_{(t,k,k+1)} \leftarrow \mathsf{Transfer}(ID_t, pk_s, pk_{k-1}, pk_k, sk_{k+1}, \Sigma_{(t,k-1,k)})$: Taking as input a tag's identity ID_t, the public key pk_s of supplier, a pair of public/private key (pk_k, sk_k) of current owner and a new owner's public key pk_{k+1}, outputs an authenticated transfer proof $\Sigma_{(t,k,k+1)}$. It is run by the current owner.
- $\sigma_{(t,k)} \leftarrow \mathsf{OwnerProve}(ID_t, sk_k, \Sigma_{(t,k-1,k)}, \sigma_{(t,k-1)})$: Taking as input a tag's identity ID_t, a private key sk_k of current owner and an authenticated transfer proof $\Sigma_{(t,k-1,k)}$, outputs a proof $\sigma_{(t,k)}$ of ownership. It is run by the current owner.
- $\{true, false\} \leftarrow \mathsf{OwnerVerify}(ID_t, pk_s, pk_{k-1}, pk_k, \sigma_{(t,k)})$: Taking as input a tag's identity ID_t, the supplier's verification key pk_s, the previous owner's verification key pk_{k-1} and the current owner's verification key pk_k and a proof $\sigma_{(t,k)}$ of ownership, outputs $true$ if the proof is valid, outputs $false$ otherwise.

Without loss of generality, we describe the Auth algorithm in the protocol. While, it is unnecessary to the security of ownership transfer protocols. In the paper, Auth is assumed to be a privacy-preserving authentication protocol. The interaction of one ownership transfer is depicted as in Fig.2.

4 Proposed Protocol

The mathematical preliminaries and concrete construction of the proposed scheme are presented in the section.

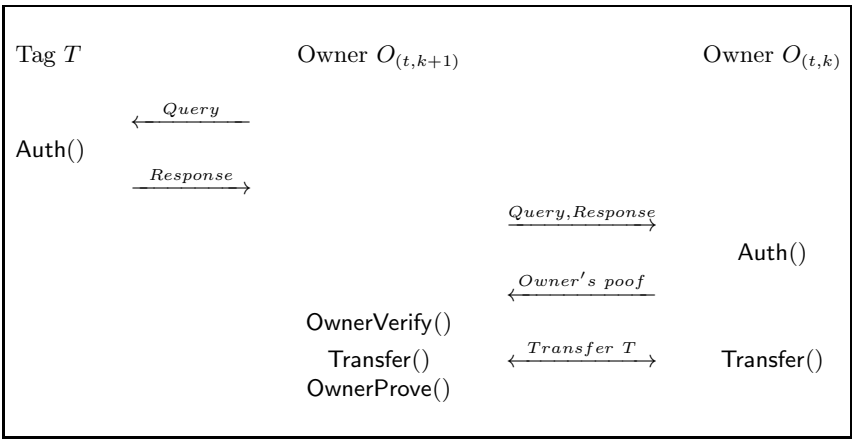


Fig. 2. Ownership transfer protocol

4.1 Preliminaries

Bilinear Maps. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three multiplicative cyclic groups of same prime order p. g and h are generators of group $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear mapping (pairing) and $(g, h, p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is a bilinear group. Let ψ be a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ that $\psi(h) = g$. We say it is a symmetric bilinear group if $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$. A bilinear pairing satisfies the properties as follows:

- **Bilinearity:** for all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$, we have the equation $e(g^a, h^b) = e(u, v)^{ab}$.
- **Non-Degeneracy:** for all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, if g, h are generators respectively, we have $e(g, h) \neq 1$ is a generator of $\mathbb{G}_T$.
- **Efficiency:** There is an efficient algorithm to calculate $e(g, h)$ for all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$.

Definition 1 (Computational Diffie-Hellman (CDH) assumption). *Let g be a generator of group $\mathbb{G}_1$. Given a tuple $< g, g^a, g^b >$, where $a, b \in_R \mathbb{Z}_p^*$, the CHD problem is to output $g^{ab} \in \mathbb{G}_1$. We say that the (ϵ, t)-CDH assumption is hold in $\mathbb{G}_1$, if no t-time algorithm $\mathcal{A}$ can solve the CDH problem in $\mathbb{G}_1$ with advantage at least ϵ.*

4.2 Construction

- Setup: Select a symmetric bilinear paring $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where the order of group $\mathbb{G}$ and $\mathbb{G}_T$ are the same p. Let $g, h \in \mathbb{G}$ be two generators. $H_1 : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p^*$, $H_2 : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p^*$ and $F : \{0,1\}^* \to \{0,1\}^l$, where l is a security parameter, are collision-resistant cryptographic hash functions. Sets the public parameters $params = (\mathbb{G}, \mathbb{G}_T, g, h, p, e, H)$.

- KeyGen: Randomly chooses $x \in \mathbb{Z}_p^*$ and sets the public/private key pair as $(pk, sk) = (x, g^x)$.
- TagInit: Let the public/private key pairs of a manufacturer M and a supplier respectively be $(pk_m, sk_m) = (x, g^x)$ and $(pk_s, sk_s) = (\alpha, g^\alpha)$. Firstly, the manufacturer and the supplier interacts as in Fig.3. The manufacturer generates an ownership proof σ_0 for the supplier. It randomly chooses an authentication key y from the key space $\mathcal{S}$ and sets the tag state $c = (y, F(\sigma_{(t,0)}))$. The supplier is the owner $O_{(t,0)}$.

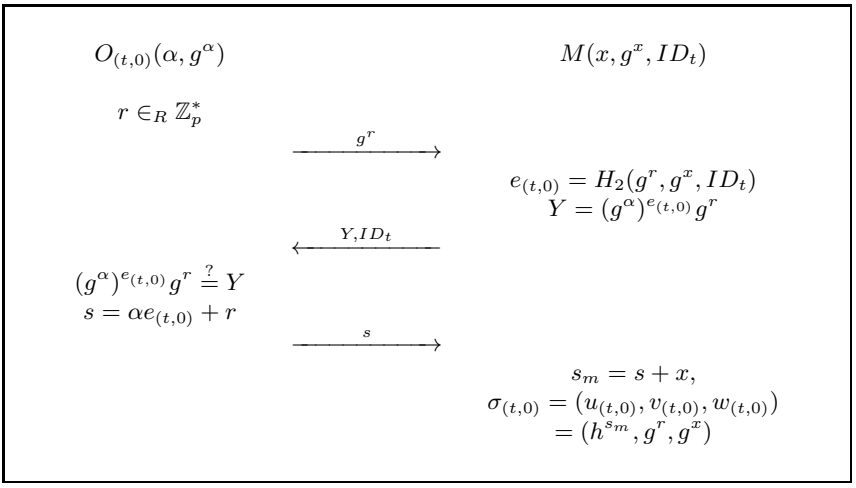


Fig. 3. Ownership initiation

- Auth: It is a general symmetric-key based authentication protocol. The current owner $O_{(t,k)}$ interacts the tag T using a pre-shared symmetric authentication key y. Once the authentication protocol outputs 1, the owner collects the tag's information $info$ which includes the tag's identity ID_t, ownership proof $\sigma_{(t,k)}$, etc.
- Transfer: To transfer the ownership, the current owner $O_{(t,k)}$ interacts with the new owner $O_{(t,k+1)}$. If the current owner is a supplier, it follows the description as in Fig.4. Otherwise, it follows the description as in Fig.5. Assume that the identity of tags and public information of two owners are mutually known.
- OwnerProve: To generate a proof of ownership, the current owner $O_{(t,k)}$ retrieves the proof $\Sigma_{(t,k-1,k)} = (s_{(t,k-1,k)}, R_{(t,k-1,k)})$ of authenticated transfer and the ownership poof $\sigma_{(t,k-1)}$ of owner $O_{(t,k-1)}$. Computes $s_{(t,k)} = s_{(t,k-1,k)} + sk_k$, where sk_k is the private key of $O_{(t,k)}$, and sets the proof $\sigma_{(t,k)} = (u_{(t,k)}, v_{(t,k)}, w_{(t,k)}) = (h^{s_{(t,k)}}, v_{(t,k-1)}, R_{(t,k-1,k)})$. In the case $k = 1$, set $v_{(t,1)} = g^{r_t}$, where g^{r_t} is from Fig.4.
- OwnerVerify: On input a proof $\sigma_{(t,k)} = (u_{(t,k)}, v_{(t,k)}, w_{(t,k)})$ of tag T, there are three cases. The verifier checks as follows

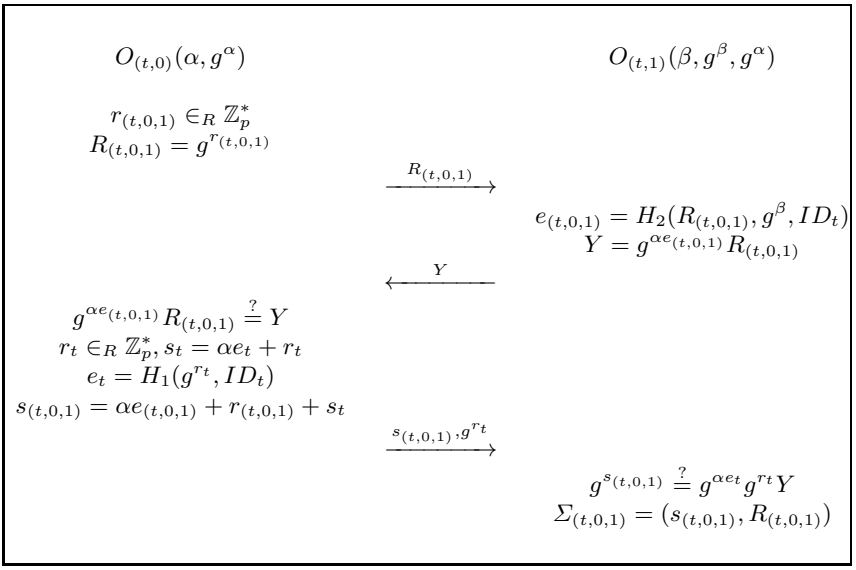


Fig. 4. Transfer from supplier to new owner

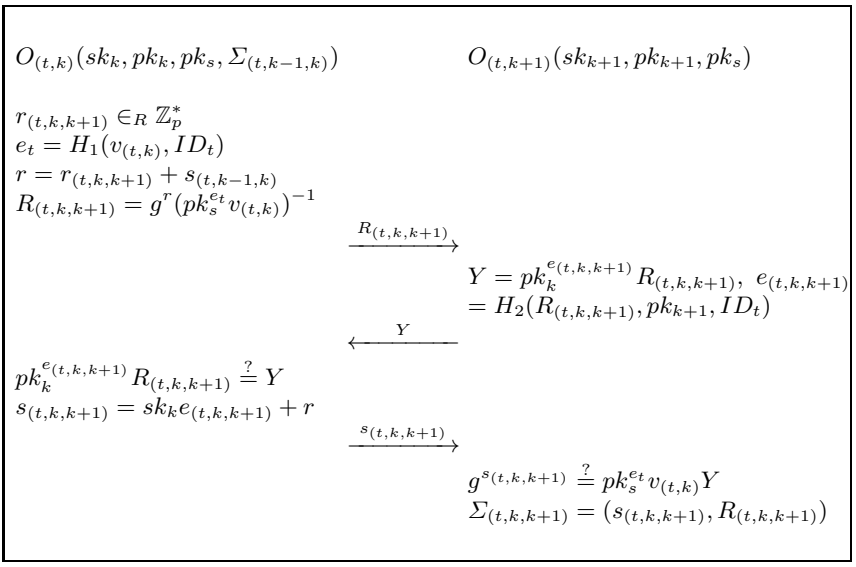


Fig. 5. General transfer from current owner to new owner on level k

- Case 1 ($k = 0$):

$$e_{(t,0)} = H_2(v_{(t,0)}, w_{(t,0)}, ID_t), \quad e(g, u_{(t,0)}) \stackrel{?}{=} e(pk_s^{e_{(t,0)}} v_{(t,0)} w_{(t,0)}, h).$$

- Case 2 ($k = 1$):

$$e_t = H_1(v_{(t,1)}, ID_t), \quad e_{(t,0,1)} = H_2(w_{(t,1)}, pk_1, ID_t),$$

$$e(g, u_1) \stackrel{?}{=} e(pk_s^{e_t + e_{(t,0,1)}} pk_1 v_{(t,1)} w_{(t,1)}, h).$$

- Case 3 ($k > 1$):

$$e_t = H_1(v_{(t,k)}, ID_t), \quad e_{(t,k-1,k)} = H_2(w_{(t,k)}, pk_k, ID_t),$$

$$e(g, u_{(t,k)}) \stackrel{?}{=} e(pk_s^{e_t} pk_k pk_{k-1}^{e_{(t,k-1,k)}} v_{(t,k)} w_{(t,k)}, h).$$

Outputs *true* if any equation holds, otherwise outputs $false$.

Correctness. Without loss of generality, we show the correctness of our RFID ownership transfer protocol in Case 3 as follows:

$$\begin{aligned}
&e(g, u_{(t,k)}) = e(g, h^{s_{(t,k-1,k)} + sk_k}) \\
&= e(g, h^{sk_{k-1} e_{(t,k-1,k)} + r + sk_k})) \\
&= e(pk_{k-1}^{e_{(t,k-1,k)}} g^r pk_k, h) \\
&= e(pk_s^{e_t} g^{r_t} pk_{k-1}^{e_{(t,k-1,k)}} g^r pk_k (pk_s^{e_t} g^{r_t})^{-1}, h) \\
&= e(pk_s^{e_t} pk_k pk_{k-1}^{e_{(t,k-1,k)}} v_{(t,k)} w_{(t,k)}, h).
\end{aligned}$$

5 Security Models of Ownership Transfer Protocols

The security of a RFID ownership transfer protocol usually relies on the underlining authentication protocols. It is extremely hard to provide the strong security if a symmetric-key authentication protocol is employed. Typically, the security model of symmetric-key based ownership transfer protocols does not provide corruption oracle which outputs the state of a tag. Once the key is exposed, the security of tag is completely compromised. Elkhiyaoui, Blass and Molva [1] recently presented a ROTIV protocol secure against the key corruption. It applies the public key cryptography in the authentication while the tag is only required to compute a hash function. However, the proposed security model cannot capture the adversary who can rewrite the content of a tag. It is possible when an adversary gains the key of tag. In this section, we enhance the security models of ownership transfer protocols. A general assumption is that owners are not able to launch collusion attacks in an ownership transfer [10].

5.1 Adversaries and Oracles

The ability of the adversary is essentially restricted by the actions that he is allowed to carry out. In security models, we specify the actions of adversary via the oracle queries. We now define the oracles which are used in the security models of ownership transfer protocols in this paper.

Definition 2 (Oracles). *The adversary plays with a challenger by given public information of the system and the following oracle calls.*

- $(O, pk) \leftarrow \mathsf{SetupOwner}(ID)$*: Taking as input an identity* ID*, it creates an owner* O *and runs the algorithm* KeyGen *to output a public key* pk*.*
- $T \leftarrow \mathsf{TagInit}(ID_t)$*: Creates a tag* T *with the identity* ID_t *and sets the authentication key* y*. It runs the algorithm* TagInit *and outputs the tag* T*.*
- $(ID_t, \sigma_{(t,k)}) \leftarrow \mathsf{Auth}(T, O_k)$*: Taking as input a current owner* O_k *and a tag* T*, it outputs the identity* ID_t *of tag and its ownership proof* $\sigma_{(t,k)}$ *if* T *is valid, outputs* $\perp$ *otherwise.*
- $c \leftarrow \mathsf{CorruptTag}(T)$*: Taking as input a tag* T*, and outputs the complete internal state* c *of* T*. Note that the oracle does not destroy the tag* T *and the tag is available in the future oracle calls.*
- $sk \leftarrow \mathsf{CorruptOwner}(ID)$*: Taking as input an owner's identity* ID*, and outputs the private key* sk *of the owner.*
- $\{0, 1\} \leftarrow \mathsf{Rewrite}(T, c', y)$*: Taking as input a tag* T*, a new state* c' *and an authentication key* y*, it rewrites the state by* c' *and outputs* 1 *if the key is valid,* 0 *otherwise.*
- $\sigma_{(t,k)} \leftarrow \mathsf{OwnerProve}(T, ID_s, ID_{k-1}, ID_k)$*: Taking as input a tag* T*, an identity* ID_s *of supplier, an identity* ID_{k-1} *of previous owner and an identity* ID_k *of current owner, it outputs an ownership proof* $\sigma_{(t,k)}$ *of the tag.*
- $\Sigma_{(t,k,k+1)} \leftarrow \mathsf{Transfer}(T, ID_k, ID_{k+1})$*: Taking as input a tag* T*, an identity* ID_k *of current owner and an identity* ID_{k+1} *of new owner, it outputs an authenticated ownership transfer proof* $\Sigma_{(t,k,k+1)}$ *of the tag.*

Definition 3 (Type I and Type II adversary). *The adversary is defined by the oracle calls and the goal of the experiment.*

- Type I Adversary*(*$\mathcal{A}_I$*): is also allowed to query all above oracles except the* CorruptOwner*. It aims to output a valid proof of authenticated transfer which cannot be detected during the transfer.*
- Type II Adversary*(*$\mathcal{A}_{II}$*): is allowed to query all above oracles. It aims to output a valid proof of ownership of the target tag which cannot be detected in the ownership verification.*

5.2 Security Models

We define the security models of ownership transfer protocols in this section. Each model captures the capability of different adversaries. A security model is defined as an experiment which plays between the adversary and the challenger.

We denote that the security parameters as r, s and n, which are respectively the number of owner initiations, the number of oracle calls and the number of tag initiations. There are two experiments defined in our security model. An RFID ownership transfer protocol is secure iff it is secure in both experiments. The security models defined in this section are suitable to ownership transfer protocols in the two-party model.

Security against Type I Attack. Type I adversary is a person who attempts to forge a valid proof of authenticated transfer. $\mathcal{A}_I$ interacts with the challenger via oracle calls and outputs a proof of transfer. It is described as in experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[r,s,n]$ in Fig.6.

Experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[r,s,n]$:

- **Setup:** The challenger runs the algorithm Setup to generate public parameters *params* and returns to $\mathcal{A}_I$. It initiates a supplier S^*.
- **Phase 1(Learning):**
  - $\mathcal{A}_I$ can query all above oracles except CorruptOwner to C.
  - Outputs two sets $\mathcal{T} = \{T_1, \ldots, T_n\}$ and $\mathcal{O} = \{O_1, \ldots, O_r\}$, which are created tags and owners.
- **Phase 2(Forge):**
  - $\mathcal{A}_I$ submits a target tag $\mathsf{T}^* \in \mathcal{T}$, current owner O_k^* and new owner O_{k+1}^* to $\mathcal{C}$, such that $(O_k^*, O_{k+1}^*) \in \mathcal{O} \cup \{S^*\}$.
  - $\mathcal{A}_I$ queries oracles Auth, CorruptTag, Rewrite, Transfer and OwnerProof to $\mathcal{C}$.
  - $\mathcal{A}_I$ outputs a proof Σ^* of authenticated ownership transfer.

Exp outputs success if $true \leftarrow \mathsf{OwnershipVerify}(ID_t^*, pk_s^*, pk_k^*, pk_{k+1}^*, \mathsf{OwnerProve}(ID_t^*, sk_{k+1}^*, \Sigma^*, \sigma^*))$, such that $\Sigma^* \nleftarrow Transfer(ID_t^*, ID_k^*, ID_{k+1}^*)$.

Fig. 6. Type I security experiment of the ownership transfer protocols

Definition 4. *An ownership transfer protocol is (r,s,n,ϵ)-secure against the Type I attack, if any $\mathcal{A}_I$ who succeeds in $\mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[r,s,n]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{secure}_{\mathcal{A}_I,\mathcal{S}}[r,s,n]] \leq \epsilon,$$

where ϵ is negligible.

Security against Type II Attack. The Type II adversary acts as a person who attempts to forge a valid proof of ownership. $\mathcal{A}_{II}$ interacts with the challenger $\mathcal{C}$ via oracle calls and outputs a proof of ownership at the end of the experiment. The experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[r,s,n]$ is defined as in Fig. 7.

Definition 5. *An ownership transfer protocol is (r,s,n,ϵ)-secure against the Type II attack, if any $\mathcal{A}_{II}$ who succeeds in $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[r,s,n]$ has advantage*

$$\Pr[success \leftarrow \mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[r,s,n]] \leq \epsilon,$$

where ϵ is negligible.

Experiment $\mathbf{Exp}^{secure}_{\mathcal{A}_{II},\mathcal{S}}[r,s,n]$:

- **Setup:**The challenger runs the algorithm Setup to generate public parameters *params* and returns to $\mathcal{A}_{II}$. It initiates a supplier S^*.
- **Phase 1(Learning):**
  - $\mathcal{A}_{II}$ can query all above oracles to $\mathcal{C}$.
  - Outputs two sets $\mathcal{T} = \{T_1, \ldots, T_n\}$ and $\mathcal{O} = \{O_1, \ldots, O_r\}$, which are created tags and owners.
- **Phase 2(Forge):**
  - $\mathcal{A}_{II}$ submits a target tag $\mathsf{T}^* \in \mathcal{T}$, previous owner O^*_{k-1} and current owner O^*_k to $\mathcal{C}$, such that $(\mathsf{O}^*_{k-1}, \mathsf{O}^*_k) \in \mathcal{O} \cup \{S^*\}$.
  - $\mathcal{A}_{II}$ queries oracles Auth, CorruptTag, Rewrite, Transfer and OwnerProve to $\mathcal{C}$.
  - $\mathcal{A}_{II}$ outputs a proof σ^* of ownership.

Exp outputs *success* if it satisfies the restrictions as follows,

1. $true \leftarrow \mathsf{OwnershipVerify}(ID^*_t, pk^*_s, pk^*_{k-1}, pk^*_k, \sigma^*)$,
2. $\sigma^* \nleftarrow \mathsf{OwnerProve}(\mathsf{ID}^*_t, \mathsf{ID}^*_s, \mathsf{ID}^*_{k-1}, \mathsf{ID}^*_k)$,
3. $sk^*_k \nleftarrow \mathsf{CorruptOwner}(ID^*_k) \ \vee \ (\Sigma^* \nleftarrow \mathsf{Transfer}(ID^*_t, ID^*_{k-1}, ID^*_k) \wedge sk^*_{k-1} \nleftarrow \mathsf{CorruptOwner}(ID^*_{k-1}))$.

Fig. 7. Type II security experiment of the ownership transfer protocols

Lemma 1. *If an ownership transfer protocol is secure against the Type II attack, it is secure against the Type I attack.*

Due to the page limitation, the proof of Lemma 1 is referred to the full version.

6 Security Analysis

An ownership transfer protocol is secure if it is against two types of attacks defined in Section 5.2. Without loss of generality, we analyse the security of proposed protocol on the k-th level. According to Lemma 1, we only show the security proof of the proposed protocol in Type II experiment.

Theorem 1. *The proposed ownership transfer protocol is* (r, s, n, ϵ)*-secure against the Type II attack if the CDH assumption is held.*

Due to the page limitation, the proof of Theorem 1 is referred to the full version.

7 Conclusion

In this paper, we defined a new secure model of ownership transfer protocols. It enhances the existing security models. We provided a definition of RFID ownership transfer and proposed a secure ownership transfer protocol. It achieves a single verification key to all the tags from an owner. The protocol satisfies all the security requirements. A formal proof of our proposed protocol was given.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 415–432. Springer, Heidelberg (2002)
2. Abyaneh, M.R.S.: On the privacy of two tag ownership transfer protocols for RFIDs. CoRR abs/1202.4663 (2012)
3. van Deursen, T., Mauw, S., Radomirović, S., Vullers, P.: Secure ownership and ownership transfer in RFID systems. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 637–654. Springer, Heidelberg (2009)
4. Elkhiyaoui, K., Blass, E.-O., Molva, R.: ROTIV: RFID ownership transfer with issuer verification. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 163–182. Springer, Heidelberg (2012)
5. Fernàndez-Mir, A., Trujillo-Rasua, R., Castellà-Roca, J., Domingo-Ferrer, J.: A scalable RFID authentication protocol supporting ownership transfer and controlled delegation. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 147–162. Springer, Heidelberg (2012)
6. Fouladgar, S., Afifi, H.: An effcient delegation and transfer of ownership protocol for RFID tags. In: First International EURASIP Workshop on RFID Technology. ACM (2007)
7. Kapoor, G., Zhou, W., Piramuthu, S.: Multi-tag and multi-owner RFID ownership transfer in supply chains. Decision Support Systems 52(1), 258–270 (2011)
8. Kulseng, L., Yu, Z., Wei, Y., Guan, Y.: Lightweight mutual authentication and ownership transfer for RFID systems. In: INFOCOM, pp. 251–255. IEEE (2010)
9. Molnar, D., Soppera, A., Wagner, D.: A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 276–290. Springer, Heidelberg (2006)
10. Ng, C.Y., Susilo, W., Mu, Y., Safavi-Naini, R.: Practical RFID ownership transfer scheme. Journal of Computer Security 19(2), 319–341 (2011)
11. Osaka, K., Takagi, T., Yamazaki, K., Takahashi, O.: An efficient and secure RFID security method with ownership transfer. In: Wang, Y., Cheung, Y., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 778–787. Springer, Heidelberg (2007)
12. Rizomiliotis, P., Rekleitis, E., Gritzalis, S.: Security analysis of the song-mitchell authentication protocol for low-cost RFID tags. Comm. Letters. 13(4), 274–276 (2009)
13. Saito, J., Imamoto, K., Sakurai, K.: Reassignment scheme of an RFID tag's key for owner transfer. In: Enokido, T., Yan, L., Xiao, B., Kim, D., Dai, Y., Yang, L.T. (eds.) EUC-WS 2005. LNCS, vol. 3823, pp. 1303–1312. Springer, Heidelberg (2005)
14. Song, B.: RFID tag ownership transfer. In: 4th Workshop on RFID Security - RFIDSec (2008)

Leakage Resilient Authenticated Key Exchange Secure in the Auxiliary Input Model*

Guomin Yang[1], Yi Mu[1], Willy Susilo[1], and Duncan S. Wong[2]

[1] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
{gyang,ymu,wsusilo}@uow.edu.au
[2] Department of Computer Science
City University of Hong Kong
duncan@cityu.edu.hk

Abstract. Authenticated key exchange (AKE) protocols allow two parties communicating over an insecure network to establish a common secret key. They are among the most widely used cryptographic protocols in practice. In order to resist key-leakage attacks, several leakage resilient AKE protocols have been proposed recently in the bounded leakage model. In this paper, we initiate the study on leakage resilient AKE in the auxiliary input model. A promising way to construct such a protocol is to use a digital signature scheme that is *entropically-unforgeable under chosen message and auxiliary input attacks*. However, to date we are not aware of any digital signature scheme that can satisfy this requirement. On the other hand, we show that in the *random oracle model*, it is sufficient to use a digital signature scheme that is secure under *random message and auxiliary input attacks* in order to build a secure AKE protocol in the auxiliary input model, while the existence of such a digital signature scheme has already been proven. We will also give a comparison between the existing public-key encryption based and digital signature based leakage resilient AKE protocols. We show that the latter can provide a higher level of security than the former.

Keywords: Leakage resilient cryptography, authenticated key exchange, auxiliary input model.

1 Introduction

Leakage Resilient Cryptography. Traditional cryptographic systems always assume that the user secret keys are absolutely secure and out of the adversary's reach. However, in recent years, various kinds of side-channel attacks [21,9,22,19] have shown that we can extract some partial information of the user secret keys stored in a computing device by observing the physical output of a computation (e.g. running time, power consumption, radiation, etc.). In order

* This work is supported by the ARC Future Fellowship (FT0991397).

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 204–217, 2013.

to defend against different types of side-channel (or more general, key leakage) attacks, *leakage resilient cryptography* have become a popular research topic in recent years.

Before constructing a leakage resilient cryptosystem, we must first build an appropriate security model to define the information an adversary can learn in a key leakage attack. There are several leakage models that have been defined in the literature. In the *relative leakage* model [2], the leakage function h can be any polynomial-time computable function with bounded output length. More specifically, let k denote the size of a user secret key sk, then size of $h(sk)$ must be significantly smaller than k (e.g. the size of $h(sk)$ is less than $k/2$). Later, in [26], the restriction on the size of $h(sk)$ is relaxed by requiring that the secret key sk should still have a sufficient amount of min-entropy left after the adversary has observed $h(sk)$.

Another leakage model that has been extensively studied in the literature is the *bounded retrieval* model (BRM) [13,17,3]. In BRM, the size of the leakage can be arbitrarily large, however, users can increase their secret key size flexibly so as to allow for a large amount of leakage. The main goal of this setting is to ensure that increasing the size of the user secret key should not result in significant increase in the computation or communication cost.

Recently, Dodis et al. [16,14] defined another leakage model named the *auxiliary input* model. In this model, an adversary is allowed to see a computationally hard-to-invert function (e.g. a one-way permutation) of the secret key. In other words, the auxiliary input model has eliminated the leakage bound, and therefore can capture a larger class of leakage functions.

Authenticated Key Exchange. Authenticated Key Exchange (AKE) protocols are mechanisms that allow two parties communicating over an insecure network to establish a common secret key. They are a central piece for building secure communication channels. The design and analysis of AKE protocols have been extensively studied in the last three decades for different network settings (e.g. [6,7,5,11,1,24,27,10,29,28]). The first formal security model for AKE was proposed by Bellare and Rogaway [6]. The Bellare-Rogaway (or BR, for short) model and its variants are nowadays the *de facto* standard for analyzing the security of an AKE protocol. In particular, the Canetti-Krawczyk (CK) model [11], which can be considered as a combination of the BR model and the Bellare-Canetti-Krawczyk (BCK) model [4], has been used to prove the security of many practical AKE protocols such as the ISO protocol [20] (named SIG-DH in [11]) and the Internet Key Exchange (or SIGMA) protocol [12,23]. In FC'11, Yang et al. [28] extended the CK model to consider AKE under bad randomness. Two new models were proposed in [28], one formalized the reset attacks, and the other one formalized the bad randomness attacks. Some generic methods for enhancing the security of existing AKE protocols (such as ISO and SIGMA) were also proposed.

Leakage Resilient AKE. Several leakage resilient AKE protocols have been proposed recently. In [3], Alwen et al. extended the CK model to the bounded retrieval setting, and showed that in BRM, a leakage resilient AKE protocol can

be constructed from an entropically-unforgeable digital signature scheme secure under chosen-message attacks. Later, in [15], Dodis et al. showed that we can also construct leakage resilient AKE protocols based on public-key encryption (PKE) schemes secure in the bounded leakage model. In Sec. 4, we will review these two constructions and give a comparison between PKE-based and signature-based leakage resilient AKE protocols. In ASIACCS'11, based on the eCK security model proposed by LaMacchia, Lauter, and Mityagin [24], Moriyama and Okamoto [25] presented a new bounded leakage model for AKE protocols. They also proposed a two-pass implicitly-authenticated leakage resilient AKE protocol and proved its security in their security model.

Our Contributions. In this paper, we initiate the study on leakage resilient AKE in the auxiliary input model. Based on the result in [3], we can expect that such a protocol can be built by using a digital signature scheme that is *entropically-unforgeable under chosen message and auxiliary input attacks*. However, a problem arises when using this approach: to date we are not aware of any digital signature scheme that can satisfy the requirement. Although Faust et al. [18] have recently proposed a signature scheme that is secure under chosen-message and auxiliary input attacks, they assume the adversary can only see an *exponentially hard-to-invert function*, rather than a *computationally hard-to-invert function*, of the user secret key.

In this paper, we show that in the *random oracle model* [8], it is sufficient to use a digital signature scheme that is secure under *Random Message and Auxiliary Input Attacks* in order to build a secure AKE protocol in the auxiliary input model, while the existence of such a digital signature scheme has recently been proved in [18]. The key to ensure that this condition is sufficient comes from the specific design requirement for AKE protocols: an AKE participant not only receives but also generates random challenges in each AKE session. We will elaborate on this in Sec. 5.

It is worth noting that we may also use public-key encryption schemes secure in the auxiliary input model to achieve our goal. However, as we will show in Sec. 4, in the CK-model, Signature-based AKE protocols will offer better security than PKE-based protocols when the adversary can reveal the session state of a party during a protocol execution.

2 Preliminaries

Notations. We only consider probabilistic polynomial time (PPT) algorithms in this paper. In general, all the PPT algorithms have a security parameter 1^k as input, however, this input is usually omitted. We use $x \leftarrow S$ to denote the operation of randomly selecting x from a set S, and $y \leftarrow A(x)$ to indicate that y is the output of running an algorithm A on input x.

Leakage Functions. We follow the work of Dodis et al. [14] to define the class of admissible leakage functions $\mathcal{H}$ with regard to a public-key cryptosystem. We define $\mathcal{H}_{\mathsf{pkow}}(\ell(k))$ as the class of polynomial time computable functions

$h : \{0,1\}^{|pk|+|sk|} \rightarrow \{0,1\}^*$ such that given $(pk, h(pk, sk))$, no PPT adversary can find sk with probability greater than $\ell(k) \geq 2^k$, where (pk, sk) denote a random key pair generated by running the key generation algorithm of the public-key cryptosystem. In the rest of the paper, we will simply use $\mathcal{H}_{\mathsf{pkow}}$ to denote $\mathcal{H}_{\mathsf{pkow}}(\mathsf{negl}(k))$ where $\mathsf{negl}(\cdot)$ can be any negligible function.

Digital Signature. A digital signature scheme $\mathcal{DS}$ consists of three polynomial time algorithms.

- $\mathcal{DS}.\mathsf{SKG}(1^k)$: the key generation algorithm takes a security parameter 1^k as input and outputs a private signing key sk and a public verification key vk.
- $\mathcal{DS}.\mathsf{Sig}(sk, m)$): the signing algorithm takes a signing key sk and a message m from the message space $\mathcal{M}$ as input and outputs a signature σ.
- $\mathcal{DS}.\mathsf{Ver}(vk, m, \sigma)$): the verification algorithm takes a verification key vk, a message m, and a signature σ as input and outputs a bit '1' or '0'.

Correctness. For any $k \in \mathbb{N}$, $(vk, sk) \leftarrow \mathcal{DS}.\mathsf{SKG}(1^k)$, and $m \in \mathcal{M}$, we have

$$1 \leftarrow \mathcal{DS}.\mathsf{Ver}(vk, m, \mathcal{DS}.\mathsf{Sig}(sk, m)).$$

Unforgeability under Random Message and Auxiliary Input Attacks [18]. We say $\mathcal{DS}$ satisfies Random Message Unforgeability under Random Message and Auxiliary Input Attacks (RU-RMAA) with respect to a class of admissible leakage functions $\mathcal{H}$ if for any polynomial time algorithm $\mathcal{F}$, and any function $h \in \mathcal{H}$,

$$\mathbf{Adv}^{\mathsf{RU\text{-}RMAA}}_{\mathcal{DS},\mathcal{H},\mathcal{F}}(k) = \Pr\left[\begin{array}{c}(vk, sk) \leftarrow \mathcal{DS}.\mathsf{SKG}(1^k); m^* \leftarrow \mathcal{M};\\ \sigma^* \leftarrow \mathcal{F}^{\mathcal{O}(sk,\cdot)}(vk, h(vk, sk), m^*) :\\ \mathcal{DS}.\mathsf{Ver}(vk, m^*, \sigma^*) = 1\end{array}\right]$$

is negligible in k, where the oracles $\mathcal{O}(sk, \cdot)$ is defined as

$$\text{Oracle } \mathcal{O}(sk, \cdot): \quad m \leftarrow \mathcal{M}; \quad \text{return } (m, \mathcal{DS}.\mathsf{Sig}(sk, m)).$$

Decisional Diffie-Hellman (DDH) Assumption: Let g denote a generator of a cyclic group $\mathbb{G}$ with prime order q. The DDH assumption says for any polynomial time algorithm $\mathcal{D}$,

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{D}}(k) = \Pr[\mathcal{D}(g, g^a, g^b, Z) = 1 | Z = g^{ab}] - \Pr[\mathcal{D}(g, g^a, g^b, Z) = 1 | Z = g^r]$$

is negligible in k where a, b, r are randomly selected from $\mathbb{Z}_q$.

3 Security Model and Definition

3.1 System Model

An Authenticated Key Exchange (AKE) protocol consists of two probabilistic polynomial time algorithms: the Long-Lived Key generation algorithm SKG and a protocol execution algorithm P. In this paper, we focus on the public key

setting where the algorithm SKG returns a public key and the corresponding private key upon each invocation.

Protocol Participants. Let $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$ denote the set of users. Each user $U \in \mathcal{U}$ holds a public/private key pair (pk_U, sk_U) that is generated by honestly executing the Long-Lived Key generation algorithm SKG. A user may run many instances concurrently. We denote instance i of user U by Π_U^i.

Protocol Execution. A protocol execution algorithm P determines how an instance behaves in response to messages from the environment. Upon receiving an incoming message M_{in}, an instance executes the protocol P and generates

$$(M_{\text{out}}, \mathsf{dec}, \mathsf{sid}_U^i, \mathsf{pid}_U^i, \text{ssk}, St_U^i) \leftarrow \text{P}(U, pk_U, sk_U, St_U^i, M_{\text{in}}).$$

The first component M_{out} corresponds to the responding message, and the second component dec denotes the *decision* of the instance. A session id sid_U^i and partner id pid_U^i may be generated during the protocol execution. When the decision is acc, the instance holds a session key ssk which is to be used by upper layer applications. The instance may also update its internal state St_U^i.

Partnership. The partnership between two instances is defined via parter ID (pid) and session ID (sid). The pid names the party with which the instance believes it has just exchanged a key, and the sid is an identifier which uniquely labels the AKE session. We say two instances Π_U^i and Π_V^j are partners if $\mathsf{pid}_U^i = V, \mathsf{pid}_V^j = U$ and $\mathsf{sid}_U^i = \mathsf{sid}_V^j$.

3.2 Security Model

We consider an adversary $\mathcal{A}$ with full control over the routing and scheduling of network messages. Our adversarial model is defined via a game between the adversary $\mathcal{A}$ and a game simulator $\mathcal{SIM}$. $\mathcal{SIM}$ first tosses a random coin b which will be used later in the game. $\mathcal{SIM}$ then generates for each $U \in \mathcal{U}$ a public/secret key pair (pk_U, sk_U) and gives pk_U and auxiliary input $h_U(pk_U, sk_U)$ to $\mathcal{A}$ where $h_U \in \mathcal{H}_{\mathsf{pkow}}$. $\mathcal{A}$ is allowed to make the following oracle queries to the simulator:

- Send(U, i, m): This query allows the adversary to send a message m to an instance Π_U^i. If the message m is sent by another instance $\Pi_{U'}^j$ with the intended receiver U, then this query models a passive attack. Otherwise, it models an active attack by the adversary. The simulator then simulates the reaction of Π_U^i upon receiving the message m by running $\text{P}(U, pk_U, sk_U, St_U^i, m)$, and returns to $\mathcal{A}$ the response (if there is any) that Π_U^i would generate.
- Corrupt(U): This query allows the adversary to corrupt a party U. By making this query, the adversary learns the long-term secret key sk_U of user U.
- StateReveal(U, i): This query allows the adversary to learn the current state information St_U^i held by the instance Π_U^i.

- $\textsc{Reveal}(U, i)$: This query allows the adversary to learn the session key that has been generated by the instance Π_U^i. If the instance Π_U^i does not hold any session key, then a special symbol $\perp$ is returned to the adversary.
- $\textsc{Test}(U^*, i^*)$: This query can only be made to a *fresh* instance $\Pi_{U^*}^{i^*}$ (as defined below). If the instance $\Pi_{U^*}^{i^*}$ holds a session key $\text{ssk}_{U^*}^{i^*}$, then $\mathcal{SIM}$ does the following
  - if the coin $b = 1$, $\mathcal{SIM}$ returns $\text{ssk}_{U^*}^{i^*}$ to the adversary;
  - otherwise, a random session key is drawn from the session key space and returned to the adversary.

  Otherwise, a special symbol $\perp$ is returned to the adversary.

SK-security without PFS. We define session key security without perfect forward secrecy as follows.

We say an instance Π_U^i is *fresh* if

- $\mathcal{A}$ has never made a $\textsc{Corrupt}$ query to U or pid_U^i; and
- $\mathcal{A}$ has never made a $\textsc{Reveal}$ query to Π_U^i or its partner; and
- $\mathcal{A}$ has never made a $\textsc{StateReveal}$ query to Π_U^i or its partner.

At the end of the game, the adversary outputs a bit b' as her guess for b. The adversary's advantage in winning the game is defined as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{AKE}}(k) = |2\Pr[b' = b] - 1|.$$

Definition 1. *We say an AKE protocol is SK-Secure without perfect forward secrecy in the auxiliary input model if the following conditions hold.*

1. *If two uncorrupted parties complete matching sessions then they both output the same key.*
2. *For any PPT adversary A, and any* $\{h_U \in \mathcal{H}_{\mathsf{pkow}}\}_{U \in \mathcal{U}}$, $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{AKE}}(k)$ *is a negligible function of* k.

SK-security with PFS. In order to define perfect forward secrecy, we follow the approach of Canetti and Krawczyk [11] by introducing a new type of oracle query

- $\textsc{Expire}(U, i)$: Upon receiving this query, the simulator erases all the state information St_U^i and the session key ssk_U^i held by the instance Π_U^i.

The *freshness* of an instance Π_U^i is now redefined as follows:

- $\mathcal{A}$ makes a $\textsc{Corrupt}(U)$ query only after an $\textsc{Expire}(U, i)$ query; and
- $\mathcal{A}$ has never made a $\textsc{Reveal}$ query to Π_U^i; and
- $\mathcal{A}$ has never made a $\textsc{StateReveal}$ query to Π_U^i; and
- if Π_U^i has a partner instance Π_V^j, then $\mathcal{A}$ also obeys the above rules with respect to Π_V^j; otherwise, $\mathcal{A}$ has never made a $\textsc{Corrupt}(\mathsf{pid}_U^i)$ query.

Defined the adversary's advantage in winning the PFS game as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{AKE\text{-}PFS}}(k) = |2\Pr[b' = b] - 1|.$$

Definition 2. *We say an AKE protocol is* SK-Secure *with perfect forward secrecy in the auxiliary input model if the following conditions hold.*

1. *If two uncorrupted parties complete matching sessions then they both output the same key.*
2. *For any PPT adversary A, and any $\{h_U \in \mathcal{H}_{\mathsf{pkow}}\}_{U \in \mathcal{U}}$, $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{AKE\text{-}PFS}}(k)$ is negligible in k.*

4 SIG-DH vs PKE-DH

Several leakage resilient AKE protocols [3,15,25] have been proposed recently in the bounded leakage/retrieval model. In this section, we briefly review the Signature-based Diffie-Hellman protocol (eSIG-DH) [3] and the PKE-based Diffie-Hellman protocol (Enc-DH) [15], and give a comparison between them.

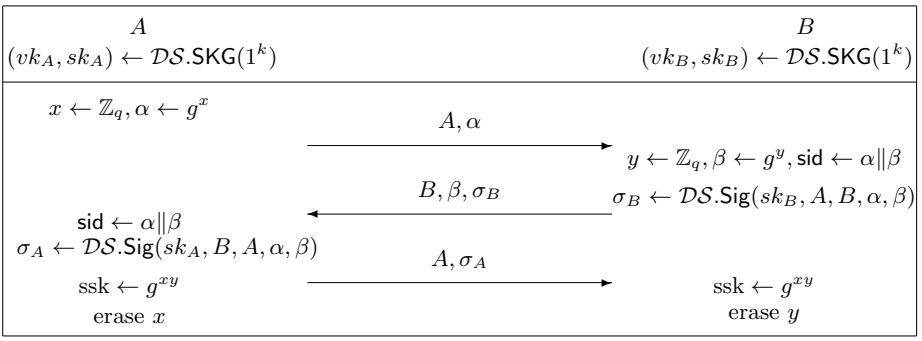


Fig. 1. The eSIG-DH Protocol [3]

The eSIG-DH protocol [3] is presented in Fig. 1. It is an extension of the SIG-DH protocol [11] in the bounded retrieval setting. The protocol makes use of a digital signature scheme $\mathcal{DS}$ that is entropically-unforgeable under chosen message attacks in the bounded retrieval model to achieve mutual authentication. In contrast, the Enc-DH protocol [15] (Fig. 2) is based on a leakage resilient PKE scheme supporting labels. The idea behind Enc-DH is that only the real user who has the decryption key can decrypt a ciphertext and answer the challenge.

Deniable Authentication. As pointed out by Dodis et al. in [15], due to the non-repudiation property of the digital signatures, it is obvious that the eSIG-DH protocol cannot provide the feature of deniable authentication. On the other hand, the Enc-DH protocol can achieve such a property, since the messages generated by user A in fact can be simulated by user B, and vice versa.

Security under StateReveal *Query.* There is actually another big difference between the eSIG-DH and Enc-DH protocols: if we consider the full CK model where the adversary is able to make StateReveal queries, then an adversary can launch the following attack against the Enc-DH protocol:

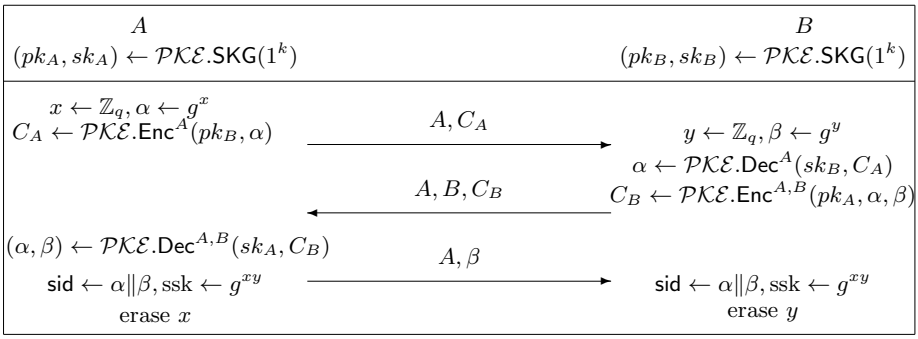


Fig. 2. The Enc-DH Protocol [15]

1. The adversary activates an instance of A to start a new AKE session with B, and faithfully delivers the first message (A, C_A) to B.
2. Upon receiving the response (A, B, C_B) from B, the adversary makes a StateReveal query to B and obtains α.
3. The adversary then generates $\beta' = g^{y'}$ and $C'_B \leftarrow \mathcal{PKE}.\mathsf{Enc}^{A,B}(pk_A, \alpha, \beta')$, and sends (A, B, C'_B) to user A.
4. User A would accept the session, send the third message (A, β'), and output the session key $\mathrm{ssk}_A = g^{xy'}$.

Since the adversary knows the value of y', she can derive the session key and win the game. It is worth noting that in the above attack, the instance of user A does not have a partner, but it is still fresh according to the definition. In other words, the adversary can successfully break the authentication mechanism employed under the Enc-DH protocol if she can make StateReveal queries. On the other hand, it is easy to check that such a problem does not exist in the eSIG-DH protocol.

5 A Leakage Resilient AKE Protocol Secure in the Auxiliary Input Model

In this section, we present a leakage resilient AKE protocol that is secure in the auxiliary input model. The scheme is based on a signature scheme that is *random message unforgeable under random message and auxiliary input attacks* (RU-RMAA) [18].

5.1 The aSIG-DH AKE Protocol

The only difference between our new protocol and the eSIG-DH protocol (Fig. 1) resides in the computation of the digital signatures. In the eSIG-DH protocol [3], an entropically-unforgeable signature scheme secure in the bounded-retrieval

model is used, while in the aSIG-DH protocol, each party will first compute a hash digest of the message, and then sign the hash digest using an RU-RMAA secure digital signature scheme [18]. Another important remark we should make is that same as the SIG-DH [11] and eSIG-DH [3] protocols, we assume that the signing operation is an atomic operation done by an independent module.

Why RU-RMAA Security is Sufficient. Readers may wonder that given an RU-RMAA secure digital signature scheme $\mathcal{DS} = (\mathsf{SKG}, \mathsf{Sig}, \mathsf{Ver})$, if we first hash the message M and then sign the hash digest $H(M)$ using $\mathcal{DS}.\mathsf{Sig}$, will we obtain an entropically-unforgeable signature scheme secure under chosen message attacks in the random oracle model (i.e. H is modelled as a random oracle)? Unfortunately, in general the answer is No! Consider that $\mathcal{DS}.\mathsf{Sig}$ is a randomized algorithm, then in a chosen message attack, when the adversary makes two signing queries with the same message m, two distinct yet valid signatures should be returned to the adversary. However, such signing queries cannot be answered by using the signing oracle defined in the RU-RMAA security game. Fortunately, in an AKE protocol, each participant will generate a fresh challenge in an AKE session. To impersonate a participant A without knowing A's signing key, the adversary needs to forge a valid signature on the fresh challenge sent by B. On the other hand, to answer the SEND queries made by the adversary to the participant A, the simulator can make use of the signing oracle in the RU-RMAA security game since the signed message will also contain a fresh challenge generated by A (i.e. a message to be signed by A would not appear in two different sessions).

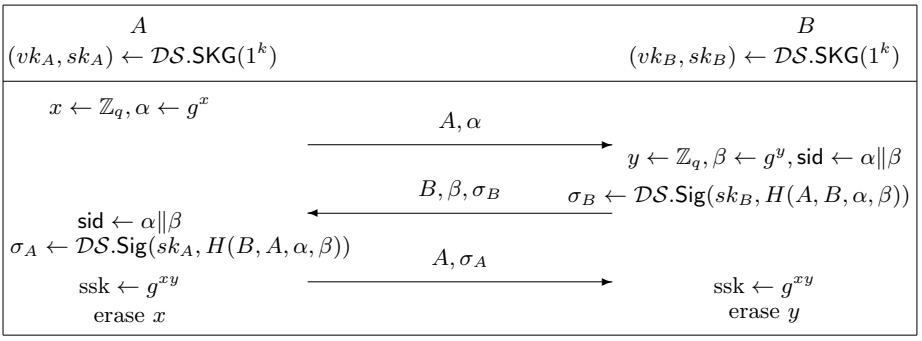


Fig. 3. The aSIG-DH Protocol

Theorem 1. *The aSIG-DH protocol is SK-secure with perfect forward secrecy in the auxiliary input model if the digital signature scheme $\mathcal{DS}$ is RU-RMAA secure w.r.t. $\mathcal{H}_{\mathsf{pkow}}$, the DDH assumption holds in group $\mathbb{G}$, and H is a random oracle.*

Proof. The first condition in the definition of SK-security is easy to see. Below we prove that the aSIG-DH protocol also satisfies the second condition. We define a sequence of games $G_i (i \geq 0)$ where G_0 is the original game defined in our security model with PFS. We also define $\mathbf{Adv}_i$ as the advantage of the adversary in game G_i (i.e. $\mathbf{Adv}_0 = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{AKE\text{-}PFS}}(k)$).

Game G_1. Let forge denote the event that $\mathcal{A}$ successfully forges a valid signature of a user U (i.e. an instance Π_V^j receives a message/signature pair $((V, U, \alpha, \beta), \sigma_U)$ in a SEND query such that $\mathcal{DS}.\mathsf{Ver}(pk_U, (V, U, \alpha, \beta), \sigma_U) = 1$ and there is no instance of U which has sent a valid signature on (V, U, α, β) to the $\mathcal{A}$) before corrupting U. If a forge event happens, then the simulator aborts the game and outputs a random bit b'. Then we have

$$\Pr[b' = b \text{ in } G_0 | \neg \mathsf{forge}] = \Pr[b' = b \text{ in } G_1 | \neg \mathsf{forge}]$$

and

$$\Pr[b' = b \text{ in } G_0] - \Pr[b' = b \text{ in } G_1] \leq \Pr[\mathsf{forge}].$$

Therefore, we have

$$\mathbf{Adv}_0 \leq \mathbf{Adv}_1 + 2\Pr[\mathsf{forge}].$$

In the following, we show that the event forge happens only with a negligible probability.

CLAIM. The event forge happens only with a negligible probability if $\mathcal{DS}$ is RU-RMAA secure with respect to $\mathcal{H}_{\mathsf{pkow}}$ and H is a random oracle.

Proof. Suppose there exists an adversary $\mathcal{A}$ and a set of leakage functions $\mathbb{S} = \{h_1, h_2, \cdots, h_n\} \subset \mathcal{H}_{\mathsf{pkow}}$ w.r.t. the set of users $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$ such that a forge event would occur with a non-negligible probability, we show that there exists another algorithm $\mathcal{F}$ and a leakage function $h \in \mathcal{H}_{\mathsf{pkow}}$ such that $\mathcal{F}$ can win the RU-RMAA security game also with a non-negligible probability.

Let h^* denote a leakage function randomly selected from $\mathbb{S}$, and $\mathcal{F}$ is given a challenge $(vk^*, h^*(vk^*, sk^*), m^*)$ as input where $(vk^*, sk^*) \leftarrow \mathcal{DS}.\mathsf{SKG}(1^k)$ and m^* is randomly selected from the message space $\mathcal{M}$ of $\mathcal{DS}$. Wlog, assume h^* is the i-th function in the set $\mathbb{S}$ (i.e. $h^* = h_i$). $\mathcal{F}$ then sets the challenge public key vk^* as the public key of the user U_i. $\mathcal{F}$ then generates the long-term keys for all the remaining users in $\mathcal{U}$ by running $\mathcal{DS}.\mathsf{SKG}(1^k)$. In addition, $\mathcal{F}$ randomly selects $\zeta \leftarrow [1, q_H]$ where q_H denotes the number of hash queries $\mathcal{A}$ would make in the game. $\mathcal{F}$ then passes $\{vk_j, h_j(vk_j, sk_j)\} (1 \leq j \leq n)$ to $\mathcal{A}$ and answers $\mathcal{A}$'s oracle queries as follows.

$\mathcal{F}$ answers $\mathcal{A}$'s hash oracle queries as follows: when $\mathcal{A}$ submits a hash query, $\mathcal{F}$ first checks if the same input has been queried before. If yes, then the same output is returned to $\mathcal{A}$. Otherwise, $\mathcal{F}$ checks if the input has the format $(\cdot, U_i, \cdots)$. If not, a random element in $\mathcal{M}$ is selected and returned to $\mathcal{A}$; otherwise, $\mathcal{F}$ issues a signing query to its signing oracle to obtain (m, σ), and sets m as the hash value of $(\cdot, U_i, \cdots)$. When $\mathcal{A}$ makes the ζ-th hash query, $\mathcal{F}$ sets m^* as the hash value and returns m^* to $\mathcal{A}$.

When $\mathcal{A}$ makes a SEND query to an instance of U_i, if a signature of U_i on the hash value of $(U_j, U_i, \alpha, \beta)$ is required in order to answer this query, $\mathcal{F}$ first

checks if a signature σ corresponding to $H(U_j, U_i, \alpha, \beta)$ has been obtained from its signing oracle before. If yes, σ is returned to $\mathcal{A}$. Otherwise, $\mathcal{F}$ first makes a signing query to obtain (m, σ), then sets m as the hash value of $(U_j, U_i, \alpha, \beta)$ and uses σ as the corresponding signature to answer the SEND query. Since each instance of U_i will generate a fresh Diffie-Hellman component (either α or β), with overwhelming probability, $(U_j, U_i, \alpha, \beta)$ would never repeat in difference instances of U_i.

$\mathcal{F}$ simulates other operations performed by each instance honestly, and answers all the REVEAL and STATEREVEAL queries as usual. If $\mathcal{A}$ makes a CORRUPT (U_i) query during the game, $\mathcal{F}$ aborts the game and outputs nothing.

If $\mathcal{A}$ successfully forges a signature σ^* of U_i on the hash value m^*, then $\mathcal{F}$ outputs σ^* and halts. Otherwise, $\mathcal{F}$ outputs nothing and halts when $\mathcal{A}$ halts. Since U_i and ζ are randomly selected, it is clear that

$$\mathbf{Adv}^{\mathsf{RU\text{-}RMAA}}_{\mathcal{DS},\mathcal{H}_{\mathsf{pkow}},\mathcal{F}}(k) = \frac{1}{n \cdot q_H}\Pr[\mathsf{forge}].$$

Hence, we have

$$\mathbf{Adv}_0 \leq \mathbf{Adv}_1 + 2nq_H\mathbf{Adv}^{\mathsf{RU\text{-}RMAA}}_{\mathcal{DS},\mathcal{H}_{\mathsf{pkow}},\mathcal{F}}(k).$$

Game G_2. In game G_2, we change game G_1 as follows: the simulator randomly chooses an instance (say the i-th instance) $\Pi^{i^*}_{U^*}$ among all the instances created in the game, if the TEST query is not performed on $\Pi^{i^*}_{U^*}$, the simulator aborts and outputs a random bit b'. Let n_I denote the number of instances created in the game, then we have

$$\begin{aligned}\Pr[b' = b] &= \Pr[b' = b|\textsc{Test}(U^*, i^*)]\Pr[\textsc{Test}(U^*, i^*)] \\ &\quad +\Pr[b' = b|\neg\textsc{Test}(U^*, i^*)]\Pr[\neg\textsc{Test}(U^*, i^*)] \\ &= \Pr[b' = b \text{ in } G_1]\frac{1}{n_I} + \frac{1}{2}(1 - \frac{1}{n_I}) \\ &= \frac{1}{2} + \frac{1}{n_I}(\Pr[b' = b \text{ in } G_1] - \frac{1}{2})\end{aligned}$$

and

$$\mathbf{Adv}_1 = n_I\mathbf{Adv}_2.$$

Game G_3. In game G_3, we change game G_2 by replacing the Diffie-Hellman key $g^{x^*y^*}$ in the test session with a random element $g^r \in \mathbb{G}$. Below we show that if the adversary's advantage changes significantly in game G_3 , we can construct a distinguisher $\mathcal{B}$ to break the Decisional Diffie-Hellman (DDH) assumption.

$\mathcal{B}$ is given a challenge (g^a, g^b, Z), in which with equal probability, Z is either g^{ab} or a random element of $\mathbb{G}$. $\mathcal{B}$ simulates game G_2 honestly by generating all the long-term secret keys for all the users. When simulating the i-th instance $\Pi^{i^*}_{U^*}$ and its partner, $\mathcal{A}$ sets $g^{x^*} = g^a, g^{y^*} = g^b$ and Z as the corresponding session key. Finally, if $\mathcal{A}$ wins the game, $\mathcal{B}$ outputs 1, otherwise, $\mathcal{B}$ outputs 0.

Since a forge event would not happen on $\mathsf{pid}_{U^*}^{i^*}$ before $\mathsf{pid}_{U^*}^{i^*}$ is corrupted, we can guarantee that a partner instance of $\Pi_{U^*}^{i^*}$ must exist. So the Diffie-Hellman components in the test session must be g^a and g^b. If $Z = g^{ab}$, then $\mathcal{A}$ is in game G_2; otherwise, if Z is a random element of $\mathbb{G}$, then $\mathcal{A}$ is in game G_3. Therefore we have

$$\begin{aligned}\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}(k) &= \Pr[\mathcal{B} \text{ outputs } 1 | Z = g^{ab}] - \Pr[\mathcal{B} \text{ outputs } 1 | Z = g^r] \\ &= \Pr[\mathcal{A} \text{ wins the game} | Z = g^{ab}] - \Pr[\mathcal{A} \text{ wins the game} | Z = g^r] \\ &= \frac{1}{2}(\mathbf{Adv}_2 - \mathbf{Adv}_3)\end{aligned}$$

and

$$\mathbf{Adv}_2 \leq \mathbf{Adv}_3 + 2\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}(k).$$

It is clear that the adversary $\mathcal{A}$ has no advantage than random guess in game G_3 (i.e. $\mathbf{Adv}_3 = 0$). Hence, we have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{AKE\text{-}PFS}}(k) \leq 2n_I\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}(k) + 2nq_H\mathbf{Adv}_{\mathcal{DS},\mathcal{H}_{\mathsf{pkow}},\mathcal{F}}^{\mathsf{RU\text{-}RMAA}}(k).$$

□

6 Conclusion

In this paper, we initiated the study on leakage resilient authenticated key exchange in the auxiliary input model. We showed that in the random oracle model, we can build an AKE protocol secure under auxiliary input attacks based on a digital signature scheme that is random message unforgeable under random message and auxiliary input attacks (RU-RMAA). We also showed the differences between signature-based and public-key encryption-based Diffie-Hellman protocols and concluded that signatured-based protocols can offer a higher level of security than encryption-based ones when the adversary is allowed to learn the state information of a protocol participant. We leave the construction of a secure AKE against auxiliary input attacks without random oracles as our future work.

References

1. Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A.D., Reingold, O.: Just fast keying: Key agreement in a hostile Internet. ACM Trans. Inf. Syst. Secur. 7(2), 242–273 (2004)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)

4. Bellare, M., Canetti, R., Krawczyk, H.: Modular approach to the design and analysis of key exchange protocols. In: ACM STOC 1998, pp. 419–428 (1998)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
7. Bellare, M., Rogaway, P.: Provably secure session key distribution — the three party case. In: ACM STOC 1995, pp. 57–66 (1995)
8. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73 (1993)
9. 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)
10. Boyd, C., Cliff, Y., Gonzalez Nieto, J.M., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 69–83. Springer, Heidelberg (2008), http://eprint.iacr.org/2008/007
11. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001), http://eprint.iacr.org/2001/040/
12. Canetti, R., Krawczyk, H.: Security analysis of IKE's signature-based key-exchange protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002), http://eprint.iacr.org/2002/120/
13. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)
14. Dodis, Y., Goldwasser, S., Tauman Kalai, Y., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (2010)
15. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010)
16. Dodis, Y., Tauman Kalai, Y., Lovett, S.: On cryptography with auxiliary input. In: ACM STOC 2009, pp. 621–630 (2009)
17. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
18. Faust, S., Hazay, C., Nielsen, J.B., Nordholt, P.S., Zottarel, A.: Signature schemes secure against hard-to-invert leakage. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 98–115. Springer, Heidelberg (2012)
19. 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)
20. Entity authentication mechanisms - Part 3: Entity authentication using asymmetric techniques. ISO/IEC IS 9798-3 (1993)
21. 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)

22. 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)
23. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003)
24. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
25. Moriyama, D., Okamoto, T.: Leakage resilient eCK-secure key exchange protocol without random oracles. In: ACM ASIACCS 2011, pp. 441–447 (2011)
26. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
27. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474–484. Springer, Heidelberg (2007), Full paper available at `http://eprint.iacr.org/2007/473`
28. Yang, G., Duan, S., Wong, D.S., Tan, C.H., Wang, H.: Authenticated key exchange under bad randomness. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 113–126. Springer, Heidelberg (2012)
29. Yang, G., Wong, D.S., Wang, H., Deng, X.: Two-factor mutual authentication based on smart cards and passwords. J. Comput. Syst. Sci. 74(7), 1160–1172 (2008)

Simplified PACE|AA Protocol*

Lucjan Hanzlik**, Łukasz Krzywiecki, and Mirosław Kutyłowski

Faculty of Fundamental Problems of Technology, Wrocław University of Technology
{firstname.secondname}@pwr.wroc.pl

Abstract. We present SPACE|AA protocol that merges Chip Authentication of a smart card with card owner authorization via PACE protocol implemented in German personal identity documents. It is an improvement of PACE|AA protocol presented at Financial Cryptography 2012. Moreover, we explicitly formulate privacy model implicitely used by the authors of PACE|AA.

Keywords: personal ID document, MRTD, chip authentication, active authentication, PACE, Diffie-Hellman protocol, simulatability, privacy protection.

1 Introduction

We consider authentication protocols for personal identity documents. We focus on solutions that are efficient enough from the practical point of view and secure in real world scenarios. Due to severely limited resources of a chip of an identity document and scale of deployment even slight improvements may have a significant practical impact.

The first important step in this area was introduction of biometric passports – a substantial improvement against forgeries is due to cryptographic protection mechanisms. More features are already implemented in personal identity cards in some countries. In particular, they can be used as travel documents within European Union. For this reason we shall use name Machine Readable Travel Documents or MRTD for short.

One of frequent design decisions for MRTD is to use contactless communication interface. This increases durability of a chip as it becomes completely sealed, but at the same time the document holder looses direct control over chip activation. Since the chip works in a slave mode responding to a wireless reader, the document holder might be unaware of interaction between the MRTD chip and the reader. For this reason, many citizens are reluctant to activation of electronic layers of MRTD. In order to prevent such an unwelcome interaction, additional protection is necessary.

The primary goal of cryptography in MRTD is to prevent forgery of identity documents. The simplest solution is to store personal data in chip's memory together with the signature of a document issuer. However, the signed data may be stored on a different chip when cloning a MRTD. Another problem is that signed data can be presented to third parties and the signatures serve as an undeniable proof of data authenticity.

Design of MRTD has to take into account many factors, frequently neglected in the literature: space limitations (size of keys, program code and working space), communication volume and number of rounds (important due to low speed and possible

* Supported by Foundation for Polish Science, Ventures project 2012-9/4 and MISTRZ.
** Corresponding author.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 218–232, 2013.

faults), strict bound on computation time (necessity to finalize interaction in at most few seconds), use of standard components (new components mean compatibility problems, high certification costs, etc.), patent freeness (legal disputes concerning intellectual property rights and patent costs may freeze a project).

1.1 Security Requirements

No Operation without Consent of the Holder. The protocol should guarantee that the MRTD chip cannot be used unless its holder explicitly agrees and that the reader communicates only with a reader approved by the card holder. This can be achieved by password based mechanisms (such as PACE protocol discussed later). First, a password π is either entered by the document holder to the reader or scanned by the reader from the surface of the MRTD chip. Then an interaction between the reader and the MRTD chip is started; during this interaction the chip must become convinced that the reader is using the same password π as stored inside the chip. There are the following requirements for password protection:

- probability that a reader not knowing the proper password π succeeds to convince the MRTD chip that it knows π must be negligible,
- an eavesdropper cannot derive π. So not only the password must not be transmitted in clear, but also that a brute force attack based on checking each password for consistency with the protocol transcript should be ineffective.
- a reader may attempt to execute the protocol with an MRTD chip by guessing the password π (many trials may occur undetected due to wireless communication). As entropy of passwords is limited, the advantage of this attack is non-negligible. So we have to guarantee only that an adversary has no substantial advantage over checking the passwords one by one.

Active Authentication of the MRTD Chip. The reader must become convinced via execution of a challenge-response algorithm that it communicates with a genuine MRTD chip holding a secret key x stored in the MRTD chip. We require impersonation resilience: – probability that a chip not knowing x gets accepted by the reader must be negligible. Note that an adversary may have access to other parameters (like ephemeral keys derived in genuine sessions between the MRTD chip holding x and some readers). Moreover, we demand key secrecy: an adversary cannot reconstruct x based on interaction between an MRTD chip holding x and a reader. In particular, the adversary can control the reader and execute the protocol in a malicious way and may know the password π of the MRTD.

Data Confidentiality. The protocol should guarantee that the data transmitted between the MRTD chip and the reader are not accessible to an eavesdropper. For this purpose at the initial phase the reader and the MRTD chip establish a shared key (*Key Exchange Protocol* or KE for short), the rest of the interaction is encrypted with the session key.

The protocol should guarantee key secrecy: an adversary cannot learn a session key established by an reader and an MRTD chip, even if he knows the session keys established in different sessions and learns the long time secret keys of the MRTD chip.

It must also protect against impersonation: after establishing a shared session key the reader must be sure that the session key is shared with the MRTD chip that was checked via active authentication, while the chip must be sure that the reader is the one, which was checked via password knowledge. In particular, an adversary that knows some of the secrets (password or the active authentication secret key) cannot take over the session and establish a session key on behalf of one of the protocol parties.

Privacy Protection. The protocol should guarantee that apart from a legitimate party nobody may learn which MRTD chip is present at a given place. This concerns resilience to tracing: an eavesdropper should be unable to check that a certain MRTD participates in an eavesdropped session. Unlike many KE protocols, the identity of the MRTD chip should be unknown to the reader till the password authentication terminates successfully. This addresses the problem of readers that engage MRTD chips in protocol runs in order to learn the identities for the sake of tracing the users. In particular, the runs might be incomplete.

Moreover, the protocol should prevent transfer of authentication proof: a transcript of communication together with all information that the reader can reveal, should have no proof value for third parties. Violating this property would cause severe personal data protection problems. This problem does not arise if a protocol is *simulatable*, i.e. everybody can generate fake transcripts that are indistinguishable from the transcripts from real interactions.

1.2 Related Work and Paper Contribution

SPEKE. Password authenticated key exchange (PAKE) allows two parties to establish private and authenticated communication based on their shared (low-entropy) password (see e.g. [1]) and [2]). SPEKE is a version of PAKE based on Diffie-Hellman key exchange where the generator g is created by a deterministic function over the password - e.g. a hash value (see [3], and its security proof in [4]). However, it is susceptible to password guessing attack, where an adversary is able to test multiple possible passwords in a single impersonation attempt [5].

PACE. Password Authenticated Connection Establishment (PACE) protocol [6] was proposed by the German Federal Office for Information Security (BSI) for the deployment in MRTD. The purpose of PACE is to establish a secure channel based on weak passwords, like the personal data of the passport holder. Subsequent run of additional Active Authentication (AA) protocol verifies the passport's validity. It has been approved as a standard solution by International Civil Aviation Organization (ICAO) [7]. A slightly different design proposed by Gemalto and Sagem Securite is called PACE-IM [7]. This paper concerns the German version PACE v1 Generic Mapping (PACE-GM).

PACE|AA. PACE|AA [8] is an extension of PACE combining it with active authentication in a single protocol. The main idea is to reuse some randomness from the steps of PACE protocol to construct a modified Schnorr signature over a card identity and send it as an active authentication (AA) message at the end of the PACE|AA.

Security Models. There are many security models for the problems concerned in this paper. PACE protocol was proved to be secure in the Abdalla et al. model of authenticated key exchange (AKE) [9], a model stronger than the classical one from [1]. Several other papers describe security requirements of AKE protocols in different model settings; relations between these models were discussed in [10]. Possible attack scenarios including: various key compromising settings e.g. [11,12] and impersonation threats e.g. [13,14] were addressed accordingly. Another model [15] captures various leakage threats in AKE protocols of [16,17]. This repeatedly occurring model upgrading shows that we have to be very careful about their practical relevance and completeness.

Our Contribution. Our goal is to simplify and fine tune the PACE|AA protocol. We aim to achieve this with minimal changes to both PACE and AA in order to ease implementation and backward compatibility. We propose a SPACE|AA protocol that

- like PACE|AA, differs from PACE by just one step on each side and one message sent after executing PACE,
- it is simplified conceptually compared to PACE|AA,
- it has more transparent properties regarding non-tracebility,
- computational complexity is slightly improved compared with PACE|AA,
- secret keys for active authentication are much better protected in case of leakage of ephemeral secrets from the MRTD chip.

Our idea is to replace modified Schnorr signature with a version static of Diffie-Hellman authentication with partial disclosure of exponents.

Apart from a new optimized construction, we discuss a security model for this kind of protocols. We examine protocols in a broader context and in a more application specific (protocols for MRTD-s) way than the Abdalla et al. model used to analyze PACE and PACE|AA. We propose an approach that shows how the attack and adversary scenarios have been found by systematic search over all cases.

We consider rigorously non-transferability of the proof of presence and security threats concerning tracebility of smart cards. These issues must have been in mind of designers of PACE, but yet have not been covered by published security models.

2 Simplified PACE|AA Protocol

Our protocols (see specification below) bind one of the ephemeral keys and the private key x_A of a card A in a simplier way than for PACE|AA. The first base construction is the same as PACE|AA, but instead of sending a modified Schnorr signature the card sends $w := y_A/x_A$, where g^{x_A} is the public key of A. Thereby the PACE part of the protocol remains intact, and the reader can verify the card by checking that $X_A^w = Y_A$. This base version targets the requirements of the full compatibility with PACE part.

However, we recognize the shortcomings of these solutions in case of successful attacks on ephemeral data. This can be a serious security threat, since a less secure memory may be used to store ephemeral keys. The leakage of the ephemeral secret y_A makes long-term secret x_A easily computable, both in PACE|AA, and in our base version of the protocol. Therefore we propose the Ephemeral Key Leakage Resilient

Variant of SPACE|AA, in which the agreement of the first shared key h with Diffie-Hellman protocol is replaced by a blinded static Diffie-Hellman protocol where the static long-term key x_A is used on the side of the card. Namely, the card sends $Y_A = g^{x_A y_A} = X_A^{y_A}$ for y_A chosen at random and afterwards discloses $w := y_A$. Note that our variants differ only by implementation details on the MRTD chip (and not on the reader).

Table 1. PACE|AA and Simplified PACE|AA (SPACE|AA) protocols

Card		**Reader**
π - password, x_A - private key $X_A = g^{x_A}$ - public key $cert_A$ - certificate for X_A $\mathcal{G} = (a, b, p, q, g, k)$ - parameters		π - password, input from owner
$K_\pi := H(0\|\|\pi)$ choose at random $s \leftarrow \mathbb{Z}_q$		$K_\pi := H(0\|\|\pi)$
$z := ENC(K_\pi, s)$	$\xrightarrow{\mathcal{G}, z}$	abort if $\mathcal{G}$ incorrect $s := DEC(K_\pi, z)$
choose $y_A \leftarrow \mathbb{Z}_q^*$		choose $y_B \leftarrow \mathbb{Z}_q^*$
Version: SPACE|AA, Original PACE Variant		
$Y_A := g^{y_A}$	$\xleftarrow{Y_B}$	$Y_B := g^{y_B}$
Version: SPACE|AA, Ephemeral Key Leakage Resilient Variant		
$Y_A := g^{x_A y_A}$	$\xleftarrow{Y_B}$	$Y_B := g^{y_B}$
Variants End		
abort if $Y_B \notin \langle g \rangle \backslash \{1\}$	$\xrightarrow{Y_A}$	abort if $Y_A \notin \langle g \rangle \backslash \{1\}$
Version: SPACE|AA, Original PACE Variant		
$h := Y_B^{y_A}$		$h := Y_A^{y_B}$
Version: SPACE|AA, Ephemeral Key Leakage Resilient Variant		
$h := Y_B^{x_A y_A}$		$h := Y_A^{y_B}$
Variants End		
$\hat{g} := h \cdot g^s$ choose $y'_A \leftarrow \mathbb{Z}_q^*$		$\hat{g} := h \cdot g^s$ choose $y'_B \leftarrow \mathbb{Z}_q^*$
$Y'_A := \hat{g}^{y'_A}$	$\xleftarrow{Y'_B}$	$Y'_B := \hat{g}^{y'_B}$
check $Y'_B \neq Y_B$	$\xrightarrow{Y'_A}$	check $Y'_A \neq Y_A$
$K := {Y'_B}^{y'_A}$ $K_{ENC} := H(1\|\|K)$ $K'_{SC} := H(2\|\|K)$ $K_{MAC} := H(3\|\|K)$ $K'_{MAC} := H(4\|\|K)$		$K := {Y'_A}^{y'_B}$ $K_{ENC} := H(1\|\|K)$ $K'_{SC} := H(2\|\|K)$ $K_{MAC} := H(3\|\|K)$ $K'_{MAC} := H(4\|\|K)$
$T_A := MAC(K'_{MAC}, (Y'_B, \mathcal{G}))$	$\xleftarrow{T_B}$	$T_B := MAC(K'_{MAC}, (Y'_A, \mathcal{G}))$
abort if T_B invalid	$\xrightarrow{T_A}$	abort if T_A invalid

Table 1. (*Continued.*)

Version: PACE|AA, Deniable Schnorr Variant	
$\sigma := y_A + H(5\|\|Y_A, Y'_A, \mathcal{G}) \cdot x_A$	$\xrightarrow{ENC(K'_{SC},(\sigma.cert_A))}$
	decrypt the message with K'_{SC} check certificate $cert_A$ $w := \sigma^{-1}$, $r := Y_A$ $v := g^{wH(5\|\|Y_A,Y'_A,\mathcal{G})} X_A^{rw}$ abort if $v \neq Y_A$
Version: SPACE|AA	
$w := y_A/x_A$	
Version: SPACE|AA, Ephemeral Key Leakage Resilient Variant	
$w := y_A$	
Variants End	
	$\xrightarrow{ENC(K'_{SC},(w,cert_A))}$ decrypt the ciphertext with K'_{SC}
	check certificate $cert_A$ abort if $X_A^w \neq Y_A$

3 Security

Security of key exchange and authentication protocols have been considered by many authors, however many essential security features for MRTD applications are frequently not covered, even if many protocol designers have appropriate properties in mind. Our goal is a systematic approach which has to be complete in the sense that all relevant threats and attack scenarios are identified. Despite simplicity of SPACE|AA this leads to a complex analysis.

Security Targets: As starting point we list the situations where the actors of the system perform some actions or get secrets and data to which they are not entitled:
Privacy: an adversary may try to trace a MRTD chip (Trace attack), or convince a third party about an interaction that really took place (Transfer attack).
Forgery: an adversary may try to mimic the MRTD chip and try to convince a reader that it is talking with this MRTD chip (FakeCard). The second situation is that a reader mimics the reader holding the right password while talking with an MRTD chip (FakeReader). The target is to allude the consent of document holder to activate the chip. The attacks of this kind concern man-in-the-middle and hijacking a session.
Confidentiality: an adversary may try to get access to information transferred between the MRTD chip and the reader in a legitimate communication when the adversary is not controlling the reader. In particular, the adversary may try to derive the session keys, analyze session transcripts and modify interaction between a chip and a reader.
Long Term Key: an adversary may try to get the long term key x of a chip, specially when the ephemeral keys become insecure (e.g. via implementation errors).
Assumptions: For the sake of security analysis we assume that the scheme computations are done within an appropriate algebraic group where DLP, CDH, DDH assumptions hold. Moreover we assume that the hash function H used is modelled by the random oracle.

3.1 Adversary Categories

We consider different cases depending on whether the adversary has the secret key x for authentication and/or the password π. We use notation $\mathcal{A}$, $\mathcal{A}_x$, $\mathcal{A}_\pi$, $\mathcal{A}_{x,\pi}$ to denote adversaries that have, respectively, neither x nor π, only x, only π, both x and π. $\mathcal{A}_{\bullet\bullet}$ denotes adversaries, where $\bullet$ can be set x, π or blank. Note that $\mathcal{A}_\pi$ scenario is quite realistic, since the password is revealed to readers. $\mathcal{A}_x$ corresponds to the case when cryptanalysis reveals the private key of the MRTD chip. $\mathcal{A}_{x,\pi}$ corresponds to the case when the MRTD issuer misbehaves and retains copies of the secrets of the MRTD chip.

A starting point of an attack might be upgrading the adversary's category. Note also that in some scenarios an adversary has obvious advantages. For instance, $\mathcal{A}_{x,\pi}$ can act as the MRTD chip holding x and π, while $\mathcal{A}_\pi$ can start an interaction with the MRTD chip as a legitimate reader. Our goal is to show that the adversary cannot gain more.

In many cases it suffices to show immunity against an attack in the strongest adversary category: e.g. it suffices to show secrecy of the session key in the model $\mathcal{A}_{x,\pi}$. However, sometimes we cannot make such a reduction. For instance, we cannot do it for the above mentioned transfer attack – in general more secrets may mean more capabilities to create a fake protocol transcript.

As in [8], for the first version of SPACE|AA we assume that the ephemeral random parameters used as auxiliary data for active authentication are secure.

3.2 Privacy Games

Now we formulate games describing attacks aiming to break privacy. Usually security games involve parameters saying how many times certain operations are executed. In most cases we omit such parameters as they can be easily derived by the Reader.

Definition 1 (Trace Game). *The goal of the adversary is to decide if a traced communication belongs to a given card.*

Phase 1: *Challenger picks at random MRTD Chips C_0, C_1.*
Phase 2: *$\mathcal{A}^{TR}_{\bullet\bullet}$ can ask for an arbitrary number of interactions between C_0 (or C_1) and a reader knowing the password of, respectively, C_0 or C_1. $\mathcal{A}^{TR}_{\bullet\bullet}$ can arbitrarily replace any message communicated before it is delivered to the other party of the protocol (hence he can himself execute the protocol with C_0 and C_1).*
Phase 3: *Challenger picks a bit b at random.*
Phase 4: *$\mathcal{A}^{TR}_{\bullet\bullet}$ performs the same actions as in Phase 2, but now the first group of interaction is with C_b and the second one is with C_{1-b}. If $\mathcal{A}^{TR}_{\bullet\bullet}$ has the password of C_0 or C_1, then in this phase he cannot change any message.*
Phase 5: *perform the same steps as in Phase 2.*
Phase 6: *$\mathcal{A}^{TR}_{\bullet\bullet}$ answers with a bit $\bar{b}$. Finally, $\mathcal{A}^{TR}_{\bullet\bullet}$ wins if $\bar{b} = b$.*

We can also assume that the adversary knows the keys from all MRTD chips other than C_0 and C_1 and can use both the keys and the correcponding MRTD chips arbitrarily. We define the advantage of winning the Trace Game as $|\Pr[\bar{b} = b] - \frac{1}{2}|$.

Of course, during Phase 4 we cannot let $\mathcal{A}^{TR}_{\bullet\bullet}$ to interact with C_b and C_{1-b}, if he has at least one of the passwords, as in this case he may execute SPACE|AA authenticating

one of the chips. Note that it suffices to prove security for the scenarios $\mathcal{A}_x$ and $\mathcal{A}_{\pi,x}$, as the advantage of $\mathcal{A}$ is bounded by the advantage of $\mathcal{A}_x$, and the advantage of $\mathcal{A}_\pi$ is bounded by the advantage of $\mathcal{A}_{\pi,x}$.

Definition 2 (Transfer Game). *There are Prover and Verifier; each of them may hold some secrets of an MRTD chip C. Prover has to convince Verifier that at a given time it communicates with C – e.g. by presenting a communication transcript. This is described by the following game:*

Phase 1: *The Prover picks MRTD chip C. Prover and Verifier as well may perform any number of interactions with C. If Prover is interacting with C, then Verifier may observe the communication and even influence it in any way.*

Phase 2: *Prover chooses $b \in \{0, 1\}$ at random. If $b = 0$, then Prover interacts with C in any way as well as observes interaction between C and third parties. In this case Prover records the transcript T of his communication with C. If $b = 1$, then Prover creates a faked transcript T by itself, in absence of C. Finally, Prover presents T to Verifier.*

Phase 3: *Verifier returns a bit $\bar{b}$. He wins if $\bar{b} = b$.*

We define the advantage of winning the Transfer Game as $|\Pr[\bar{b} = b] - \frac{1}{2}|$.

Note that Transfer Game must be considered for each type of adversaries, as lack of secrets might increase the strength of the proof by using the argumnet like: *"I could not create it without the secrets that are used by the protocols parties"*. The proof may be created during a faulty execution as well (e.g. by adversary $\mathcal{A}$ with no secrets). We also have to consider Verifiers holding x or π, or even both x and π.

3.3 Faking Games

Definition 3 (FakeCard1 Game). *The adversary's goal is to authenticate as an MRTD chip C against a reader that knows the password of the chip C.*

Phase 1: *The adversary $\mathcal{A}_{\bullet\bullet}^{FC1}$ chooses an MRTD chip C.*

Phase 2: *$\mathcal{A}_{\bullet\bullet}^{FC1}$ may listen to an arbitrary number of real interactions between C and a reader knowing password of C. During the interaction, the adversary may replace any message sent with a message of his choice. $\mathcal{A}_{\bullet\bullet}^{FC1}$ may also interact with any other MRTD chips and break their keys.*

Phase 3: *$\mathcal{A}_{\bullet\bullet}^{FC1}$ executes SPACE|AA with a reader knowing the password π of C.*

$\mathcal{A}_{\bullet\bullet}^{FC1}$ wins if the reader executes successfully the whole protocol and does not abort.

We define the advantage of winning FakeCard1 Game as probability that the reader does not abort. Note that we do not demand $\mathcal{A}_{\bullet\bullet}^{FC1}$ to learn the session key. FakeCard1 does not make sense for $\mathcal{A}_{x,\pi}^{FC1}$ as in this case the adversary has a fully functional copy of the MRTD chip. The advantage of $\mathcal{A}^{FC1}$ is bounded by both the advantage of $\mathcal{A}_\pi^{FC1}$ and $\mathcal{A}_x^{FC1}$, so it suffices to consider FakeCard1 Game for $\mathcal{A}_\pi^{FC1}$ and $\mathcal{A}_x^{FC1}$.

Definition 4 (FakeCard2 Game). *The adversaries goal is to authenticate as MRTD chip C against a reader that does not know the password π of C. The game is exactly the same as in case of FakeCard1 except that the adversary may enter an arbitrary password π' to the reader.*

For $\mathcal{A}^{FC2}_{x,\pi}$ and $\mathcal{A}^{FC2}_{x}$, as in both cases the adversary succeeds by definition. So it suffices to consider $\mathcal{A}^{FC2}_{\pi}$.

Definition 5 (FakeReader Game). *The adversary's goal is to execute SPACE\|AA with an MRTD chip without having the proper password, i.e. pretending a legitimate reader.*

Phase 1: *The adversary $\mathcal{A}^{FR}_{\bullet\bullet}$ chooses an MRTD chip C.*
Phase 2: *$\mathcal{A}^{FR}_{\bullet\bullet}$ may listen to an arbitrary number of interactions of C with a reader knowing the password of C. During the interaction, the adversary may replace any message sent with a message of his choice. $\mathcal{A}^{FR}_{\bullet\bullet}$ may also interact with any other MRTD chips and break their keys.*
Phase 3: *C executes SPACE\|AA protocol with $\mathcal{A}^{FR}_{\bullet\bullet}$.*

$\mathcal{A}^{FR}_{\bullet\bullet}$ wins if C terminates SPACE\|AA without aborting it prematurely.

The advantage of winning the FakeReader Game is the probability that $\mathcal{A}^{FR}_{\bullet\bullet}$ wins. By definition, $\mathcal{A}^{FR}_{\bullet,\pi}$ can win the game. So it suffices to consider $\mathcal{A}^{FR}_{x}$, as the advantage of $\mathcal{A}^{FR}$ is bounded by the advantage of $\mathcal{A}^{FR}_{x}$.

3.4 Transmission Security

Since SPACE|AA has to create a secure channel for information exchange between an MRTD chip and a reader, we have to make sure that there is no information leakage. We use here a variant of real-or-random paradigm of Abdalla et al. [9]. We formulate a strong game that encompasses for instance such issues as Perfect Forward Security.

Definition 6 (Transmission Security Game)

Phase 1: *Adversary $\mathcal{A}^{TS}_{\bullet\bullet}$ picks an MRTD chip C.*
Phase 2: *$\mathcal{A}^{TS}_{\bullet\bullet}$ may listen to an arbitrary number of interactions with C of the readers knowing the password of C. $\mathcal{A}^{TS}_{\bullet\bullet}$ may replace any message sent by a message of its choice. $\mathcal{A}^{TS}_{\bullet\bullet}$ may also interact with any other MRTD chips and break their keys.*
Phase 3: *C performs SPACE\|AA up to the moment immediately before exchanging messages T_B and T_A. $\mathcal{A}^{TS}_{\bullet\bullet}$ may replace or modify any message sent, however $\mathcal{A}^{TS}_{\bullet,\pi}$ cannot modify both Y_B and Y'_B sent to C by the reader.*[1]
Phase 4: *The challenger chooses a bit b. If $b = 1$, then the session keys K_{ENC}, K_{MAC}, K_{SC}, K'_{MAC} are replaced by the keys R_{ENC}, R_{MAC} R'_{SC}, R'_{MAC} chosen at random.*
Phase 5: *Exchange and verification of T_B and T_A is executed. Active authentication step is executed.*
Phase 6: *The challenger reveals K_{ENC} and K_{MAC}, if $b = 0$, or R_{ENC} and R_{MAC}, if $b = 1$.*
Phase 7: *$\mathcal{A}^{TS}_{\bullet\bullet}$ returns a bit $\bar{b}$. $\mathcal{A}^{TS}_{\bullet\bullet}$ wins if the interaction between C and the reader is not aborted and $\bar{b} = b$.*

[1] Note that otherwise $\mathcal{A}^{TS}_{\bullet,\pi}$ could connect with C and eliminate the attacked reader completely. However, this is legitimate as $\mathcal{A}^{TS}_{\bullet,\pi}$ knows π.

We define the advantage of winning the Transmission Security Game as $|\Pr[\bar{b} = b] - \frac{1}{2}|$. It suffices to prove security against $\mathcal{A}^{TS}_{x,\pi}$ and $\mathcal{A}^{TS}_{x}$, as the advantage of $\mathcal{A}^{TS}$ is bounded by the advantage of $\mathcal{A}^{TS}_{x}$, and the advantage of $\mathcal{A}^{TS}_{\pi}$ is bounded by the advantage of $\mathcal{A}^{TS}_{x,\pi}$.

Of course, the Transmission Security Game does not reflect directly the real attack scenario. However, if the advantage in the game above is negligible, then any attack against SPACE|AA deriving information from the workload transmission would also work when the session keys K_{ENC}, K_{MAC} are replaced by random keys (both on the MRTD chip and on the reader). However, if the session keys are chosen at random, then the data from SPACE|AA protocol execution bring no advantage for the adversary.

Remark 1. The considered security property is quite strong: even if the adversary holds a clone of a card C, the only way to get the information sent by C is to replace entirely the reader. Of course, this is possible only in case of $\mathcal{A}^{TS}_{\bullet,\pi}$, which is acceptable in the application model concerned.

3.5 Long Term Key Security on Ephemeral Data Leakage

Definition 7 (Get-x-Ephemeral-Leakage Game). *The adversary's goal it to get the authentication key x of a chip, in case the ephemeral keys are leaked.*

Phase 1: *Adversary $\mathcal{A}^{EK}_{\pi}$ chooses an MRTD chip C with the public key $X = g^x$.*
Phase 2: *$\mathcal{A}^{EK}_{\pi}$ gets all transcripts $\{T_i\}_{i=1}^{q}$ of protocol runs of C, including all results of computations on the chip and the reader, altogether with respective ephemeral keys, and session keys, except for x. Also, $\mathcal{A}^{EK}_{\pi}$ may interact with all other MRTD chips and break their keys.*
Phase 3: *$\mathcal{A}^{EK}_{\pi}$ outputs a key x'. $\mathcal{A}^{EK}_{\pi}$ wins if $x' = x$.*

We define the advantage of $\mathcal{A}^{EK}_{\pi}$ as the probability of its winning the Get-x-Ephemeral-Leakeage Game. Note that for PACE|AA the adversary may easily win the game.

4 Security Proofs

4.1 Simulatability

One of the fundamental properties of SPACE|AA is that transcripts of interactions with an MRTD chip can be created or simulated without knowledge of the secrets x and π of the MRTD chip.

Basic transcript. If the adversary is by monitoring communication with the MRTD chip, then a basic transcript consists of the following values:

$$z, Y_B, Y_A, Y'_B, Y'_A, T_B, T_A, \mathrm{ENC}(K'_{\mathrm{SC}}, (w, cert_A)).$$

Full transcript. A reader can leak all data it creates during an interaction with an MRTD chip. In this case the transcript of interaction consists of the following data:

$$\pi, z, y_B, Y_A, y'_B, Y'_A, T_B, T_A, w, cert_A.$$

Full transcript without password. Transcripts generated when the reader is not using the correct password. The transcript contains:

$$z, y_B, Y_A, y'_B, Y'_A, T_B.$$

Note that the transcript contains neither T_A nor $\mathrm{ENC}(K'_{\mathrm{SC}}, (w, cert_A))$ as the MRTD chip interrupts the communication. Of course, it may happen also when the password used by the reader was correct and the communication was not interrupted.

Lemma 1. *There is a simulator S_{full} that creates full transcripts with exactly the same probability distribution as the full transcripts of real interactions with an MRTD chip C, provided that the simulator has the password π of C.*

Proof. S_{full} chooses w at random and computes $Y_A := X_A^w$. Also, it chooses the following elements at random: z, Y'_A, y_B, and y'_B. The remaining elements are computed according to the specification of SPACE|AA protocol.

It is easy to see that Y_A has the same probability distribution as Y_A generated in the original way. Indeed, it does not matter whether we choose y_A at random or take $y_A = x_A \cdot w$, where w is chosen at random, as the order of the group is a prime. The same argument applies for y'_A. □

Lemma 2. *There is a simulator S_{nopwrd} that creates transcripts that have exactly the same probability distribution as the full transcripts without password for an MRTD chip A and a reader R executing a strategy St.*

Proof. The simulator chooses Y_A and Y'_A, z at random. Then it provides the remaining values created by the reader according to the same strategy St as used by the reader in a real interaction. As in Lemma 1, we can choose Y'_A directly. Note that we can choose Y_A at random without w, as w will not be included in the transcript. □

Lemma 3. *There is a simulator S_{basic} that creates transcripts that are indistinguishable from the basic transcripts for an MRTD chip A provided that DDH Assumption holds.*

Proof. The proof is similar to the previous two lemmas, however, there are some problems, as the simulator cannot derive $\hat{g}$ according to specification of SPACE|AA. However, due to DDH Assumption, the simulator can replace the shared key h with a random key. With the same effect, $\hat{g} = h \cdot g^s$ can be replaced with a random element.

The element Y_A is computed just as in case of the simulator from Lemma 1. The elements z, Y'_B, K are computed at random. The remaining necessary elements are computed according to the specification of SPACE|AA. Again, note that K derived in this way is indistinguishable from the key derived according to SPACE|AA. □

Note that the results of Lemmas 1, 2 and 3 hold even if a party trying to distinguish the transcripts holds π and x of the MRTD chip.

4.2 Drafts of Security Proofs

By Lemmas 1, 2 and 3, Prover may create transcripts of interaction with C that are indistinguishable from real interactions. Therefore we get easily the following result:

Theorem 1. *In case of SPACE|AA, probability of winning the TRANSFER Game for each adversary scenario is negligible.*

Theorem 2. *Probability of winning Trace Game by $\mathcal{A}^{TR}_{x,\pi}$ for SPACE|AA protocol is negligible provided that the DDH Assumption holds and the encryption function is a semantically secure permutation.*

Proof (Draft). According to Lemma 1, Phases 2 and 5 can be omitted, as $\mathcal{A}^{TR}_{x,\pi}$ can replace them with a simulator. In Phase 4 $\mathcal{A}^{TR}_{x,\pi}$ receives only transcripts of communication performed by C_b and C_{1-b}, since it knows their passwords.

Again by Lemma 1 transcripts from Phase 4 can be created using a simulator. Note that by definition the simulator chooses the elements z and K at random. Hence, the only value that can be used to link cards is $\mathit{ENC}(K'_{SC}, (w, cert_A))$. However, the key K'_{SC} is derived using a random key K thus we can replace the plaintext in $\mathit{ENC}(K'_{SC}, (w, cert_{C_i}))$ by a random string U of the same size. □

The second case which we have to consider is slightly different as the adversary can be active. However, he does not know the correct password and can guess it at most.

Theorem 3. *Assume that DDH Problem is hard and that the MAC function is deterministic. Then probability of winning Trace Game by active $\mathcal{A}^{TR}_{x}$ adversary cannot be non negligibly higher than probability of successful authentication in PACE without knowing the right password.*

Proof (Draft). We shall first show that if the adversary changes at least one message until delivery of T_B, then the MRTD chip C_i aborts after verification of T_B.

If only T_B is changed, then C_i aborts as derivation of K, K'_{SC} and the *MAC* function are deterministic. If the adversary does not know discrete logarithm of either Y_B or Y'_B, then computing either h or K would mean solving CDH Problem. However, we shall prove in Theorem 9 and 10 that possibility to derive a message T_B acceptable by C_i is equivalent with solving DDH Problem.

In order to know discrete logarithm of Y_B and Y'_B sent by the adversary to C_i, the adversary must replace the values sent by the reader. However, finding the right T_B or reshaping the T_B sent by the reader would mean successful execution of PACE with C_i without knowing the correct password. Thus C_i interrupts the protocol execution with high probability after receiving T_B.

Now observe that since Y'_A (computed by C_i) and Y'_B (computed by the reader) are random values from the point of view of the adversary it remains to show that T_B gives no information about C_i. It is easy to see that the key K used to compute T_B depends on values Y'_A (possibly changed by the adversary) and y'_B (random exponent of the reader). In case when Y'_A is transmitted unchanged, then T_B is random from the point of view of the adversary. Finally, notice that if Y'_A is changed by the adversary, then T_B depends only on values chosen by the adversary and the reader, thus T_B can be computed without communication with C_i i.e. without using the password π_i of C_i.

□

Theorem 4. *Non-negligible probability of winning the FakeCard1 Game by $\mathcal{A}^{FC1}_{\pi}$ for SPACE|AA leads to a non-negligible probability of solving CDH Problem.*

Proof (Draft). We shall consider a malignant reader that tries to force $\mathcal{A}_{\pi}^{FC1}$ to solve cases of CDH Problem. Let (g, g^x, g^y) be the case of CDH, so the goal is to force $\mathcal{A}_{\pi}^{FC1}$ to compute g^{xy}. First, $\mathcal{A}_{\pi}^{FC1}$ sets the public key of MRTD chip C to $X_C = g^x$.

Note that we can eliminate entirely Phase 2 of FakeCard1 Game. Indeed, it can be performed by $\mathcal{A}_{\pi}^{FC1}$ without any contact with MRTD chip C but with a simulator (see Lemma 1) and so the adversary cannot learn anything during this phase.

Now assume that $\mathcal{A}_{\pi}^{FC1}$ succeeds in Phase 3. The malignant reader tosses a coin, with probability $\frac{1}{2}$ it behaves according to the protocol and computes $Y_B := g^{y_B}$. However, with probability $\frac{1}{2}$ it uses $Y_B := (g^y)^r$, where r is chosen at random. In this case the reader is unable to create T_B (as it does not know the discrete logarithm of Y_B and therefore cannot derive h and consequently cannot derive K) and the interaction will terminate after sending T_B. However, even if $\mathcal{A}_{\pi}^{FC1}$ knows about the malignant reader, inevitably it can react as for the correct protocol execution in case when the reader is using the second option. Note that T_B sent by the reader cannot be used by the adversary to derive the key K'_{SC} since this would imply performing a key recovery attack on the MAC scheme and a preimage attack on the hash function H by $\mathcal{A}_{\pi}^{FC1}$.

The only w that is positively verified by the honest reader is y_a/x. So no matter how $\mathcal{A}_{\pi}^{FC1}$ gets Y_A, we have $y_A = x \cdot w$. Moreover, in order to derive the key K that matches the key used by the honest reader, $\mathcal{A}_{\pi}^{FC1}$ has to compute $h = Y_B^{y_A}$. If the reader is malignant, then $h = (g^{yr})^{x \cdot w} = g^{xy \cdot rw}$. Note that computing h is equivalent to computing g^{xy}, as w is known by $\mathcal{A}_{\pi}^{FC1}$ and r can be revealed to $\mathcal{A}_{\pi}^{FC1}$. □

The second case is an adversary that knows key x, but not π. Note that $\mathcal{A}_{x}^{FC1}$ can be immediately converted into an adversary against PACE: the new adversary simply skips the last active authentication step. The same reasoning can be used in the FakeReader Game for the adversary $\mathcal{A}_{x}^{FR1}$. So we have:

Theorem 5. *If $\mathcal{A}_{x}^{FC1}$ can win FakeCard1 Game with a probability p, then it is possible to win FakeCard1 Game against PACE protocol with the same probability.*

Theorem 6. *If $\mathcal{A}_{x}^{FR1}$ can win FakeReader Game with a probability p, then it is possible to win FakeReader Game against PACE protocol with the same probability.*

The strategy to prove security against Transmission Security Game is the same as in [8]. However, we shall see that it is possible to extend the argument a little bit to incorporate active attacks.

Theorem 7. *Probability of winning the Transmission Security Game by $\mathcal{A}_{x,\pi}^{TS}$ which is passive during execution of SPACE\|AA is negligible, provided that DDH Problem is hard.*

Proof (Draft). Again Phase 2 can be omited using a simulator. Note that if the adversary is passive during the execution of SPACE|AA, then by Lemma 3 the challenger can compute a transcript and present a part of it $T_1 = (z, Y_A, Y_B, Y'_A, Y'_B)$ in Phase 3 to the adversary. In the next phases the remaining part $(T_A, T_B, ENC(K'_{SC}, (w, cert_A)))$ is presented to the adversary, if $b = 1$, and is recomputed using random keys, if $b = 0$. Finally, note that since the simulator from Lemma 3 chooses the key K at random, hence the keys reaveled in Phase 6 (independently of b) are random and do not depend on the values from transcript T_1. Hence the probability that $\bar{b} = b$ is equal to $\frac{1}{2}$. □

When considering Transmission Security Game for an active adversary, note that before C sends the active authentication message, it must accept T_B. Therefore, it suffices to consider modifications Y_B and Y'_B delivered to the MRTD chip C, as only these messages influence the key K computed on the side of C.

Theorem 8. *Probability of winning Transmission Security Game by $\mathcal{A}_x^{TS}$ adversary which is active during execution of SPACE\|AA is at most negligibly higher than probability of FakeReader Game for PACE (for an adversary not holding the password).*

Proof (Draft). Note that in case of SPACE|AA the adversary gets one extra message - namely the authentication message sent by the card C. However, this message is sent provided that T_B is accepted by C. As the protocol execution up to this step is identical with PACE and does not depend on active authentication secret x, we get thereby an adversary $\mathcal{A}$ that succeeds as a reader without the password against PACE. □

The last case is the adversary $\mathcal{A}_\pi^{TS}$ which has to transmit at least one of the original messages Y_B, Y'_B exchanged between the chip C and a reader holding password π.

Theorem 9. *Probability of winning Transmission Security Game by $\mathcal{A}_\pi^{TS}$ adversary which delivers to C the original Y'_B sent by a reader interacting with C is negligible, provided that DDH Problem is hard, and that the MAC algorithm is deterministic.*

Proof (Draft). Assume that there is an adversary $\mathcal{A}_\pi^{TS}$ winning Transmission Security Game considered in the theorem. Let $(\hat{g}, Y'_A, Y'_B, Z)$ be an input for DDH Problem. We can construct a transcript of SPACE|AA protocol up to the moment when T_B is sent, using the elements $\hat{g}, Y'_A, Y'_B$. Then we apply the strategy of adversary $\mathcal{A}_\pi^{TS}$ to generate T_B. On the other hand, we generate T_B for the key Z and check if the results coincide. If yes, then the algorithm says that (Y'_A, Y'_B, Z) is a DH-triple.

It is easy to see that in this way we provide an algorithm having non-negligible advantage in solving DDH Problem. □

The same holds if $\mathcal{A}_\pi^{TS}$ delivers the original Y_B instead of Y'_B. The proof is essentially the same as in case of Theorem 9:

Theorem 10. *Probability of winning Transmission Security Game by $\mathcal{A}_\pi^{TS}$ adversary which delivers to C the original Y_B sent by a reader interacting with C is negligible, provided that DDH Problem is hard, and that the MAC algorithm is deterministic.*

Theorem 11. *The probability of winning the Get-x-Ephemeral-Leakage Game by $\mathcal{A}_\pi^{EK}$ for the Ephemeral Key Leakage Resilient Variant of SPACE\|AA protocol is negligible, provided that DLP Assumption holds.*

Proof (Draft). Assume that the advantage of $\mathcal{A}_\pi^{EK}$ is not negligible. Then we break the DLP assumption, for a given instance g^a: we treat the g^a as the public key $X = g^x$ of the attacked card. We create transcripts $\{T_i\}$ by choosing for each T_i ephemeral keys y_A, y_B, y'_A, y'_B at random, put $Y'_A := X_A^{y_A} = (g^a)^{y_A}$, and subsequently compute all the intermediate protocol values for the chip and the reader. Such transcripts are given to the adversary $\mathcal{A}_\pi^{EK}(\{T_i\}_{i=1}^q)$ who yields a with a non negligible probability. □

Acknowledgment. We would like to thank Jens Bender and Dennis Kügler for discussions and German BSI for cooperation leading to development of the protocol presented in this paper. In particular, the idea of SPACE|AA emerged independently in BSI.

References

1. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: [18], pp. 139–155
2. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using diffie-hellman. In: [18], pp. 156–171
3. Jablon, D.P.: Extended password key exchange protocols immune to dictionary attacks. In: WETICE, pp. 248–255. IEEE Computer Society (1997)
4. MacKenzie, P.: On the security of the SPEKE password-authenticated key exchange protocol. Cryptology ePrint Archive, Report 2001/057 (2001)
5. Zhang, M.: Analysis of the speke password-authenticated key exchange protocol. IEEE Communications Letters 8(1), 63–65 (2004)
6. Bender, J., Fischlin, M., Kügler, D.: Security analysis of the PACE key-agreement protocol. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 33–48. Springer, Heidelberg (2009)
7. ISO/IEC JTC1 SC17 WG3/TF5 for the International Civil Aviation Organization: Supplemental access control for machine readable travel documents. Technical Report (2011)
8. Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: The PACE|AA protocol for machine readable travel documents, and its security. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 344–358. Springer, Heidelberg (2012)
9. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
10. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
11. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
12. Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
13. Chen, L., Tang, Q.: Bilateral unknown key-share attacks in key agreement protocols. J. UCS 14(3), 416–440 (2008)
14. Tang, Q., Chen, L.: Extended KCI attack against two-party key establishment protocols. Inf. Process. Lett. 111(15), 744–747 (2011)
15. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A new security model for authenticated key agreement. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 219–234. Springer, Heidelberg (2010)
16. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
17. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
18. Preneel, B. (ed.): EUROCRYPT 2000. LNCS, vol. 1807. Springer, Heidelberg (2000)

Expressing User Access Authorization Exceptions in Conventional Role-Based Access Control

Xiaofan Liu[1,2], Natasha Alechina[1], and Brian Logan[1]

[1] School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK
[2] School of Computer and Communication, Hunan University, Hunan, 410081, P. R. China
{lxx,nza,bsl}@cs.nott.ac.uk

Abstract. In this paper we present a systematic categorization of the user access authorization exceptions which may occur in conventional role-based access control models. We propose a slightly revised NIST RBAC model which allows us to express all the authorization exceptions we consider. We give a formal definition of the model and show how it can be implemented in DATALOG with negation to give simple and efficient algorithm for computing authorization decisions. As an illustration, we present a simple case study from the domain of medical informatics and show how a range of different kinds of authorization exceptions that may arise in such a domain can be expressed in our approach.

1 Introduction

Role-based access control (RBAC) models have been advocated as a way of reducing the complexity in discretionary and mandatory access control. In role-based access control, users are assigned to roles, and roles are associated with sets of permissions. A user request for access to a particular object (resource) is *authorized* if the user is assigned to a role that has the appropriate permission for the object. To simplify the management of permissions associated with roles, such models frequently utilize a role hierarchy which allows senior roles to implicitly include all the permissions associated with junior roles in the hierarchy. Conventional RBAC models, such as RBAC96 [15] and NIST RBAC [7] (which is based on RBAC96), adopt a closed policy, that is, a user has a particular permission if one of the user's roles has the permission, otherwise the user does not have the permission. However, in many practical applications, the positive authorizations allowed by an RBAC policy admit *user access authorization exceptions*, or *authorization exceptions* for short. For example:

Example 1. A medical records system has a set of roles including "Doctor" and "Cardiologist", some of which are associated with permissions that allow a user to read a patient's records. However, patient Alice may stipulate that a particular doctor, Tom (who happens to be her brother-in-law), should not have access to her records whichever role Tom is assigned to in order to protect Alice's privacy.[1]

[1] Experience in the UK suggests that patients sometimes wish to restrict access to their health records by their relatives or particular health care workers [2].

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 233–247, 2013.

Conventional RBAC models do not support such authorization exceptions to default access policies [2,3,13].

In RBAC models with role hierarchies (called *Hierarchical RBAC* in [7]), a user assigned to a role may be granted permissions both through their assigned role, and through junior roles from which their role inherits. In such models, in addition to the authorization exceptions discussed above, authorization exceptions may also result from role inheritance. Specifically, while users assigned to senior roles inherit all permissions associated with junior roles, some permissions associated with one or more junior roles should not be granted to users assigned to a senior role. For example:

Example 2. Assume that the role "IT supervisor" is senior to (and inherits permissions from) the role "IT professional". However, a permission that allows a user to alter source code associated with "IT professional" should not be inherited by "IT supervisor", because users assigned to "IT supervisor" have a background in management rather than in computer science.

Authorization exceptions in Core and Hierarchical RBAC have long been recognized, and proposals for handling some of those exceptions can be found in the literature [15,8,11,2,3,6,13]. However, to the best of our knowledge, there has been no systematic analysis of authorization exceptions in conventional RBAC. In this paper we provide an analysis of the range of possible authorization exceptions in conventional RBAC and propose extensions to the RBAC model that allow both core and inherited exceptions to be formalized. In addition, we show how our formalization of RBAC with exceptions can be expressed as a stratified program of recursive DATALOG with negation, giving a simple and efficient algorithm for computing authorization decisions.

The remainder of this paper is organized as follows. We briefly introduce the NIST RBAC model in Section 2. Authorization exceptions in core RBAC are explored in Section 3, and authorization exceptions existing in hierarchical RBAC are discussed in Section 4. In section 5, we introduce DATALOG with negation and show how to express our model in it. We conduct a case study in Section 6. Then, related work is shown in Section 7. Finally, we present our conclusions and indicate some potential areas of future work in Section 8.

2 NIST RBAC

The NIST RBAC model [7] (adopted as ANSI standard ANSI INCITS 359-2004) is arguably the most influential approach to RBAC. The NIST RBAC reference model is defined in terms of four model components: *Core RBAC*, *Hierarchical RBAC*, *Static Separation of Duty Relations* and *Dynamic Separation of Duty Relations*. We focus on the first two model components, Core RBAC and Hierarchical RBAC. In Core RBAC permissions are associated with roles and roles are assigned to users. Core RBAC is mandatory for all RBAC models. Hierarchical RBAC adds a role hierarchy, which defines an inheritance relation among roles, to Core RBAC. Informally, role r_1 inherits from role r_2 if users assigned to r_1 have all permissions associated with r_2. In what follows, we broadly follow the principles, definitions and reference models given in the NIST RBAC model. However, as proposed by Li et al [10] and Power et al [12], we do not consider sessions as defined in the Core RBAC model.

3 Authorization Exceptions in Core RBAC

In this section, we introduce the formal definition of the NIST Core RBAC model defined in [7]. We then define authorization exceptions in Core RBAC and show how to express these exceptions by adding new constructs into the NIST Core RBAC model definition.

3.1 NIST Core RBAC

The NIST RBAC model defines Core RBAC as follows [7]:

Definition 1. (NIST Core RBAC)

- *$USERS$, $ROLES$, ACS, and OBS denote sets of users, roles, actions, and objects, respectively.*
- *$UA \subseteq USERS \times ROLES$ is a many-to-many mapping user-to-role assignment relation.*
- *$assigned\_users(r : ROLES) \rightarrow 2^{USERS}$ maps a role r onto a set of users. Formally, $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA)\}$.*
- *$PRMS = 2^{ACS \times OBS}$ is the set of permissions.*
- *$PA \subseteq PRMS \times ROLES$ is a many-to-many mapping permission-to-role assignment relation.*
- *$assigned\_permissions(r : ROLES) \rightarrow 2^{PRMS}$ maps a role r onto a set of permissions. Formally, $assigned\_permissions(r) = \{p \in PRMS \mid (p, r) \in PA)\}$.*
- *$Ac(p : PRMS) \rightarrow \{ac \subseteq ACS\}$ is the permission-to-action mapping, which gives the set of actions associated with permission p.*
- *$Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$ is the permission-to-object mapping, which gives the set of objects associated with permission p.*

The definition of Core RBAC given in Definition 1 is somewhat redundant, as pointed out in [10]. Given $USERS$, $ROLES$, ACS, OBS, UA and PA, all the other relations and functions in the NIST definition of Core RBAC are definable. In the interests of brevity, we therefore omit the definable relations and functions. For technical reasons (the ease of expressing definitions in first-order logic and translation to DATALOG) we also "flatten" the set of permissions: instead of $PRMS = 2^{ACS \times OBS}$ we set it to be $PRMS \subseteq ACS \times OBS$.[2] This means that PA becomes a relation between a single permission tuple and a role, rather than between a set of tuples and a role.

More importantly, in the definition above, there is no relation explicitly connecting a user to a permission. Every user assigned to a role implicitly has all the permissions associated with that role. This key idea of RBAC allows a compact and transparent representation of an access control policy. However it does not allow authorization exceptions to be expressed. We therefore modify the NIST Core RBAC definition as follows: we add a predicate $AUTH$ which stands for "user is authorized". We add a condition (Core RBAC User Authorization) which states that if a user is assigned to a role, and a role is associated with a permission, then the user has (is authorized for) this permission:

$$\forall p \forall u \forall r(UA(u, r) \wedge PA(p, r) \rightarrow AUTH(p, u))$$

[2] In a given system, there may exist some action-object pairs that are not permissions.

Definition 2. (Core RBAC with User Authorization Relation)

- *$USERS$, $ROLES$, ACS, OBS, UA are as in Definition 1.*
- *$PRMS \subseteq ACS \times OBS$ is a set of permissions (a subset of the set of action-object pairs).*
- *$PA \subseteq PRMS \times ROLES$ is a many-to-many mapping permission-to-role assignment relation.*
- *$AUTH \subseteq PRMS \times USERS$ is a many-to-many permission-to-user authorization relation.*
- *The following condition (Core RBAC User Authorization) holds:*

$$(CRBAC\ C) \quad \forall p \forall u \forall r(UA(u,r) \wedge PA(p,r) \rightarrow AUTH(p,u))$$

3.2 Authorization Exceptions in Core RBAC

Previous work has focused on a single type of authorization exception in Core RBAC [2,13], specifically, "a particular user should not be authorized for a particular permission, such as reading a patient's record, irrespective of the role the user is assigned to", as illustrated in Example 1. We refer to this kind of authorization exception as *core exceptions for all roles*. However, for some access control policies, core exceptions for all roles provide insufficient granularity. For example:

Example 3. Assume that, in addition to roles "Doctor" and "Cardiologist", a medical records system has an additional role "Accident & Emergency Doctor", and Alice is being treated in the Emergency department. Though Alice may stipulate that "Tom does not have permission to access Alice's record", it is reasonable that Tom can access Alice's record when Tom is working in the Emergency department, i.e., when Tom is assigned to the role "Accident & Emergency Doctor".

Clearly, in the example above, it is inappropriate to disallow Tom to read Alice's record for all roles. Instead, an access control policy should stipulate an authorization exception for a user when assigned to a particular role. We refer to this kind of authorization exception as *core exceptions for one role*, and we refer to core exceptions for a role and for all roles collectively as *core exceptions*.

3.3 Expressing Core Exceptions in Core RBAC

Core exceptions can be expressed using a relation $CE_{p,u,r}(p,u,r)$, denoting that u assigned to r is not authorized for p. It may seem that *core exception for all roles* should be expressed as $CE_{p,u}(p,u)$ meaning u is not authorized for p in any role. However, logically speaking, $CE_{p,u}(p,u)$ can be derived from $CE_{p,u,r}(p,u,r)$, i.e., $CE_{p,u}(p,u) =_{def} \forall r\, CE_{p,u,r}(p,u,r)$. We therefore only need to add $CE_{p,u,r}(p,u,r)$ to Definition 2 to express core exceptions as below:

1. we add $CE_{p,u,r} \subseteq PRMS \times USERS \times ROLES$, an authorization exception relation.
2. we replace Core RBAC User Authorization condition (CRBAC C) with Core RBAC User Authorization with Core Exceptions:

$$(CRBAC\ C) \quad \forall p \forall u \forall r(UA(u,r) \wedge PA(p,r) \wedge \neg CE_{p,u,r}(p,u,r) \rightarrow AUTH(p,u))$$

We can now define Core RBAC with core exceptions as follows:

Definition 3. (Core RBAC with Core Exceptions)

- *USERS, ROLES, ACS, OBS, UA, PRMS, PA and AUTH are the same as in Definition 2.*
- $CE_{p,u,r} \subseteq PRMS \times USERS \times ROLES$ *is an authorization exception relation.*
- *The following condition (Core RBAC User Authorization with Core Exceptions) holds:*

$$(CRBAC\ C) \quad \forall p \forall u \forall r(UA(u,r) \wedge PA(p,r) \wedge \neg CE_{p,u,r}(p,u,r) \rightarrow AUTH(p,u))$$

Both examples 1 and 3 can be expressed in core RBAC with core exceptions, by stating core exceptions

$$CE_{p,u,r}(\text{'Read Alice's Record'}, Tom, \text{'Doctor'})$$
$$CE_{p,u,r}(\text{'Read Alice's Record'}, Tom, \text{'Cardiologist'})$$

The only difference between the two cases is that in example 3 we do not have the core exception

$$CE_{p,u,r}(\text{'Read Alice's Record'}, Tom, \text{'Accident \& Emergency Doctor'})$$

so if Tom is assigned to the accident and emergency doctor role, he will be authorized to read Alice's record.

4 Authorization Exceptions in Hierarchical RBAC

In the Hierarchical RBAC model, users assigned to a role have the permissions associated with the role, as well as permissions associated with all junior roles. In a hierarchical model, authorization exceptions may result not only from the role the user is assigned to, i.e., *core exceptions*, but also from role inheritance, i.e., *inheritance exceptions*.

In this section, we first briefly recall the Hierarchical RBAC model presented in [7] and discuss authorization exceptions in Hierarchical RBAC in detail. We then add some new constructs to Definition 3 to incorporate exceptions arising from role inheritance.

4.1 NIST Hierarchical RBAC

In NIST Hierarchical RBAC, roles are hierarchically organized into a role-subrole relationship called a role hierarchy. Based on the role hierarchy, role inheritance is interpreted using a graph where each node represents a role and a directed edge from role r_1 to r_2 indicates that role r_1 inherits the permissions associated with role r_2. Role inheritance is a partial order that is reflexive, transitive, and antisymmetric. Inheritance is reflexive because a role inherits its own permissions; transitive because permissions are inherited along the role hierarchy; and antisymmetry rules out cycles in the role hierarchy, that is, roles can not inherit from each other.

The NIST RBAC model distinguishes both general and limited role hierarchies. However, in the interests of generality, we consider only general role hierarchies below. NIST Hierarchical RBAC defined in [7] is introduced as follows:

Definition 4. (NIST RBAC with General Role Hierarchies)

- *$USERS$, $ROLES$, ACS, OBS, UA, $PRMS$, PA are the same as in Definition 1.*
- *$RH \subseteq ROLES \times ROLES$ is a partial order on ROLES called the inheritance relation. $RH(r_1, r_2)$ means r_1 inherits from r_2.*
- *The following condition holds: if $RH(r_1, r_2)$ then any user assigned to r_1 is a member of r_2 and every permission assigned to r_2 is assigned to r_1. More precisely:*

$$(HRBAC\,1) \quad \forall u \forall r_1 \forall r_2 (RH(r_1, r_2) \wedge UA(u, r_1) \rightarrow UA(u, r_2))$$

$$(HRBAC\,2) \quad \forall p \forall r_1 \forall r_2 (RH(r_1, r_2) \wedge PA(p, r_2) \rightarrow PA(p, r_1))$$

As in NIST Core RBAC, there is no relation explicitly connecting a user with a permission in the definition above. In order to express inheritance exceptions, we therefore modify the NIST RBAC with General Role Hierarchies definition by adding an additional condition (Hierarchical RBAC with User Authorization from Role Inheritance) as follows:

$$(HRBAC\,I) \quad \forall p \forall u \forall r_1 \forall r_2 (UA(u, r_1) \wedge RH(r_1, r_2) \wedge PA(p, r_2) \rightarrow AUTH(p, u))$$

Based on Definition 2 and Definition 4, our revised RBAC model with authorization both from a single role and role inheritance is as follows.

Definition 5. (Hierarchical RBAC with User Authorization Relation)

- *$USERS$, $ROLES$, ACS, OBS, UA, $PRMS$, PA and $AUTH$ are the same as in Definition 2.*
- *The following condition (Core RBAC with User Authorization) holds:*

$$(CRBAC\,C) \quad \forall p \forall u \forall r (UA(u, r) \wedge PA(p, r) \rightarrow AUTH(p, u))$$

- *$RH \subseteq ROLES \times ROLES$ is a partial order on ROLES called the inheritance relation. $RH(r_1, r_2)$ means r_1 inherits r_2.*
- *The following conditions hold: if $RH(r_1, r_2)$ then any user assigned to r_1 is a member of r_2 and every permission assigned to r_2 is assigned to r_1. More precisely:*

$$(HRBAC\,1) \quad \forall u \forall r_1 \forall r_2 (RH(r_1, r_2) \wedge UA(u, r_1) \rightarrow UA(u, r_2))$$

$$(HRBAC\,2) \quad \forall p \forall r_1 \forall r_2 (RH(r_1, r_2) \wedge PA(p, r_2) \rightarrow PA(p, r_1))$$

- *The following condition (Hierarchical RBAC with User Authorization from Role Inheritance) holds:*

$$(HRBAC\,I) \quad \forall p \forall u \forall r_1 \forall r_2 (UA(u, r_1) \wedge RH(r_1, r_2) \wedge PA(p, r_2) \rightarrow AUTH(p, u))$$

4.2 Authorization Exceptions in Hierarchical RBAC

It is easy to see that core exceptions may exist for each role in a Hierarchical RBAC model. In addition, a Hierarchical RBAC model may also have inheritance exceptions. In this section, we explore inheritance exceptions in detail.

Assume that role r_1 is a direct descendent of role r_2 in a role hierarchy, i.e., $RH(r_1, r_2)$,[3] $P_2 = \{p_1, p_2, \ldots, p_m\}$ is a set of permissions associated with r_2 and $U_1 = \{u_1, u_2, \ldots, u_n\}$ is a set of users assigned to r_1. In hierarchical RBAC, a user from U_1 has all permissions in P_2. However, in many cases, it is necessary to specify exceptions, for example, that no user from U_1 has permission $p_k \in P_2$. We refer to such exceptions as *direct inheritance exceptions for all users* since the exception applies to two directly related roles in a role hierarchy and affects all users assigned to the senior role (see example 2).

Inheritance exceptions have been extensively studied in the access control literature [15,8,11,6,13]. As far as we know, with the exception of [13], previous work on inheritance exceptions has focused on *direct inheritance exceptions for all users*. However, we believe that in some situations it can be useful to restrict inheritance exceptions between two directly related roles to a particular user. Consider the following example:

Example 4. Assume the role "Doctor" is a direct descendant of the role "Clinician" and hence *RH("Doctor", "Clinician")*. The permission "reading Alice's record" is associated with the role "Clinician" and George is assigned to the role "Doctor". Alice stipulates that George (who is a friend of Alice) should not be permitted to read Alice's record, while other users assigned to the role "Doctor" can.

We refer to such inheritance exceptions as *direct inheritance exceptions for one user*, as the exception applies to two directly related roles in a role hierarchy and affects only a particular user assigned to the senior role.

In a role hierarchy, an inheritance exception may apply not only between a role and its direct descendant, but also between a role and one of its indirect descendants. For example:

Example 5. Assume "Nurse in emergency department" is a direct descendant of "Nurse" and "Nurse" is a direct descendant of "Clinician", hence "Nurse in emergency department" is an indirect descendant of "Clinician". It may be regulated that users assigned to the role "Clinician" are authorized for a permission p like "updating patient's record" and users assigned to the role "Nurse in emergency department" can not be authorized for p while users assigned to the role "Nurse" can.

We can formulate the meaning of such inheritance exceptions as follows. Assume R is a set of roles, $r_1, \ldots, r_i, \ldots, r_j, \ldots, r_k, \ldots, r_n$ is a sequence of directly related roles in a role hierarchy defined by a partial order RH over R, and P_k is a set of permissions associated with r_k. By default, users assigned to r_i are also authorized for P_k. However, assume that a particular permission $p \in P_k$ should not be inherited by r_i. We refer to such inheritance exceptions as *indirect inheritance exceptions*, as they apply to two indirectly related roles. As in the case of direct inheritance exceptions, we distinguish two types of

[3] r_1 is a direct descendent of r_2 if there exists no role r_3 ($r_3 \neq r_1$, $r_3 \neq r_2$) in the role hierarchy such that $RH(r_1, r_3)$ and $RH(r_3, r_2)$.

indirect inheritance exceptions: *indirect inheritance exceptions for all users* and *indirect inheritance exceptions for one user*. As with direct inheritance exceptions for a user, inheritance exceptions between two indirectly related roles have not been studied in the literature.[4]

4.3 Expressing Authorization Exceptions in Hierarchical RBAC

We observe that inheritance exceptions, either for all users or a particular user, can be denoted by a relation $IE_{p,u,r}(p,u,r)$ regardless of whether they are indirect or direct. It may seem that since inheritance happens between two roles, inheritance exception should be expressed as $IE_{p,u,r,r'}(p,u,r,r')$, which means u assigned to r is not authorized for a permission p which is assigned to r'. However, this would stop u being authorized for p in the direct case, but not in the indirect case, since u's role r will inherit p from the roles between r and r' in the role hierarchy. We could specify exceptions $IE_{p,u,r,r'}(p,u,r,r'')$ for all such r'', but in situations where many roles are involved, it is undesirable to have to explicitly state this for each role which inherits from r'. Thus, we simply use $IE_{p,u,r}$ to prevent a user assigned to r from inheriting p from any roles. $IE_{p,r}(p,r)$ denoting that any user assigned to r is not authorized for p is just a special case of $IE_{p,u,r}(p,u,r)$. Logically speaking, $IE_{p,r}(p,r) =_{def} \forall u IE_{p,u,r}(p,u,r)$.

Further, we adopt a single authorization exception relation $EXP_{p,u,r}(p,u,r)$ to replace both $IE_{p,u,r}(p,u,r)$ and $CE_{p,u,r}(p,u,r)$. Otherwise in order to prevent a user u from being authorized for p we may have to state both $IE_{p,u,r}(p,u,r)$ and $CE_{p,u,r}(p,u,r)$. For example, assume that $UA(u,r)$ and $PA(p,r)$, and we wish to prevent u from being authorized for p. If we state $CE_{p,u,r}(p,u,r)$, u will not be authorized for p by the Core RBAC User Authorization with Core Exception condition. However, the model would need to have a condition for authorization which stems from role inheritance, along the lines of

$$\forall p \forall r_1 \forall r_2 (UA(u,r_1) \wedge RH(r_1,r_2) \wedge PA(p,r_2) \wedge \neg IE_{p,u,r}(p,u,r) \rightarrow AUTH(p,u))$$

In our example, if we do not state explicitly that $IE_{p,u,r}(p,u,r)$ also holds, we get

$$UA(u,r) \wedge RH(r,r) \wedge PA(p,r) \wedge \neg IE_{p,u,r}(p,u,r) \rightarrow AUTH(p,u)$$

and since the antecedent holds by the reflexivity of RH, $AUTH(p,u)$ will be derived.

In order to express inheritance exceptions, we believe it is necessary to abandon condition HRBAC 1 of Definition 5.[5] The reason why HRBAC 1 is problematic is as follows.

Assume R is a set of roles, $r_1,\ldots,r_i, \ldots, r_n$ is a sequence of directly related roles such that $RH(r_i, r_{i+1})$, r_n has permission p, and hence all roles from r_1 to r_n also have

[4] An exception is [13], in which authors discuss "nesting access policy statements", which is just a different expression of inheritance exception between two indirectly related roles.

[5] Note that although the NIST RBAC model adopts the most widely used definition of role hierarchy, many researchers agree that alternative interpretations may be appropriate in particular circumstances [14]. For example, [10] argues convincingly that only one of HRBAC 1 or HRBAC 2 should be used to define role inheritance.

p by HRBAC 2. Assume that u assigned to r_i should not be authorized for p. Such an exception could be expressed as $EXP_{p,u,r}(p, u, r_i)$. However, by (HRBAC 1), u could still get the permission from the roles r_{i+1} to r_n because u is implicitly assigned to $r_{i+1} \ldots, r_n$ by HRBAC 1. An alternative approach would be to retain condition HRBAC 1, and instead add exception statements $EXP(p, u, r')$ for all roles r' such that $PA(p, r')$ and $RH(r, r')$ (in other words, all roles r' which have permission p and to which u is implicitly assigned by HRBAC 1); however this may involve adding a large number of additional exceptions.

We now give our full definition of Hierarchical RBAC with authorization exceptions including core exceptions and inheritance exceptions.[6]

Definition 6. (Hierarchical RBAC with Unified Authorization Exceptions)

- *$USERS$, $ROLES$, ACS, OBS, UA, $PRMS$, PA and $AUTH$ are the same as in Definition 2.*
- *$RH \subseteq ROLES \times ROLES$ is a partial order on $ROLES$ called the inheritance relation. $RH(r_1, r_2)$ means r_1 inherits r_2.*
- *The following condition holds: if $RH(r_1, r_2)$ then every permission assigned to r_2 is assigned to r_1. More precisely:*

$$(HRBAC) \quad \forall p \forall r_1 \forall r_2 (RH(r_1, r_2) \wedge PA(p, r_2) \rightarrow PA(p, r_1))$$

- *$EXP_{p,u,r} \subseteq PRMS \times USER \times ROLES$.*
- *The following condition (Hierarchical RBAC with Unified Authorization Exceptions) holds:*

$$(HRBAC\,EXP) \quad \forall p \forall u \forall r (UA(u, r) \wedge PA(p, r) \wedge \neg EXP_{p,u,r}(p, u, r) \rightarrow AUTH(p, u))$$

5 Expressing HRBAC with Exceptions in DATALOG

In this section we show how Hierarchical RBAC with Unified Authorization Exceptions can be expressed as a stratified program of recursive DATALOG with negation. In fact, various extensions of DATALOG have been widely adopted in access control field because of their easy-to-read syntax and precise semantics [9]. For example, in [4], Bertino et al. show how to express access control models in D-DATALOG program. Formulating our model in DATALOG immediately gives us a simple and efficient algorithm for computing authorization decisions because DATALOG already has efficient query evaluation algorithm [1].

We first briefly introduce recursive DATALOG with negation programs and stratified programs from [1]. Then we show how to express Hierarchical RBAC with Unified Authorization Exceptions as a stratified program.

[6] For simplicity, we leave a single reflexive and transitive inheritance relation *RH* in our definition, but in our DATALOG implementation, we use a direct inheritance relation *DRH* instead and define *RH* as its reflexive transitive closure, again as suggested in [10].

5.1 Backgroud

Definition 7. *A* recursive DATALOG with negation program *is a finite set of* rules *of the form*

$$R_1(u_1) \leftarrow R_2(u_2), \ldots, R_n(u_n)$$

where

- *An* atom *is a n-ary-predicate with n terms. A* literal *is an atom or negated atom.*
- $n \geq 1$, $R_1, \ldots, R_n$ *are literal names and* $u_1, \ldots, u_n$ *are free tuples of appropriate arities.*
- *Each variable occurring in* u_1 *must occur in at least one of* $u_2, \ldots, u_n$.
- $R_1(u_1)$ *is called the* head *of the rule and* $R_2(u_2), \ldots, R_n(u_n)$ *forms the* body.
- *A rule without a body is called a* fact.

Next we introduce the concept of a *stratified* program. Unlike arbitrary recursive DATALOG with negation programs, stratified programs have a well-behaved semantics (a unique minimal model where all the consequences the program are true). Let P be a DATALOG program. A predicate appearing only in the body of a rule is referred to as an *extensional* predicate, while an *intentional* predicate is a predicate occurring in the head of a rule. The *extensional schema*, referred to as $edb(P)$, consists of the set of all extensional predicate names, whereas *intensional schema*, denoted as $idb(P)$, consists of all the intensional ones. The union of $edb(P)$ and $idb(P)$ is called the *schema* of P which is denoted as $sch(P)$. The *semantics* of a DATALOG program is a mapping from database instances over $edb(P)$ to database instances over $idb(P)$.

Now consider a program P in which *idb* predicates are defined by one or more rules of P and negation applies to predicates, such as R, appearing both in the body and the head of a rule, i.e., $R \in idb(P)$. Then program P could be considered as consisting of several parts. Specifically, for each *idb* predicate R', if the part of P defining R' comes before the negation of R' is used, we can simply compute R' before its negation must be evaluated. Such a way of treating P is called a *stratification* of P and is precisely defined as follows.

Definition 8. *A stratification of a recursive* DATALOG *with negation program* P *is a sequence of* DATALOG *with negation programs* $P^1, \ldots, P^n$ *such that for some mapping* σ *from* $idb(P)$ *to* $[1..n]$.

- $\{P^1, \ldots, P^n\}$ *is a partition of* P.
- *For each predicate* R, *all the rules in* P *defining* R *are in* $P^{\sigma(R)}$ *(i.e., in the same program of the partition)*
- *If* $R(u) \leftarrow \ldots R'(v) \ldots$ *is a rule in* P, *and* R' *is an idb predicate, then* $\sigma(R') \leq \sigma(R)$.
- *If* $R(u) \leftarrow \ldots \neg R'(v) \ldots$ *is a rule in* P, *and* R' *is an idb predicate, then* $\sigma(R') < \sigma(R)$.

Given a stratification $P^1, \ldots, P^n$ of P, each P^i is called a *stratum* of the stratification, and σ is called the *stratification mapping*. Not all programs are stratifiable [1]. However it is straightforward to determine whether a program is stratifiable. Specifically, let P be a DATALOG with negation program. The *precedence graph* G_P of P is the labeled graph whose nodes are the *idb* relations of P. Its edges are the following:

- if $R(u) \leftarrow \ldots R'(v) \ldots$ is a rule in P, then $< R', R >$ is an edge in G_P with label + (called a positive edge).
- if $R(u) \leftarrow \ldots \neg R'(v) \ldots$ is a rule in P, then $< R', R >$ is an edge in G_P with label - (called a negative edge).

Proposition 1. *A recursive* DATALOG *with negation program P is stratifiable iff its precedence graph G_P has no cycle containing a negative edge.*

A proof is given in [1, p.380].

5.2 Result

To express Definition 6 in DATALOG with negation we need the following *edb* predicates: UA for user-role assignment, DPA (for direct assignment of permissions to roles), DRH (for direct inheritance relation), EXP (for exceptions), which will be used to state the facts concerning user assignment etc.[7] We also need *idb* predicates RH, PA and $AUTH$ defined as follows:

$$
\begin{array}{ll}
r_1 & RH(r_1, r_2) \leftarrow DRH(r_1, r_2) \\
r_2 & RH(r_1, r_2) \leftarrow DRH(r_1, r_3), RH(r_3, r_2) \\
r_3 & PA(a, o, r) \leftarrow DPA(a, o, r) \\
r_4 & PA(a, o, r_1) \leftarrow DPA(a, o, r_2), RH(r_1, r_2) \\
r_5 & AUTH(a, o, u) \leftarrow PA(a, o, r), UA(u, r), \neg EXP(a, o, u, r)
\end{array}
$$

Now, we can easily prove that the program implementing Hierarchical RBAC with exceptions is stratifiable.

Proposition 2. *The program implementing Hierarchical RBAC with exceptions is stratifiable.*

Proof. It is easy to see that there is no cycle containing a negative edge in the precedence graph. The only negative edge is from EXP to $AUTH$, and there is no edge from $AUTH$ for any predicate.

6 Case Study

We illustrate our approach using a simple medical informatics case study. The permissions are taken from the appendix of [5], which defines a vocabulary for permissions in healthcare information systems. We used the DATALOG Educational System (DES), a free PROLOG-based implementation of a basic deductive database system.[8]

In our case study, role "nurse in emergency department" inherits from role "nurse" and role "nurse" inherits from role "clinician". In the DATALOG program, the direct inheritance relation is encoded as follows:

[7] We use pairs (action, object) to stand for permissions, and omit a permission relation $P(a, o)$ for brevity.

[8] For more details of DES, see `http://www.fdi.ucm.es/profesor/fernan/DES/index.html`

```
drh(nurse,clinician).
drh(nurse_in_emergency_department,nurse).
```

We assume three users, Jessica who is a nurse in the Emergency Department, and Kate and Ellen who are nurses. This corresponds to the following user-role assignment:

```
ua(jessica,nurse_in_emergency_department).
ua(kate,nurse).
ua(ellen,nurse).
```

The direct role-permission assignment is given below. Recall that the first two arguments constitute a permission, namely an action and an object (patient in this case), and the third argument is a role:

```
dpa(read_patient_test_report,alice,clinician).
dpa(read_patient_test_report,sherry,clinician).
dpa(read_patient_test_report,mina,clinician).
dpa(read_patient_test_report,katherine,clinician).

dpa(sign_history_and_physical,alice,clinician).
dpa(sign_history_and_physical,sherry,clinician).
dpa(sign_history_and_physical,mina,clinician).
dpa(sign_history_and_physical,katherine,clinician).

dpa(create_history_and_physical,alice,clinician).
dpa(create_history_and_physical,sherry,clinician).
dpa(create_history_and_physical,mina,clinician).
dpa(create_history_and_physical,katherine,clinician).

dpa(update_progress_note,alice,nurse).
dpa(update_progress_note,sherry,nurse).
dpa(update_progress_note,mina,nurse).
dpa(update_progress_note,katherine,nurse).

dpa(append_progress_note,alice,nurse_in_emergency_department).
dpa(append_progress_note,sherry,nurse_in_emergency_department).
dpa(append_progress_note,mina,nurse_in_emergency_department).
dpa(append_progress_note,katherine,nurse_in_emergency_department).
```

We also assume the existence of the following authorization exceptions: users assigned to the role of nurse should not be authorized to sign a patient's history and physical. In addition, Kate should not be authorized to read patient test report of Alice. In DATALOG:

```
exp(sign_history_and_physical,alice,kate,nurse).
exp(sign_history_and_physical,sherry,kate,nurse).
exp(sign_history_and_physical,mina,kate,nurse).
exp(sign_history_and_physical,katherine,kate,nurse).

exp(sign_history_and_physical,alice,ellen,nurse).
exp(sign_history_and_physical,sherry,ellen,nurse).
exp(sign_history_and_physical,mina,ellen,nurse).
```

```
exp(sign_history_and_physical,katherine,ellen,nurse).

exp(read_patient_test_report,alice,kate,nurse).
```

The remaining predicates are defined as in the previous section:[9]

```
rh(R1,R2) :- drh(R1,R2).
rh(R1,R2) :- drh(R1,R3), rh(R3,R2).
pa(A,O,R1) :- dpa(A,O,R1).
pa(A,O,R1) :- dpa(A,O,R2), rh(R1,R2).
auth(A,O,U) :- pa(A,O,R), ua(R,U), not(exp(A,O,U,R)).
```

DATALOG computes 43 tuples for the `auth` predicate, 48 tuples for the `pa` predicate and 3 for the `rh` predicate. Note that even in this small example, the advantage of using rules to compute the role-permission relation rather than storing it explicitly is obvious: the size of the program is much smaller and it is easier to maintain (integrity is ensured by the rules). Note that if we did not use exceptions, we would need to eliminate the inheritance between Clinician and Nurse and introduce a special role for Kate which has fewer permissions than Nurse. In the worst case, when every role, and every user assigned to a role, is involved in some exception, Hierarchical RBAC would collapse to a non-hierarchical model, losing the advantages of a compact and easy to maintain formalization.

7 Related Work

Exceptions, or preventing particular users from performing actions which their role would normally authorize them to perform, have been extensively studied in the literature, see for example [15,2,3,13]. Bacon et.al. address what we call *core exception for all roles* in [3] and propose supplementing an RBAC policy with an exception list in the OASIS model (which is based on NIST RBAC model). Our approach is similar to

[9] The implementation given above is not very efficient — we used it as an illustration as it closely corresponds to the model. For a more afficient and compact representation, we could introduce a predicate `dpa1` which means 'permitted to perform an action' (omitting the object) if a role is given a permission to perform a given action on all possible objects, for example `dpa1(read_patient_test_report, clinician)`. Similarly, we could introduce a predicate `exp1` to say that a user is not authorized to perform a given action on any object, for example `exp1(sign_history_and_physical, kate, nurse)`. We would also need two versions of `pa`, one of the form `pa(Action, Object, Role)` and another `pa1(Action, Role)`, with the obvious definitions. Finally, we need two definitions of `auth`:

```
auth(A,O,U) :-
    pa(A,O,R), ua(U,R), not(exp(A,O,U,R)), not(exp1(A,U,R)).
auth(A,O,U) :-
    pa1(A,R), ua(U,R), p(A,O),
    not(exp(A,O,U,R)), not(exp1(A,U,R)).
```

where `p` is a permission predicate.

theirs in that we adopt an exception table (EXP predicate). They do not deal with the hierarchical exceptions.

In [15], Sandhu et.al. propose extending the RBAC96 model with "private roles" which allow *direct inheritance exceptions for all users*. Specifically, assume role r_1 inherits permissions from role r_2 which is associated with a set of permissions P including p. To let r_1 inherit only the set $P \setminus \{p\}$, a private role r_2' which has permission p can be introduced. r_1 inherits permissions from r_2, which is now assigned P $\setminus\{p\}$. In this way, *direct inheritance exception for all users* is solved. However, *indirect inheritance exception* can not be handled by simply adding a "private role".

In [15], Sandhu et.al. also mention that, in some systems, certain permissions are blocked to be inherited from any roles. A typical example is a RBAC96-based model with "oriented" permissions proposed by Crampton in [6]. Namely, a permission p assigned to role r can be barred from being inherited by any role which is above r in the hierarchy. It is easy to see that *direct inheritance exception for all users* could be solved in such scheme, because no user of the role r' which is a direct descendant of r will be authorized for p. However, *indirect inheritance exception* and *direct inheritance exception for one user* can not be handled this way. Indirect inheritance exception for some indirect descendant r'' of r is not handled properly because although p is not inherited by r'', it is also not inherited by *any* roles r between r'' and r in the role hierarchy. Similarly, we cannot use this mechanism to bar just one user u of a more senior role from being authorized for p (unless we create a private role just for u).

Reid et.al. [13] propose a modified NIST RBAC model to express so-called "nesting access policy statements" which are equivalent to what we call *indirect inheritance exceptions*. They present an authorization algorithm for the modified model. Their scheme can handle several types of authorization exceptions. However, *core exceptions* can not be expressed in their approach. Our approach, on the other hand, handles all types of authorization exceptions, and arguably has a simpler way of computing authorization decisions.

8 Conclusion and Future Work

Though authorization exceptions, which are of practical importance in medical informatics, have been discussed in the literature, to the best of our knowledge there has been no proposal for a uniform treatment of different types of exceptions. To address this problem, we propose a systematic classification of user authorization exceptions. We incorporate these exceptions in the RBAC model and show how to express them in DATALOG with negation.

In future work, we plan to extend our work to dynamic access control (incorporating temporal constraints). We also believe that exceptions will be useful in role mining (extracting roles from Access Control Lists) since this process usually produces too many too specifically defined roles instead of more "natural" roles precisely due to the existence of a small number of exceptions for each natural role. We plan to look at modifying role mining procedures to produce roles with specified sets of exceptions.

Acknowledgement. We would like to thank Jason Crampton for his suggestions and comments on earlier version of this article, Bernd Blobel for directing us to the documents on RBAC in medical informatics.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (November 1994)
2. Bacon, J., Lloyd, M., Moody, K.: Translating role-based access control policy within context. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 107–119. Springer, Heidelberg (2001)
3. Bacon, J., Moody, K., Yao, W.: A model of OASIS role-based access control and its support for active security. ACM Transactions on Information and System Security 5(4), 492–540 (2002)
4. Bertino, E., Catania, B., Ferrari, E., Perlasca, P.: A logical framework for reasoning about access control models. ACM Transactions on Information and System Security 6(1), 71–127 (2003)
5. HL7 Security Technical Committee. Role Based Access Control (RBAC) Healthcare Permission Catalog. HL7 Security Technical Committee (January 2010)
6. Crampton, J.: On permissions, inheritance and role hierarchies. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 85–92 (2003)
7. Ferraiolo, D.F., Sandhu, R.S., Gavrila, S.I., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security 4, 224–274 (2001)
8. Goh, C., Baldwin, A.: Towards a more complete model of role. In: ACM Workshop on Role-Based Access Control, pp. 55–62 (1998)
9. Halpern, J.Y., Weissman, V.: Using first-order logic to reason about policies. ACM Transactions on Information and System Security 11, 21:1–21:41 (2008)
10. Li, N., Byun, J.-W., Bertino, E.: A critique of the ANSI standard on role-based access control. IEEE Security & Privacy 5(6), 41–49 (2007)
11. Moffett, J.D., Lupu, E.: The uses of role hierarchies in access control. In: ACM Workshop on Role-Based Access Control, pp. 153–160 (1999)
12. Power, D.J., Slaymaker, M., Simpson, A.C.: On formalizing and normalizing role-based access control systems. Computer Journal 52(3), 305–325 (2009)
13. Reid, J., Cheong, I., Henricksen, M., Smith, J.: A novel use of RBAC to protect privacy in distributed health care information systems. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 403–415. Springer, Heidelberg (2003)
14. Sandhu, R., Bellare, M., Ganesan, R.: Password-enabled PKI: Virtual smart cards versus virtual soft tokens. In: PKI Research Workshop (April 2002)
15. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)

Efficient Attack Detection Based on a Compressed Model*

Shichao Jin[1], Okhee Kim[1], and Tieming Chen[2]

[1] School of Software and Microelectronics, Peking University, Beijing, China
[2] School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China
{shichaojin.cs,anniekim.pku}@gmail.com, tmchen@zjut.edu.cn

Abstract. In order to achieve the goal of high efficiency in intrusion detection systems, especially in the real-time attack detection environment, a compressed model is proposed in this paper. With the emergence of the new clustering methods, such as the affinity propagation, the idea of the compressed detection model tends to be mature as it is unnecessary to define the number of centers beforehand. The compressed model resulting from both the horizontal compression and the vertical compression is built with representative training data and useful attributes in each package. In addition, a distance matrix is extracted from previous steps for processing complex data. Experimental study based on two publicly available datasets presents that the compressed model proposed can effectively speed up the detection procedure (up to 184 times) and most importantly, a minimal accuracy difference is guaranteed as well (less than 1% on average).

Keywords: Intrusion Detection, Affinity Propagation, Clustering, Model.

1 Introduction

According to the latest survey [4] in the Internet World Stats, Internet users have occupied 34.3% of the total global population. Due to the ubiquity of Internet, applications based on it have been an indispensable tool for searching various information and social interaction, such as Google and Facebook respectively. With the large volume of information streams, the design of Intrusion Detection System (IDS) specialized for the high efficiency in networks has become much more urgent as the times require for dealing with the unceasing attacks from the Internet.

Compared with the traditional way of signature-based detection, anomaly detection has enjoyed great popularity in the academic circles, as it has the potential to detect unknown attacks. Hence, anomaly detection is our major concern in this paper. Since our research focuses on speeding up the detection process based on a compressed model, our research is applicable for those model-based detection approaches. For example, according to several categories of anomaly detection divided

* Paper was partially supported by the National Natural Science Foundation of China under grant No.61103044, Zhejiang Natural Science Foundation of China under grant No.Y1110576.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 248–262, 2013.

by [2], classification based anomaly detection needs to build a model before detection, where our idea can be easily transplanted.

Being an important factor in detection process, real time must be taken into consideration regarding the practical meaning in the real environment. In this research, in order to boost detection, we have paid our attention mainly to the building model process instead of the detection phase. Our proposal is made through inspecting into the following common natures of the training data:

1. By analyzing the attributes of the training data, we can easily find that the values of some attributes (features) in the whole training data only range in a small scale, which may have less impact on the detection accuracy.
2. Some training instances are similar, because they are only different from each other on several attributes and the values of these attributes are slightly different.
3. For high dimensional training data, computing the similarity of each pair of them is time-consuming.

Training data are usually in a large scale, which can severely impede the detection since many detection methods may need to scan all of them in certain cases. Apparently, an effective and direct way to reduce time cost for detection is to minimize the volume of a model that is used in the detection process. For the purpose of effectively and efficiently handling these problems, we propose a compressed detection model lying on the observations above from training data, and the model compression includes horizontal compression, vertical compression and distance matrix extraction. The overall idea is presented in Fig. 1, where the first step is to normalize the original data followed by the horizontal compression and the vertical compression sequentially. In addition to the steps of compressing training data, a distance matrix will also be contained in the model to further improve the detection performance.

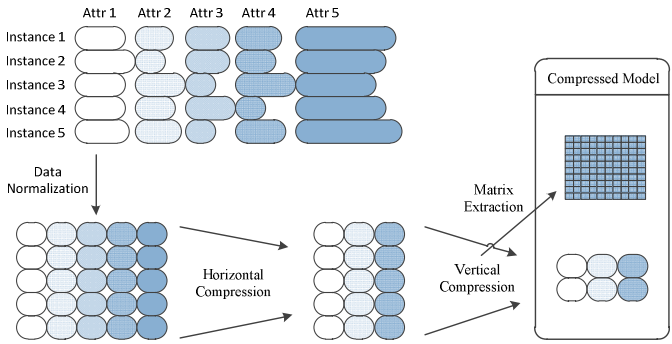


Fig. 1. The main idea of model compression

In this paper, we make the following contributions:

1. We propose a compressed detection model, which is a compact version of the original model with regard to dataset volume, and the detection speed can be improved significantly during the detection stage consequently.

2. We evaluate our approach on two publicly available test beds (CDMC2012 [9] and KDD99 [8]) and demonstrate both the accuracy difference and the speed of our method in the attack detection process.

The remainder of this paper is organized as follows. Section 2 explains the recent related work and in section 3 we describe the methodology of our compressed model in detail. Experimental performance and corresponding analysis are presented in Section 4. Concluding remarks follow in Section 5.

2 Related Work

These decades, researchers in related fields have paid much attention to the intrusion detection. Signature based detection (e.g. Snort [10]) relies on the knowledge of system vulnerabilities and known attack patterns. Thus, it is unable to detect unknown attacks. Correspondingly, anomaly detection is more dynamic and able to detect novel and unknown attacks. As a result, anomaly detection has attracted a lot of attention from researchers.

In 1998, Lee and Stolfo [1] published a data mining approach for the intrusion detection, where they proposed architecture (or a framework) for the agent-based intrusion detection, and deployed data mining methods to extract detection rules. Afterwards many researchers also focused on the way of boosting the detection speed. Sung and Mukkamala [5] improved the detection speed by extracting the useful subset of attributes with ANN and SVM. And Srilatha et al. [6] investigated the performance of Bayesian networks (BN) and Classification and Regression Trees (CART) to build lightweight IDS. In network anomaly detection, the random data sampling [11] was employed to deal with the massive traffic, but it potentially lost useful data. Li et al. [12] used the Fuzzy C-Means for the purpose of selecting smaller number of training data. Besides, with the development of the hardware, Giorgos et al. [13] made efforts to improve the detection speed by applying graphics processors. According to the nature of input data, anomalies can be classified into point, contextual and collective categories [2], where they can be dealt with a specific database algorithm.

In this paper, we attempt to speed up the attack detection through the phase of model construction. We endeavor to compress training data both horizontally and vertically, and extract the distance matrix as well for future use. The details about our compressed model will be specified in the next section.

3 Compressed Model

In this chapter, we will elaborate the details of our proposed detection model. Conventionally, data collection, data analysis, model building and detection are four sequential steps for constructing the supervised or the semi-supervised anomaly detection system, where the model building step is our major concentration. To build a compressed model, firstly we horizontally abstract the useful attributes from the training data that have been normalized. Then, we vertically compress the training data to

further pick up representative ones. Besides, during the vertical compression phase, we also extract a distance matrix of training data for future detection use.

Prior to the discussion of the compressed model, we firstly introduce the format of the dataset. The whole training and testing data are made up of a number of instances. Each instance can be seen as a row, which consists of several attributes, and can be expressed as $X_i = \{x_1, x_2, \ldots, x_j, \ldots, x_m\} (1 \leq i \leq n, 1 \leq j \leq m)$, where X_i is the i-th instance of the whole dataset X. One of the dataset used in our experiments is the KDD99 dataset, which is formed through extracting 41 attributes for each network package from DARPA1998 [15] by Lee et al. [1]. A mapping example of the attributes and their corresponding values in KDD99 dataset is listed in Table 1 and a similar mapping method is also employed in another experimental dataset, CDMC2012.

Table 1. A mapping example of attributes and the corresponding values for KDD99

Basic Attributes		*Content Attributes*		*Traffic Attributes*	
Name	Value	Name	Value	Name	Value
duration	[0, 58329]	hot	[0, 101]	count	[0, 511]
protocol_type	{TCP, UDP, ICMP}	logged_in	{0, 1}	serror_rate	[0.00, 1.00]
src_bytes	[0, 1379963888]	root_shell	{0, 1}	same_srv_rate	[0.00, 1.00]
wrong_fragment	[0, 3]	num_root	[0, 7468]	dst_host_count	[0, 255]

3.1 Data Normalization

Since the training data are made up of a large number of instances and each instance has several attributes, a challenge about the training data is that the values of different attributes are distributed on disparate scales, which may cause a bias toward certain attributes over others. Here we give an example: consider two vectors with 3 attributes, {(0, 1200, 5), (1, 1000, 10)}. Taking the Euclidean distance for example, the squared distance between vectors will be $(0 - 1)^2 + (1200 - 1000)^2 + (5 - 10)^2$, which is decided mainly by the second attribute. To balance the contribution of every attribute in the similarity calculation, we first normalize the data to the scale of [0, 1] through formula (1):

$$X_i[j] = \frac{X_i[j] - X_{min}[j]}{X_{max}[j] - X_{min}[j]} . \quad (1)$$

$X_{min}[j]$ is the smallest value of attribute j among the dataset X, while $X_{max}[j]$ is the biggest one correspondingly.

3.2 Horizontal Compression

The horizontal compression, as the first step of building a compressed detection model, mainly explores the relations and correlations of the features within an instance, and extracts useful features. The horizontal compression is useful because in some complex classification fields the false correlative features may impede the process of

attack detection. Most importantly, some features may be redundant since they may not play a role to distinguish the instance from others. Hence, it is worth mentioning that excessive features may slow down the detection consequently. As we will introduce below, the clustering methods utilized in our experiments will calculate the distance between two instances to measure their similarity. Given the high dimensionality resulting from numerous attributes in each instance, it is time-consuming to calculate the distance with all the attributes one by one. For this reason, it is meaningful to horizontally compress the training data no matter in the model building phase or in the detection process.

There are many ways to realize our horizontal compression idea. Here we choose OneR [14] as it is easy to understand. We briefly introduce the principle of OneR by taking the *protocol_type* attribute of the KDD99 dataset as an example. The process of compressing the instances horizontally using OneR is shown in Fig. 2.

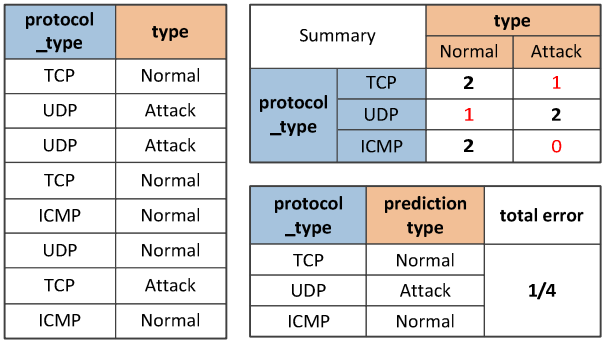


Fig. 2. An example of how OneR works

OneR extracts the rules by examining the attributes one by one. The attribute in Fig. 2 is *protocol_type*. First of all, OneR summarizes the quantitative relationship between the instance type (normal or attack) and the attribute value (TCP, UDP and ICMP). By traversing all the instances, it is concluded that there are two normal instances and one attack instance with the TCP *protocol_type* in Fig. 2. Then OneR chooses the instance type (normal or attack) with a larger frequency as the prediction type for a certain attribute value. For example, since two normal instances (one attack instance) have the TCP value, the prediction type of the TCP is 'normal', which means if the instance has the *protocol_type* value TCP, we predict that this instance is a 'normal' one. Finally, OneR calculates the total error of every attribute, and chooses the attributes with lower total error from all attributes as the representative ones.

In our experiments, we extract 10 features out of total number of 14 for CDMC2012 dataset. As to KDD99 dataset, we select 12 out of 34 numerical attributes to represent an instance.

3.3 Vertical Compression

Some training instances appear to be duplicate or similar, for example, they are probably the same packages and do the same business during a certain period of time. Thus, vertical compression is responsible for extracting a smaller set of representative training instances from a large scale of ones.

For this purpose, we here employ the recently published affinity propagation [3] approach, which is actually a cornerstone of our idea. The reason to use the affinity propagation method instead of other clustering methods, such as k-means, is straightforward, because the affinity propagation clusters the data without a predefined k, which is the number of clusters. In the case of the intrusion detection, we usually do not know how many clusters will be suitable for the training data. On the supposition that we have the knowledge about the appropriate number of clusters beforehand, the classical clustering method k-means, however, will also be taken in our experiments for a comparison purpose. In order to make the parameters settings in our experiments understandable, we would like to briefly introduce the affinity propagation and the k-means respectively.

Affinity Propagation. Affinity Propagation (AP) clusters instances by passing messages between data points iteratively. Define $X = \{X_1, X_2, \ldots, X_n\}$ as the instances to be clustered and let $d(X_i, X_j)$ denote the similarity or distance between instance i and instance j. Finally, we should minimize the sum of the distances between instances and their exemplars. The fitness function is listed below:

$$E(c) = \sum_{i=1}^{n} S\left(X_i, X_{c(i)}\right) . \tag{2}$$

Here $c(i)$ is the exemplar of instance i, and $S(X_i, X_j)$ is defined as below:

$$S(X_i, X_j) = \begin{cases} -d(X_i, X_j)^2 & if\ i \neq j \\ p & otherwise \end{cases} . \tag{3}$$

Here p stands for *preference* which is used to indicate how much an instance is likely to be chosen as an exemplar. Please note that an exemplar in AP is just like a representative instance of a cluster in k-means. The clustering procedure of AP seeks a good clustering result that can maximize the fitness function $E(c)$ by passing messages.

K-means. Given a set of instances $X = \{X_1, X_2, \ldots, X_n\}$, where each instance is a real attribute vector, k-means aims to partition the n instances into k sets $(k < n)$ $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares:

$$\sum_{i=1}^{k} \sum_{x_j \in s_i} \left\| X_j - \mu_i \right\|^2 . \tag{4}$$

Here μ_i is the mean point in S_i.

K-means firstly selects k instances randomly as cluster centroids, and then it assigns each of the remaining instances to the cluster whose centroid is the most similar

to this instance. After this, k-means refreshes all the clusters and makes the mean vector of the entire vectors within the cluster as the new centroid. K-means iteratively runs the procedure until the fitness function is convergent.

Compare AP with K-means. In summary, instead of requiring the number of clusters pre-specified in k-means, the preference in AP can affect the number of clusters to be generated as the instance with larger preference is more likely to be chosen as an exemplar.

3.4 Distance Matrix

In the process of vertical compression through the affinity propagation approach, the distances between each pair of instances are calculated and form a distance matrix as a result. These values in the matrix may be reused to further accelerate the detection.

With the nearest neighbor based detection, the new coming instance that may be normal or not will be compared with every instance stored in the model sequentially in order to find the most similar one if the brute force strategy, that is, the linear scan is employed. Otherwise, an index will be built in advance to enhance the search performance later. In both cases, computing the distance between two instances will occur. When the computational cost of the distance calculation is high, especially for high dimensional data and complex data types including graphs and strings as used in the collective anomaly detection [20], lessening the times of distance computations tends to be essential.

We here would like to briefly talk about the technique that helps to achieve the goal of reducing the computational cost, since it proves the usefulness of our distance matrix from a theoretical view. Often the objective is fulfilled by employing the triangle inequality if the distance function defined meets the requirement of metric space [19] including non-negative, identity of indiscernibles, symmetry and triangle inequality. In practice, the distance function does meet these rules usually, such as the Euclidean distance, the edit distance [17] for strings and the tree edit distance [18] for graphs.

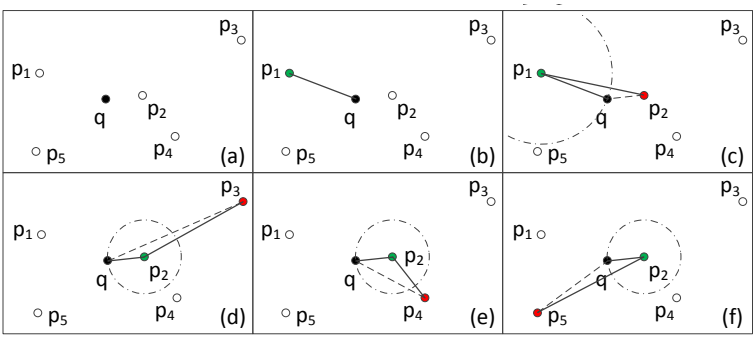


Fig. 3. Pruning strategy with the triangle inequality

We now take an example in a 2-dimensional view with the Euclidean distance for better understanding how to utilize the distance matrix. As shown in Fig. 3, q represents the new coming instance and other points are instances stored in the model. To find the most similar instance of q, all the distances between q and other points will be calculated sequentially if the linear scan is deployed. However if the distance matrix is provided, we firstly calculate $d(q,p_1)$ (see figure b) where d stands for the distance between the first and the second parameter. And next for p_2 (figure c), the minimum of $d(q,p_2)$ can be obtained from the triangle inequality $d(q,p_2) \geq |d(p_1,p_2) - d(q,p_1)|$. If $|d(p_1,p_2) - d(q,p_1)| \geq d(q,p_1)$ is satisfied, the calculation of the $d(q,p_2)$ can be omitted. Since the inequality is not satisfied, $d(q,p_2)$ is calculated and now p_2 is the closest point to q. As for p_3, since $d(q,p_3) \geq |d(p_2,p_3) - d(q,p_2)| \geq d(q,p_2)$, we do not need to calculate $d(q,p_3)$ anymore. And the rest can be done in the same manner that $d(q,p_4)$ still need to be calculated while the calculation of $d(q,p_5)$ can be saved.

In summary, pruning with the triangle inequality is the essence of our distance matrix. For a deeper investigation, one can refer to [16]. Please note that the affinity propagation algorithm is unnecessary to accommodate to the metric space according to its own explanation.

4 Experimental Evaluation

In this chapter, extensive experiments are conducted to demonstrate the effectiveness of our idea. We run all experiments on a PC with an AMD Phenom(tm) II N970 Quad-Core 2.20GHz CPU and 4GB of main memory. Please note that only one core of the CPU is used as our implementation does not support multi-thread currently and the heap size for Java is set to 1024MB.

4.1 Dataset Description

In order to enhance the credibility of our experimental results, we choose two publicly available data, KDD99 and CDMC2012 as our test objects.

KDD99 dataset is a well-known intrusion detection evaluation dataset transformed from DARPA Intrusion Detection Evaluation dataset. Although KDD99 has been criticized for various reasons, it is still a benchmark for evaluating performance of intrusion detection. KDD99 is collected in a military network environment and it contains numerous simulated connections including normal ones and attacks. Each of the connections in the KDD99 has already been broken down into 41 attributes and well labeled as normal or a specific attack type. Here we simply classify all the instances into two types, normal and attack. In addition, only numerical data are taken into consideration. As a result, 34 attributes are considered in our experiments. Due to the large volume of KDD99 and the limited size of the Java heap, we use randomly generated 8,000 instances from KDD99 for training and another 300,000 instances for detection.

Since KDD99 is an old dataset more than ten years ago which cannot reflect the current network situation to some extent, we also use the CDMC2012 dataset in our experiments. The real traffic data in CDMC2012 are collected from several types of honeypots and a mail server over 5 different networks inside and outside of Kyoto University. The dataset is composed of 14 features including label information which indicates whether each session is attack or not. Similarly, we classify the dataset into two categories, normal and attack, and we use the available training data to perform our experiments, where 5,000 instances are used for training and another 123,720 instances for detection.

4.2 Detection Approaches

After the compressed model is built, two traditional detection methods are adopted in our experiments to evaluate the resulting performance of the compressed model, namely KNN and SVM.

KNN (K Nearest Neighbor) is one of the most widely used classification methods in data mining. It finds k nearest neighbors of a given instance among all the training data. There exist various ways of realizing KNN algorithm, and for the purpose of comparing traditional KNN with our improved KNN by utilizing distance matrix, a linear scan is employed in our experiments.

SVM (Support Vector Machine) is one of the recognized machine learning methods for classification, regression and other learning tasks. Here we apply C-SVC (C–Support Vector Classification), one among the SVMs, to identify whether a package is abnormal or not. Unlike one-class SVM, which is a frequently used detection method in intrusion detection using the model built with normal class only, C-SVC is based on the model with both normal and abnormal instances.

4.3 Parameters Settings

Since parameters of clustering methods and detection approaches can significantly influence the results, here we will shed light on the way we choose them.

AP. In AP, the only parameter that should be set is the preference. In our experiments, for a comprehensive evaluation, we choose the preferences between the minimum and the maximum of the similarities to generate expected number of clusters that are separately distributed. The relationship between the number of exemplars generated with AP and the corresponding preference set is described in Table 2. Since the number of clusters generated by AP is decided by its preference, it is hardly possible to generate exactly the wanted number of exemplars. Thus, we endeavor to generate similar numbers of exemplars for the conditions with and without OneR respectively for a comparison purpose, but not exactly the same. Accordingly, it can be observed from Table 2 that the number of exemplars grows with the increment of the preference within expectation.

K-means. Since we use k-means here for the comparison purpose with AP, the parameter k in the k-means should be set the same with the number of exemplars generated by AP.

Table 2. The relationship between preference and the number of exemplars for CDMC2012 and KDD99 respectively in AP

(a) CDMC2012

With OneR		*Without OneR*	
Exemplar	Preference	Exemplar	Preference
69	-0.4384616	65	-0.8939194
120	-0.1488941	120	-0.2527850
212	-0.0434867	216	-0.0729020
326	-0.0160566	326	-0.0260614
562	-0.0041900	526	-0.0084119
689	-0.0023318	682	-0.0040380
876	-0.0011680	886	-0.0020661
1215	-0.0003477	1260	-0.0004942
1618	-0.0001059	1634	-0.0001566
2056	-0.0000116	2074	-0.0000412
2512	-0.0000006	2342	-0.0000060

(b) KDD99

With OneR		*Without OneR*	
Exemplar	Preference	Exemplar	Preference
205	-0.0482469	203	-0.4156618
324	-0.0170783	322	-0.1557253
480	-0.0072700	471	-0.0672369
625	-0.0036726	620	-0.0368273
885	-0.0015379	891	-0.0156519
1029	-0.0010654	1092	-0.0090514
1218	-0.0006846	1203	-0.0067543
1537	-0.0003841	1566	-0.0028883
2110	-0.0001585	2072	-0.0011145
3044	-0.0000385	3083	-0.0002925
4043	-0.0000077	4013	-0.0001038

KNN. As to KNN, we set the parameter k to be 1, which means that the type for each tested instance rests with its nearest neighbor in the compressed model. Although this approach is rather simple, it has shown the high effectiveness practically.

C-SVC. As for C-SVC, since the number of attributes is quite small compared with the amount of instances, we choose to deploy a nonlinear kernel, namely Radial Basis Function kernel (RBF kernel), to map data to higher dimensional spaces. To better use the RBF kernel, one should determine two parameters: C and γ. Since the test data are unknown in advance, we can only find proper parameters using foregone training data with the help of cross-validation. LibSVM [7] provides an automatic python code on "grid-searching" C and γ using cross-validation, through which the best parameters for our training data can be concluded as: for KDD99 dataset, the best $(C, \gamma) = (2048, 1.2207E-4)$ when OneR is deployed and $(C, \gamma) = (32768, 3.05176E-5)$ otherwise, and for CDMC2012 dataset $(C, \gamma) = (8192, 3.05176E-5)$ for both with and without OneR.

4.4 Experimental Results

Time Cost for Model Compression. As to horizontal compression, it takes 2.13 seconds to compress 308,000 instances from KDD99 and 0.516 seconds for 128,720 instances from CDMC2012 dataset. The time cost for generating the compressed model vertically with AP and k-means respectively is described in Table 3. It is can be concluded from Table 3 that model compression may take relatively long time. As to CDMC2012 dataset, the average time cost for compressing model vertically is 28.83 seconds. The records in the table give us a hint that the step of vertical compression can be done off-line to meet the requirement in the real-time environment, which applies to the horizontal compression as well. To be specific, we just need to

provide the original training data to the machine responsible for compressing the model, from which the generated compact model will be returned to the on-line part. From Table 3, we can also easily observe that the horizontal compression with OneR can save time on the vertical compression in most cases.

Table 3. Time cost for vertical compression for CDMC2012 and KDD99 respectively

(a) CDMC2012

With OneR			Without OneR		
Num	AP	KM	Num	AP	KM
69	38.175	2.745	65	83.539	3.479
120	54.400	3.276	120	41.403	5.336
212	39.703	5.726	216	46.520	5.600
326	36.739	6.552	326	35.429	7.394
562	46.598	11.295	526	48.565	11.850
689	37.254	13.525	682	50.748	13.354
876	33.619	15.866	886	31.358	21.262
1215	41.185	19.671	1260	43.432	26.208
1618	33.729	23.853	1634	41.543	31.761
2056	25.694	33.369	2074	30.734	33.462
2512	25.390	32.932	2342	30.460	43.742

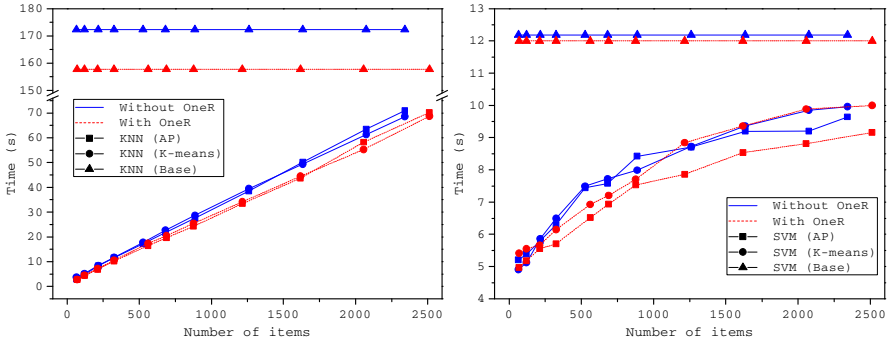
(b) KDD99

With OneR			Without OneR		
Num	AP	KM	Num	AP	KM
205	147.498	34.507	203	175.784	90.669
324	202.770	26.754	322	196.452	149.214
480	104.367	26.270	471	157.984	97.749
625	144.269	32.580	620	175.535	132.725
885	117.100	33.961	891	224.392	163.210
1029	93.866	16.241	1092	152.242	182.536
1218	97.402	56.903	1203	170.630	168.358
1537	108.379	63.698	1566	157.418	190.130
2110	92.181	75.722	2072	147.920	283.839
3044	94.564	98.608	3083	135.892	338.545
4043	95.957	147.188	4013	104.781	275.798

Detection Time. The acceleration of detection is our major contribution in this paper. The test results about it are shown in Fig. 4, 5 and 6 for both CDMC2012 and KDD99. Among them, Fig. 4 and Fig. 5 show the experimental results with regard to our horizontal and vertical compression, and Fig. 6 is mainly used to show the effectiveness of our distance matrix. Please note that baselines in Fig. 4 and Fig. 5, which do not employ the vertical compression but horizontal compression may be used, are drawn as well for a comparison purpose, while in Fig. 6 only AP is tested for compressing the model vertically, and linear KNN with and without our distance matrix is the detection method. By observing these three graphs, we can conclude that:

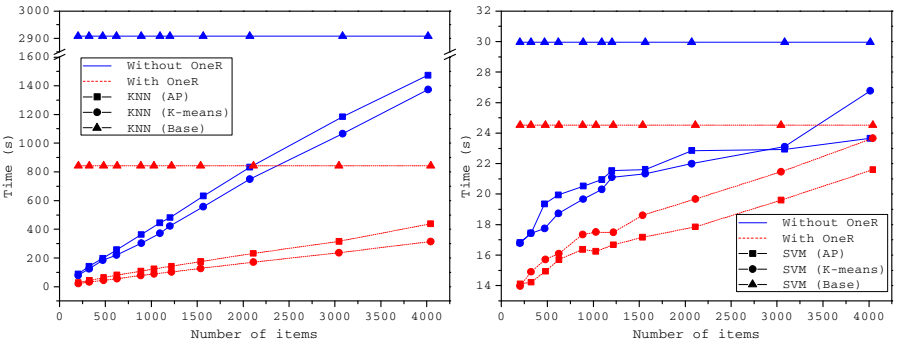
1. There is an obvious improvement of speed due to our horizontal compression when we compare every blue line with the dashed red line in each graph, where the maximal speed-up factor reaches to around 4 in KDD99 with KNN method.
2. It is reasonable to see from all of these figures that the smaller amount of clusters is, the shorter time it will take for detection for both KNN and SVM. To put it differently, one may tend to use smaller number of training instances to meet the need of real time if only the detection accuracy can be ensured.
3. In Fig. 6, comparing the line of linear KNN with the line of KNN with distance matrix, we can find that using our distance matrix can significantly improve the detection speed especially for the KDD99 dataset without the horizontal compression. One of the possible reasons is that since the original KDD99 dataset has more attributes, it will take a relatively long time to calculate the distance between instances, where our pruning strategy with distance matrix will play a more important role.

In summary, if we consider the baseline without the horizontal compression and the record with our full compressed model, the speedup reaches to around 132 times (172.363 versus 1.3) for CDMC2012 and almost 184 times (2907.775 versus 15.718) for KDD99 when KNN is used as the detection method.



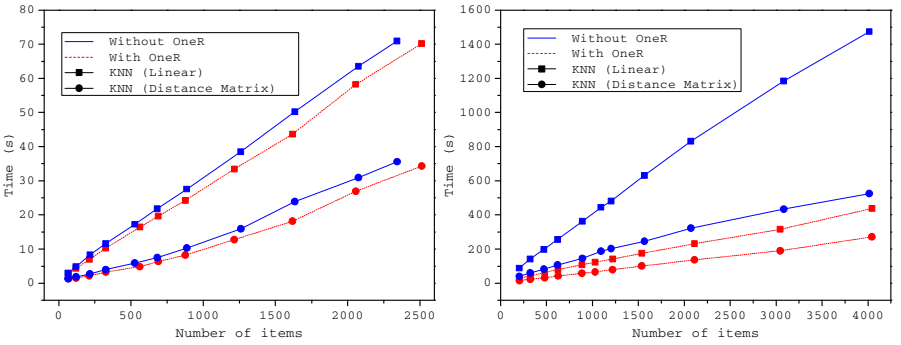
(a) KNN (b) SVM

Fig. 4. Comparative results of detection time with KNN and SVM respectively for CDMC2012



(a) KNN (b) SVM

Fig. 5. Comparative results of detection time with KNN and SVM respectively for KDD99



(a) CDMC2012 (b) KDD99

Fig. 6. Comparative results of detection time with and without distance matrix for CDMC2012 and KDD99 respectively

Accuracy Difference. Although the resulting efficiency that has been empirically proved above is our major focus in this paper, the detection accuracy is another important factor for a real detection system. However, we will concentrate on the difference of detection accuracy (with and without the compressed model) instead of the direct detection accuracy, because our compressed model is not proposed to contribute to the accuracy improvement, and the detection methods which are related to the direct detection accuracy are still the common ones. We would like to survey the difference with two standard measures here, namely recall and false positive rate. Recall is the ratio between the number of correctly detected anomalies and the total number of anomalies. False positive rate is the ratio between the number of data records from normal class that are misclassified as anomalies and the total number of data records from normal class. Table 4 and Table 5 record the recall and the false positive rate (not the relative difference) respectively for CDMC2012, and results of KDD99 are shown in Table 6 and Table 7. The last row of every table below is the benchmark without vertical compression.

Table 4. Recall with and without horizontal compression for CDMC2012

(a) With horizontal compression

R \ *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
69	97.1653	95.2388	97.9342	93.2254
120	96.7909	95.8339	97.0098	93.4226
212	96.2301	96.9319	96.9320	94.3126
326	95.9425	96.0365	95.0868	93.4243
562	95.6024	89.9765	94.2782	93.8766
689	95.6458	90.1013	94.3886	93.9074
876	95.8755	90.2768	94.4935	93.9707
1215	95.9280	77.8365	94.7739	94.0503
1618	95.9443	78.5709	94.7395	94.3614
2056	95.8430	62.8817	94.7467	94.3813
2512	95.8068	64.4736	94.5279	94.3849
5000	96.0980		94.3126	

(b) Without horizontal compression

R \ *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
65	95.7145	95.3075	96.5955	94.6834
120	95.7308	78.0481	97.0785	87.6411
216	94.9855	79.8101	95.8412	90.6983
326	95.0054	78.9110	94.3343	93.9743
526	94.9602	79.1842	94.5333	94.2873
682	95.0217	76.0836	94.8860	94.7902
886	95.1954	87.6972	94.8444	94.8318
1260	95.1592	90.2840	94.9367	94.7684
1634	95.1592	88.6849	95.0000	94.8082
2074	95.1429	62.7985	95.6078	94.9331
2342	95.0742	62.8636	95.1302	94.9729
5000	95.4016		96.5955	

Table 5. False positive rate with and without horizontal compression for CDMC2012

(a) With horizontal compression

FPR \ *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
69	2.1786	2.1581	2.3042	2.2881
120	2.2735	2.0894	2.3042	2.2808
212	2.1815	2.0617	2.3042	2.2589
326	2.0967	2.0295	2.2881	2.1800
562	1.9798	1.7928	2.2750	2.2443
689	1.9842	1.8162	2.2750	2.2443
876	1.9258	1.7840	2.2443	2.2443
1215	1.9535	1.5079	2.2794	2.2443
1618	1.8907	1.5108	2.2487	2.2443
2056	1.8717	1.2946	2.2443	2.2443
2512	1.8498	1.3004	2.2443	2.2443
5000	1.8279		2.2443	

(b) Without horizontal compression

FPR \ *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
65	1.1338	1.1163	2.2560	2.1625
120	1.1455	0.7715	2.3042	1.9740
216	1.1163	0.8358	2.2911	1.9959
326	1.1178	0.0801	2.2677	2.1376
526	1.1192	0.8022	2.2662	2.2691
682	1.1222	0.7905	2.2691	2.2721
886	1.1178	1.0243	2.2735	2.2428
1260	1.1178	1.0491	2.2428	2.2428
1634	1.1178	1.0184	2.2443	2.2428
2074	1.1178	0.5611	2.2458	2.2443
2342	1.1002	0.5625	2.2443	2.2443
5000	1.0929		2.2531	

Table 6. Recall with and without horizontal compression for KDD99

(a) With horizontal compression

R / *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
205	98.6538	98.9963	97.1988	97.2399
324	98.0899	99.1644	97.2186	97.1364
480	98.5229	99.2246	97.2194	97.1600
625	98.8182	99.1226	97.2148	97.1783
885	99.0708	99.0237	97.2109	97.1631
1029	98.8250	99.2344	97.1905	97.1783
1218	98.9400	99.2786	97.1889	97.1958
1537	99.1515	99.4003	97.1912	97.1943
2110	98.9970	99.4551	97.1927	97.1927
3044	99.3730	99.4909	97.1904	97.1889
4043	99.3577	99.5099	97.1889	97.1882
8000	99.5738		97.2255	

(b) Without horizontal compression

R / *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
203	99.4422	99.3577	97.5557	97.5329
322	99.5487	99.4513	97.9530	97.5291
471	99.6157	99.4331	97.5702	97.5420
620	99.6355	99.6872	98.9635	98.6645
891	99.6545	99.6850	98.2315	99.2558
1092	99.7085	99.6796	99.0160	99.0648
1203	99.7192	99.6926	99.1089	99.1972
1566	99.7177	99.7260	99.1850	99.1819
2072	99.7162	99.7245	99.2550	99.2421
3083	99.7177	99.7238	99.2306	99.2459
4013	99.7184	99.7207	99.2558	99.2383
8000	99.7215		99.2436	

Table 7. False positive rate with and without horizontal compression for KDD99

(a) With horizontal compression

FPR / *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
205	0.2699	0.7509	0.0196	0.0273
324	0.1857	0.5932	0.0219	0.0125
480	0.3138	0.6780	0.0225	0.0136
625	0.3073	0.5131	0.0225	0.0136
885	0.3779	0.4561	0.0225	0.0142
1029	0.2746	0.4146	0.0214	0.0196
1218	0.2533	0.3553	0.0219	0.0237
1537	0.3345	0.3262	0.0219	0.0231
2110	0.2906	0.4217	0.0225	0.0231
3044	0.2841	0.4336	0.0243	0.0255
4043	0.3001	0.4288	0.0255	0.0249
8000	0.3565		0.0326	

(b) Without horizontal compression

FPR / *Num*	*KNN*		*SVM*	
	AP	K-means	AP	K-means
203	0.0552	0.0908	0.0338	0.0486
322	0.0374	0.0634	0.0279	0.0522
471	0.0996	0.1453	0.0285	0.0344
620	0.1311	0.2118	0.0409	0.0303
891	0.0919	0.2776	0.0350	0.0611
1092	0.0913	0.2052	0.0480	0.0469
1203	0.0747	0.2058	0.0623	0.0694
1566	0.0629	0.1975	0.0712	0.0706
2072	0.0581	0.1548	0.0605	0.0730
3083	0.0581	0.1530	0.0700	0.0694
4013	0.0581	0.0996	0.0706	0.0736
8000	0.0991		0.0937	

According to these tables, we can find that both of two measures (recall and false positive rate) from the k-means have a larger difference compared to the ones resulting from the affinity propagation (averaged at less than 1% for all the cases) which is actually the cornerstone of our idea.

5 Conclusion

In order to effectively handle the real-time problem in attack detection, a compressed model is introduced in this paper to shorten the detection time. Actually, we have made efforts to improve the performance from both compression and distance matrix extraction. To be specific, the first endeavor is to horizontally select useful attributes of the training data and then vertically extract the representative data from a larger dataset through clustering methods. Sequentially, we have proposed the idea that we can extract the distance matrix during the vertical compression phase for future detection use.

Comprehensive experiments have been conducted to demonstrate the high performance resulting from the compressed model we proposed. In the best case, it runs 184 times faster than the traditional one without our model compression, and neither the recall (detection rate) nor the false positive rate sacrifices, both of which are differentiate from traditional ones only in a very small range (less than 1% on average).

As future work, we plan to extend our compressed model to other related fields. If possible, a feasible framework could be proposed to tackle those similar problems.

References

1. Wenke, L., Salvatore, J.S.: Data Mining Approaches for Intrusion Detection. In: USENIX Security Symposium (1998)
2. Varun, C., Arindam, B., Vipin, K.: Anomaly Detection: A Survey. ACM Computing Surveys 41(3), 1–58 (2009)
3. Frey, B.J., Dueck, D.: Clustering by Passing Messages between Data Points. Science 315(5814), 972–976 (2007)
4. World Internet Usage Statistics, http://www.internetworldstats.com/stats.htm
5. Srinivas, M., Andrew, H.S.: Feature Selection for Intrusion Detection Using Neural Networks and Support Vector Machines. Annual Meeting of the Transportation Research Board (2003)
6. Srilatha, C., Ajith, A., Johnson, P.T.: Feature Duduction and Ensemble Design of Intrusion Detection Systems. Computer & Security – COMPSEC 24(4), 295–307 (2005)
7. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), 27 (2011)
8. Information and Computer Science of University of California, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
9. The 3rd Cybersecurity Data Mining Competition, http://www.csmining.org/cdmc2012/
10. Martin, R.: Snort: Lightweight Intrusion Detection for Networks. In: USENIX Systems Administration Conference LISA, pp. 229–238 (1999)
11. Daniela, B., Kavé, S., Margin, M.: A Signal Processing View on Packet Sampling and Anomaly Detection. In: IEEE INFOCOM, pp. 713–721 (2010)
12. Yang, L., Tian-Bo, L., Li, G., Zhi-Hong, T., Lin, Q.: Optimizing Network Anomaly Detection Scheme Using Instance Selection Mechanism. In: Global Telecommunications Conference, pp. 1–7 (2009)
13. Vasiliadis, G., Antonatos, S., Polychronakis, M., Markatos, E.P., Ioannidis, S.: Gnort: High Performance Network Intrusion Detection Using Graphics Processors. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 116–134. Springer, Heidelberg (2008)
14. Holte, R.C.: Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. Machine Learning 11(1), 63–90 (1993)
15. Lincoln Laboratory, Massachusetts Institute of Technology, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html
16. Enrique, V.: New Formulation and Improvements of the Nearest-neighbor Approximating and Eliminating Search Algorithm. Pattern Recognition Letters 15(1), 1–7 (1994)
17. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10 (1966)
18. Joseph, B.K.: An Overview of Sequence Comparison: Time Warps, String Edits, and Macromolecules. Siam Review 25(2) (1983)
19. Victor, B.: Metric Spaces: Iteration and Application. Cambridge University Press (1985)
20. Noble, C.C., Cook, D.J.: Graph-based Anomaly Detection. In: 9th ACM SIGKDD, pp. 631–636 (2003)

A Digital Forensic Framework for Automated User Activity Reconstruction

Jungin Kang, Sangwook Lee, and Heejo Lee

Division of Computer and Communication Engineering,
Korea University
Seoul, Republic of Korea
{keijin,ook7777,heejo}@korea.ac.kr

Abstract. User activity reconstruction is a technique used in digital forensic investigation. Using this technique, digital forensic investigators extract a list of user activities from digital artifacts confiscated at the crime scene. Based on the list, explicit knowledge about the crime, such as motive, method, time, and place, can be deduced. Until now, activity reconstruction has been conducted by manual analysis. This means that the domain of the reconstructed activities is limited to the personal knowledge of the investigators, so the result exhibits low accuracy due to human errors , and the process requires an excessive amount of time. To solve these problems, this paper proposes a digital forensic framework-SigDiff for automated user activity reconstruction. This framework uses a signature-based approach. It comprises an activity signature generation module, signature database, digital artifact collection module, and activity reconstruction module. Using SigDiff, the process of user activity reconstruction can be performed accurately with a high retrieval rate and in a reduced time span.

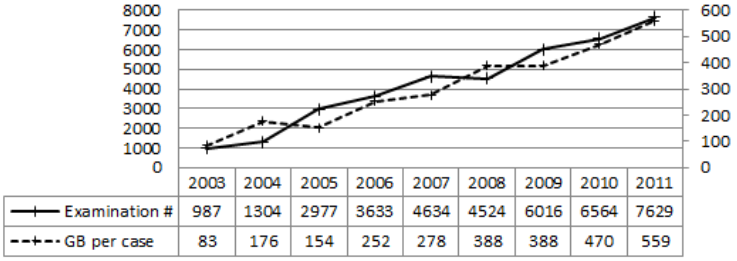
Keywords: digital forensic framework, activity reconstruction, signature-based forensics.

1 Introduction

With the increasing use of personal digital devices, the number of crimes that use digital devices as tools is rising. Criminals use digital devices to find information about victims or buy drugs and weapons. In some cases, the digital devices are used as tools for cybercrimes, such as information leakage and phishing. To respond to such crimes, investigators from governments and enterprises use digital forensic techniques. The investigators analyze digital devices to extract digital artifacts such as Web search histories and program histories . These artifacts can be evidence of user activities that were performed on the device. Using the extracted activity information, the investigators plan the direction of the investigation or present the artifacts to a court as proof of the guilt of a suspect.

According to FBI statistics [1], the number of digital forensic investigations and the storage size per case are increasing (Fig.1).The rise in storage size means

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 263–277, 2013.



	2003	2004	2005	2006	2007	2008	2009	2010	2011
Examination #	987	1304	2977	3633	4634	4524	6016	6564	7629
GB per case	83	176	154	252	278	388	388	470	559

Fig. 1. Increases in the number of digital forensic examinations and storage size

that the time required for each analysis is increasing. Garfinkel [2] classified this problem as the upcoming digital forensic crisis that needs to become a focus.

To solve the problem, digital forensic investigators use digital forensic tools to analyze digital artifacts. These tools abstract the digital data into easily understandable formats or automatically extract some important information. For example, listing the files on a disk or extracting an Internet history are frequently used functions of digital forensic tools.

However, the current tools only list the artifacts that are extracted from digital devices. This means that the reasoning process about what user activity generated the artifact is still manual work for an investigator. For example, when a user executes a messenger software on a digital device, the software will leave file and registry artifacts on the device. Current digital forensic tools only display the list of file and registry artifacts to the investigator. To deduce the messenger activity, the investigator should have additional knowledge about the relationship between the artifacts and the messenger activity.

The manual process causes the following problems: first, the domain of the reconstructed activities is highly limited by the personal knowledge and experience of the investigators; second, the activity reconstruction process is time-consuming, and the results suffer from low accuracy.

In this paper, we propose a digital forensic framework SigDiff to solve the problems of manual user activity reconstruction. This framework adopts the signature-based approach that is widely used for rapid but precise identification of data in numerous systems, such as antivirus engines or intrusion detection systems. SigDiff comprises an activity signature generation module, activity signature database, digital artifact collection module, and activity reconstruction module. Using these components, SigDiff accumulates user activity signatures based on a predefined activity model with corresponding artifacts. The activity signatures are used in digital forensic investigation for automated user activity reconstruction with a higher retrieval rate, increased accuracy, and reduced time.

The remainder of this paper is organized as follows: Section 2 presents the background on digital forensics and user activity reconstruction. Section 3 introduces previous work on signature-based user activity reconstruction. Section 4 provides a detailed description of SigDiff and its components. Section 5 presents

the proof-of-concept tools and evaluation results. Finally, in Section 6 we list future research directions with our conclusions.

2 Background

When a digital device is used , the user gives the device some input for a specific purpose. The digital device processes the series of inputs and displays the respective results to the user. In the process, some artifacts will be left on the physical media of the device.

For example(Fig.2), the user may want to send messages to someone. Web browser software is used to access the Website of a messenger, download a client installer package, install the messenger, logon to the messenger, and send messages. When this series of user activities is conducted, the Web browser software, messenger installation package, and messenger client may leave Internet history, registry, and file artifacts [3].

Digital forensics is the process in which investigators collect digital devices from a crime scene, recover artifacts from the devices, reconstruct suspect activities from the artifacts, and present the artifacts or devices to a court as evidence for the activities.

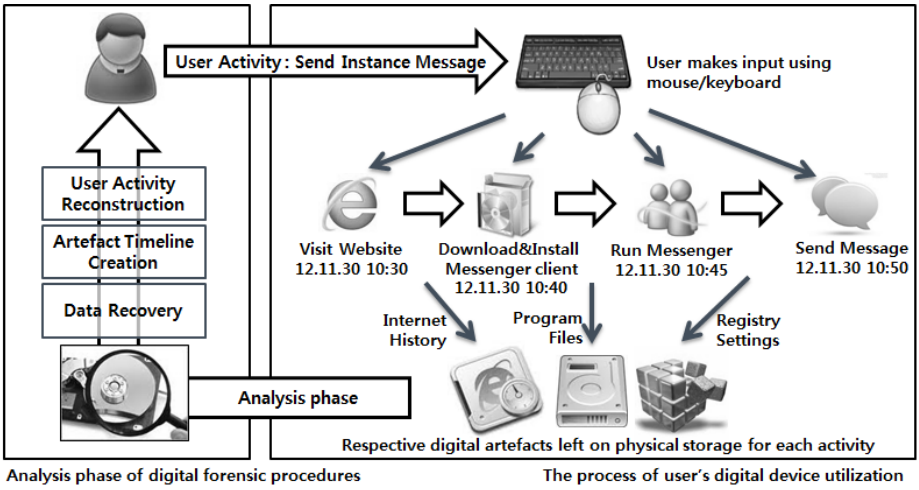


Fig. 2. Example of process for digital device utilization

In the previous example, the user may be under indictment for technology leakage. Investigators confiscate the digital device of the user and extract artifacts from the device. From the artifacts, the investigators can reconstruct a series of user activities that are related to the instant messaging.

The digital forensic investigation is performed by means of a predefined investigational procedure(Fig.3). Numerous procedural models have been published

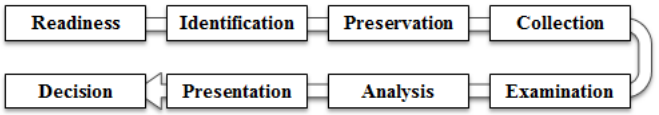


Fig. 3. DFRWS model with readiness phase

and adopted by various organizations. In this paper, the DFRWS model [4] proposed in the first Digital Forensic Research Workshop (DFRWS) is adopted, with an additional forensic readiness phase [5].

First, in the readiness phase, the organization prepares detailed procedures, tools, and human resources to prepare for an investigation. In the identification phase, the organization identifies an incident and arranges resources for an investigation. Subsequently, investigators preserve the crime scene and collect digital devices during preservation and collection phases. In the examination and analysis phases, the investigators gather meaningful information from the media seized. Finally, in the presentation phase, the investigators present the information and evidence to the court for a decision.

Typically, the collected evidence consists of physical media that store data in a bitwise manner. In the analysis phase, the investigators should interpret sequences of bits to obtain more meaningful information. In other words, the investigators abstract the data from bit level to a higher level. This process has been defined as digital forensic abstraction by Carrier [6].

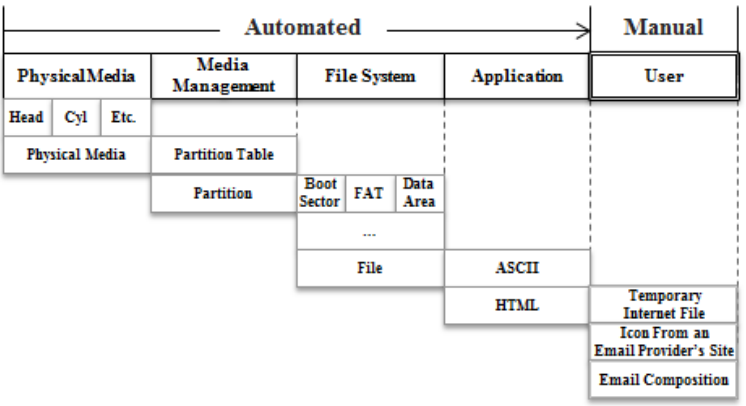


Fig. 4. Example of Carriers abstraction layers with a user layer for an HTML file

In Carriers model, the data on physical media can be successively abstracted into a media management layer, file system layer, and application layer. For some digital forensic investigations, there are requirements for one or more layers that describe user activity. For example(Fig.4), HTML data abstracted in the

application layer may be interpreted as a temporary Internet file that was created by user activity for email composition. The data abstracted in the user layer are useful for digital investigations that target a person. Using the abstracted activities, such as for Web searches, email, and SNS, the investigators can easily deduce information such as the characteristics, mentality, or recent location of the suspect.

To move efficiently up the abstraction levels, some digital forensic tools such as EnCase [7] and FTK [8] have been developed. These tools are widely used in the field by real forensic investigators. The tools interpret the collected physical-level digital data into a human readable format, mostly at the application level.

However, most current tools do not support abstraction from the application level to the user level. Consequently, investigators necessarily analyze millions of application-level artifacts by a manual process. These circumstances cause the following problems:

1. Excessive Time Consumption
   The investigators primarily identify application-level artifacts individually. In the example of the HTML file, the investigator first extracts some meaningful words from the file name and file data. After that, information is gathered from the words and the source activity is deduced. Although there are keyword-based searching techniques [9], timeline-based approaches [10] and visualization techniques [11] for reducing the amount of data to analyze, this manual process still requires an excessive amount of time.
2. Low Retrieval Rate
   To deduce the source activity, investigators should have previous knowledge of the activity. In other words, investigators are unlikely to retrieve activities for domains that are unfamiliar. Moreover, in most cases, no meaningful words from an artifact can be recognized by the investigator, causing a disregard of the artifact. For these reasons, a considerable number of user activities are omitted in the analysis phase, resulting in a low retrieval rate.
3. Decreased Accuracy
   The fact that the activity reconstruction process is performed manually means that there can be human errors. Investigators may misunderstand the meaning of extracted words, resulting in an incorrect result. Although there are examination environments such as Vise [12], the examination is difficult to perform for the entire reasoning process due to the limitation of available time.

3 Related Work

To solve the problems of time consumption, retrieval rate, and accuracy, there has been work that shares information about artifacts and source activities. The researchers have analyzed user activities with frequently used applications such as messengers [13] and Internet download managers [14]. With the information gathered, some tools have started to support limited user activity abstraction; for example, the extraction of USB storage activities [15] or Internet activities[16].

Despite the efforts that have been made, the amount of shared information is still insufficient, and the tools that provide fixed extraction functions have limited scalability. Thus, a scalable automated digital forensic system that rapidly performs activity reconstruction with a high retrieval rate and high accuracy is required. James[17] and Hargreaves[18] adopted a signature-based approach to solve the problems. Using the signature-based approach, the signature of information is stored in a database that is queried when the information is required. This approach has the advantage that known information can be searched in a fast but accurate way, and it has been adopted in various identification systems such as antivirus software and IDS/IPS . Although the retrieval rate is limited by the size of the database, at least this system is scalable and retrieves information effectively.

James[17] has proposed a novel approach to signature-based activity reconstruction. A simple activity is performed repeatedly in a virtual machine, and then artifacts with changed timestamps are filtered out. The signature is generated using the generalized path string of the artifact. Hargreaves [18] proposed a script-based signature generation method. The signature is applied to the artifact super-timeline to reconstruct higher-level events.

The previous work on signature-based user activity reconstruction was focused on adopting the approach for digital forensics, thus simple methods of activity signature generation were proposed. In this paper, we continue the work to describe a signature-based digital forensic framework that covers the entire procedure for user activity reconstruction. Using this framework, investigators can automatically reconstruct complex user activities in a significantly reduced timespan, but with a higher retrieval rate and increased accuracy.

4 SigDiff: Signature-Based Digital Forensic Framework

SigDiff, the signature-based digital forensic framework, is composed of the following parts (Fig.5): an activity signature generation module, an activity signature database, a digital artifact collection module, and an activity reconstruction module.

The activity signature generation module is used to construct the activity signature database. This module is used in the forensic readiness phase. It generates signatures using a predefined user activity model and sends those signatures to the database. The digital artifact collection module is used in the collection phase of a digital investigation. It extracts artifacts from a collected digital device or directly collects artifacts from a live digital device. The activity reconstruction module matches the collected artifacts to the signatures stored in the database and reconstructs the user activity timeline. This module is used in the forensic examination and analysis phase.

4.1 Activity Signature Generation

The activity signature generation module performs a series of processes to generate a user activity signature (Fig.6).

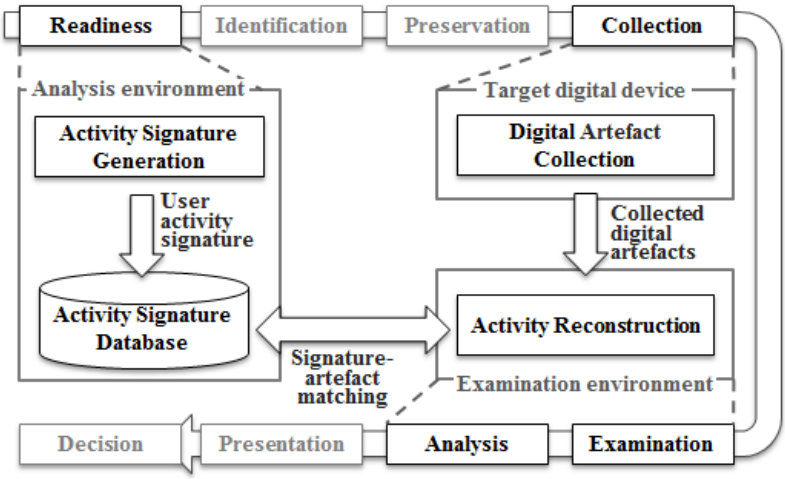


Fig. 5. SigDiff architecture

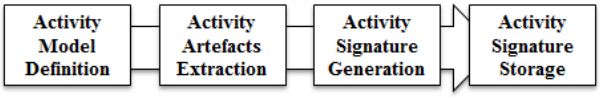


Fig. 6. Procedures for activity signature generation

First, in the activity model definition phase, the investigator defines the user activity model. Using the model, the module extracts artifacts of the activity model, generates the signature, and stores the signature in the activity signature database.

Activity Model Definition. When a user generates events such as mouse or keyboard input, the application processes the corresponding tasks. In the process, the application may leave artifacts on physical media. A user activity is a series of user-level events(Fig.7) performed for a single purpose. For example, when a user performs an activity defined as Install messenger software, a series of user inputs from clicking on Next and Finish buttons will be sent to the software installer. The software installer receives the inputs and writes messenger files on the physical media of the device. After the activity has been performed, there will be installed files. In other words, the *Activity artifacts*.

The ideal case of user activity definition is that the defined user activity includes only a single user-level event. For example, user inputs for clicking buttons of the installer may be defined as multiple activities such as first clicking the Next button and then clicking the Finish button. However, in this case, the number of activities defined will be too large, and the time required for defining

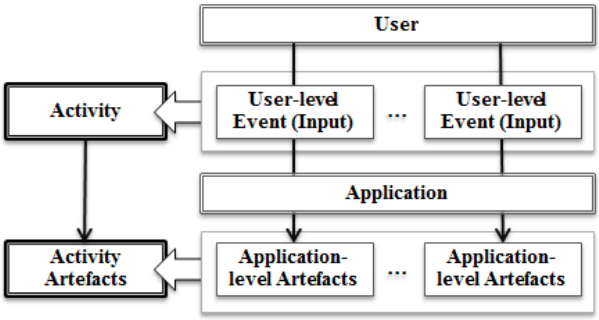


Fig. 7. Model for user activity and its artifacts

activities will be excessive. Thus, the investigator may bind a series of events that are performed for the same purpose, and define this as one activity.

There are two approaches for defining user activities with respect to corresponding user events: the model-first approach and the event-first approach.

Model-First Approach. In the model-first approach, the user activity model for a topic is first defined. The activity model is a sort of usage scenario and is formulated in a finite-state machine (FSM). Each state of the FSM is a state after some activity has been conducted. Each transition function represents an activity, which is a series of user events. For example, a simple activity scenario for a messenger can be defined with the model in (Fig.8). Once an activity model is defined, a series of user events is defined for each transition function.

Using the model-first approach, the activity artifact required by the investigation can be extracted quickly and with flexibility. In other words, this approach is adequate when the investigator has a crime situation composed of a series of activities.

Event-First Approach. In the event-first approach, a series of user-level events for a topic is first collected. The activity model FSM is defined with refined events. If the user events are collected from a large number of users, then the activity model can reflect a trend in user activity. Monitoring, collecting user events, and interpreting the events as user behavior at the user interface are research areas of human-computer interfaces[19]. Further research is required from a digital forensics perspective.

Activity Artefacts Extraction. The activity artifacts extraction process is performed on the basis of the predefined user activity model. In this framework, the user events are replicated on a virtual machine to extract the respective activity artifacts. In the virtual machine, an extraction method based on either state comparison or system monitoring is used to extract activity artifacts.

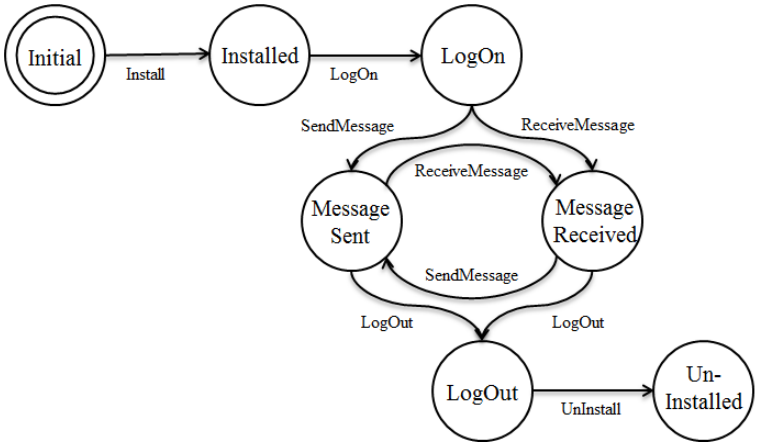


Fig. 8. Example of a messenger activity model

State-Comparison-Based Extraction. Using the state-comparison-based extraction method (Fig.9), a set of virtual machine snapshots is generated corresponding to the states in the predefined user activity model. Thus, the differences between two connected snapshots can be regarded as artifacts of an activity.

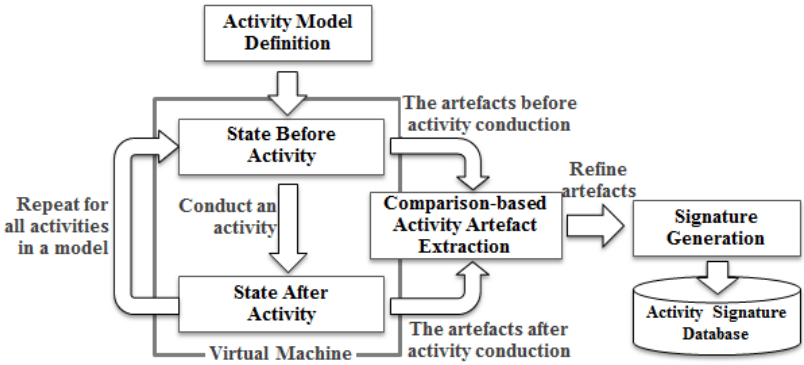


Fig. 9. State-comparison-based extraction

System-Monitoring-Based Extraction. The system-monitoring-based extraction method (Fig.10) does not save all the snapshots. A series of user inputs from the activity model is performed continuously. However, the alterations generated in the virtual machine are monitored in real time. For example, system tracking functions such as CreateFile or RegCreateKey in Windows are called to extract artifacts. Once all the activities in a model are performed, sets of artifacts and user activities are matched.

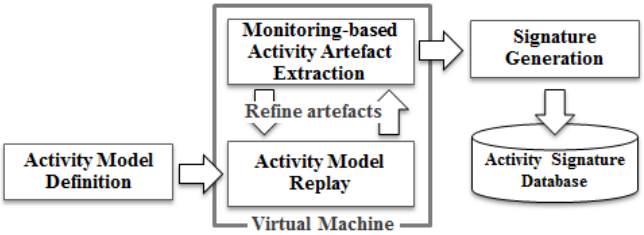


Fig. 10. System-monitoring-based extraction

The comparison-based extraction method has the advantage of scalability. If the source activity model is extended, the new activities and states can easily be added to the saved virtual machine snapshots. However, additional time is consumed for creating and comparing the snapshots. The monitoring-based approach is good for rapid artifact extraction, because the time consumption for snapshots can be reduced. Moreover, it can track the source applications of artifacts for background noise filtering. However, this approach does not respond easily to extension of the source activity model. For all approaches, the artifact refinement process is required for eliminating background noises, as mentioned by James [17]. However, the artifacts that rarely appear cannot simply be omitted, because such an artifact may be a unique sign of a specific activity. In this framework, the artifact extraction for an activity model is performed multiple times. The frequency of an artifact for all repetitions is counted as the appearance probability and will be provided to the investigator. The background noise artifact, which is defined as the artifact that matches multiple activities in the database, is eliminated in the reconstruction phase.

Activity Signature Generation. An artifact is generally composed of timestamps, metadata, and data. For example, a file artifact in a file system is composed of the file data, the file path, the size as metadata, and timestamps of reading, writing, and creation. The activity signature is generated using these elements. For example, the NSRL of the NIST [20] uses hashed file data as the signature, James [17] used the path string, and Hargreaves [18] used various sources to generate a script as the signature.

In this paper, we use a predefined variable table to partially automate the signature generation procedure. First, the investigator defines a table that contains multiple variables such as environment values, specific paths, or user information. Each variable is composed of a tag and a value. The tag is simply the name of the variable. The value is a regular expression that describes the corresponding artifact string, which can be metadata such as the file path or a string extracted from the data. The source of the artifact string is dependent on the type of artifact. Table 1 is an example of a variable table.

Table 1. Example of variable table

Tag	Value
<name>	Investigator12
<userID>	invID12
<keywords>	KeywordA|KeywordB
<%TEMP%>	C:\\Users\\Investigator12\\AppData\\Local\\Temp
<%IE_TEMP%>	C:\\Users\\Investigator12\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\Content.IE5\\[a-zA-Z0-9]{8}

Algorithm 1. Signature generation algorithm

```
1: procedure SIGNATURE_GENERATION(Artifact_strings A[0...n], Variables V[0...m])
2:     while i from 0 to n do
3:         while j from 0 to m do
4:             S[i] ← Replace_Matched(A[i], V[j])        ▷ S = list of signatures
5:         end while
6:     end while
7:     return S
8: end procedure
9: procedure REPLACE_MATCHED(Artifact_string a, Variable v)
10:    find matching part of a, v.regex
11:    s ← replace matching part of a to v.tag            ▷ s = signature
12:    return s
13: end procedure
```

Using the predefined variable table, each artifact string is compared with a regular expression for every variable. If a matching part of the artifact string is found, then that part is replaced with the tag of the matching variable. The activity signature is the processed artifact string. Algorithm 1 describes the process. Table 2 contains examples of artifact strings and the corresponding signatures that are generated using the variable table described in table 1. The generated

Table 2. Examples of artifact strings and signatures

Artifact string	Signature
C:\Users\Investigator12\AppData\Local\Temp \keywordA\invID12.log	<%TEMP%>\<keywords>\<userID>.log
C:\Users\Investigator12\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\ZG8IPVA7\siteLogo.gif	<%IE_TEMP%>siteLogo.gif
Computer\HKEY_CURRENT_USER\Software\keywordA\invID12\key	Computer\HKEY_CURRENT_USER\Software\<keywords>\<userID>\key
http://www.keywordA.com/view.php?userid=invID12&mode=sendFile	http://www.<keywords>.com/view.php?userid=<userID>&mode=sendFile

activity signatures are sent to the activity signature database with some information, such as the activity model topic, activity model, and corresponding user inputs.

4.2 Digital Artefact Collection Module

The digital artifact collection module lists artifacts from media acquired at the crime scene. This can be performed on a live system or from a media image. The listed artifacts are generated as signatures by the method, which is exactly the same as the method that was used in the activity signature generation phase before the incident. The extracted artifact signatures are sent to the activity reconstruction module for analysis.

4.3 Activity Reconstruction Module

The activity reconstruction module queries the activity signature database with the artifact signatures extracted from the crime scene. If matching signatures are found, the database sends the corresponding information about the activities. Typically, the number of artifact signatures extracted from the crime scene exceeds one million. Although the reconstruction can be performed automatically, the time consumption is still excessive.

Traditional searching techniques based on time, category, or keyword can be applied to accelerate the reconstruction. Based on the timestamps of the artifacts, the investigator can request the information for an artifact that is used in a specific or recent timeline. To perform the search based on a category or keyword, the investigator submits a specific keyword to the database and receives a list of all related signatures. The acquired list of signatures is searched for the artifact signatures extracted from the crime scene. If a matching signature is found, then the information about the signature will be requested from the activity signature database. The reconstruction time can also be reduced by a frequency-based method. First, the database calculates the list of frequently used artifacts of directory or registry paths, such as the %Program Files% directory or the HCU\Software registry key. After that, the list is sent to the reconstruction module in order of priority. Using these acceleration methods, the extracted artifacts can be automatically abstracted as user activities with a high retrieval rate and high accuracy in a reduced time.

5 Implementation and Evaluation

For proof-of-concept, tools were implemented for each module. Figure 11 is a screenshot of the activity signature generation tool that modeled user activities on the topic TrueCrypt. The tool is based on the model-first approach with activity artifacts extraction based on state comparison. It supports automated signature generation for files, registry, and Internet history. The generated signatures are sent to a signature database.

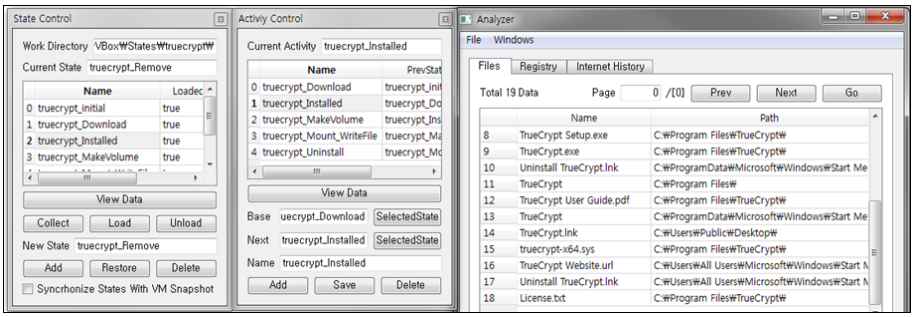


Fig. 11. Activity signature generation tool

Figure 12 is a result screen of the activity reconstruction tool after some messenger activities were automatically reconstructed. The tool compares artifact signatures extracted from the crime scene with those stored in the activity signature database.

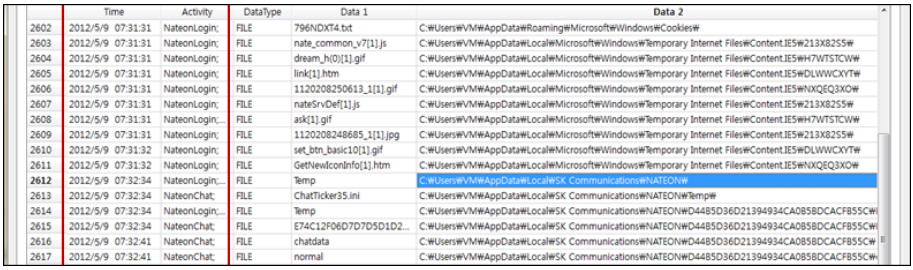


Fig. 12. Activity reconstruction tool

Using the tools, we designed multiple user activity models, including Google search, Gmail, MSN Messenger, Dropbox, and TrueCrypt as well as some popular Korean applications and Web services. During the signature generation phase, we generated 5000 more file artifact signatures from the activity models. For an evaluation, the signatures were tested on a machine so that all the modeled activities were performed. As a result, 100% of the activity signatures were successfully extracted from 85,000 file artifacts in 19.5 s on average.

6 Conclusion and Future Research Directions

In this paper, we proposed a novel signature-based digital forensic framework that assists investigators to reconstruct user activities automatically. We presented not only the processes in each module of the framework but also techniques for efficient and effective user activity reconstruction. Research on signature-based digital

forensic approaches is still in the early stages. We propose the following directions for future research:

- *Definition of criminal user behavior on digital devices.* The research will focus on what kind of user activities should be defined as the user activity model.
- *User activityevent matching algorithm.* Since the collected sequences of user events are varying, it is difficult to define an activity as a sequence of events. Thus, a formalized algorithm for activityevent matching is required.
- *Research on efficient activity signature databases.* The database could involve technologies such as cloud computing and in-memory computing to improve the query speed.
- *Automated and generalized signature generation algorithm.* In this paper, the signature generation still requires investigators to define variables manually. In future work, the generation procedure could be automated using a string pattern recognition approach with improved retrieval rate and accuracy.
- *Fast signature matching algorithm.* Querying all the signatures extracted from a crime scene still requires an excessive amount of time. It is necessary to develop a fast signature algorithm without sacrificing the accuracy and retrieval rate.

Acknowledgements. This research was supported by the R&BD Support Center of Seoul Development Institute and the South Korean government (WR080951) and the Public welfare&Safety research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012M3A2A1051118 2012051118).

References

1. Regional Computer Forensics Laboratory: Annual report for fiscal year 2003-2011 (2011)
2. Garfinkel, S.L.: Digital forensics research: The next 10 years. Digital Investigation 7, S64–S73 (2010)
3. Van Dongen, W.S.: Forensic artefacts left by Windows Live Messenger 8.0. Digital Investigation 4(2), 73–87 (2007)
4. Palmer, G.: A road map for digital forensics research-report from the first Digital Forensics Research Workshop (DFRWS), Utica, New York (2001)
5. Rowlingson, R.: A ten step process for forensic readiness. International Journal of Digital Evidence 2(3), 1–28 (2004)
6. Carrier, B.: Defining digital forensic examination and analysis tools using abstraction layers. International Journal of Digital Evidence 1(4), 1–12 (2003)
7. EnCase forensic, `http://www.guidancesoftware.com/forensic.htm`
8. Forensic toolkit, `http://accessdata.com/products/computer-forensics/ftk`
9. Beebe, N.L., Clark, J.G.: Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results. Digital Investigation 4, 49–54 (2007)
10. log2timeline, `http://log2timeline.net/`

11. Teelink, S., Erbacher, R.F.: Improving the computer forensic analysis process through visualization. Communications of the ACM 49(2), 71–75 (2006)
12. Arnes, A., Haas, P., Vigna, G., Kemmerer, R.: Digital forensic reconstruction and the virtual security testbed ViSe. Detection of Intrusions and Malware & Vulnerability Assessment, 144–163 (2006)
13. Reust, J.: Case study: AOL instant messenger trace evidence. Digital Investigation 3(4), 238–243 (2006)
14. Yasin, M., Cheema, A.R., Kausar, F.: Analysis of Internet Download Manager for collection of digital forensic artefacts. Digital Investigation 7(1), 90–94 (2010)
15. Carvey, H., Altheide, C.: Tracking USB storage: Analysis of windows artifacts generated by USB storage devices. Digital Investigation 2(2), 94–100 (2005)
16. Oh, J., Lee, S., Lee, S.: Advanced evidence collection and analysis of web browser activity. Digital Investigation 8, S62–S70 (2011)
17. James, J.I., Gladyshev, P., Zhu, Y.: Signature Based Detection of User Events for Post-mortem Forensic Analysis. Digital Forensics and Cyber Crime, 96–109 (2011)
18. Hargreaves, C., Patterson, J.: An automated timeline reconstruction approach for digital forensic investigations. Digital Investigation 9, S69–S79 (2012)
19. Hilbert, D.M., Redmiles, D.F.: Extracting usability information from user interface events. ACM Computing Surveys (CSUR) 32(4), 384–421 (2000)
20. National Institute of standards and technology, National software reference library, `http://www.nsrl.nist.gov/`

Increasing Automated Vulnerability Assessment Accuracy on Cloud and Grid Middleware*

Jairo Serrano[1], Eduardo Cesar[1], Elisa Heymann[1], and Barton Miller[2]

[1] Computer Architecture & Operating Systems
Universitat Autònoma de Barcelona
Barcelona, Spain
{jairodavid.serrano,eduardo.cesar,elisa.heymann}@uab.es
[2] Computer Sciences Department
University of Wisconsin-Madison
Madison, WI, USA
bart@cs.wisc.edu

Abstract. The fast adaptation of Cloud computing has led to an increase in novel information technology threats. The targets of these new threats range from large scale distributed system, such as the Large Hadron Collider by the CERN, to industrial (water, power, electricity, oil, gas, etc.) distributed systems, i.e. SCADA systems. The use of automated tools for vulnerability assessment is quite attractive, but while these tools can find common problems in a program's source code, they miss a significant number of critical and complex vulnerabilities. In addition, middleware systems frequently base their security on mechanisms such as authentication, authorization, and delegation. While these mechanisms have been studied in depth and can control key resources, they are not enough to assure that all application's resources are safe. Therefore, security of distributed systems have been placed under the watchful eye of security practitioners in government, academia, and industry. To tackle the problem of assessing the security of critical middleware systems, we propose a new automated vulnerability assessment approach, called Attack Vector Analyzer (AvA), which is able to automatically hint at which middleware components should be assessed and why. AvA is based on automating part of the First Principles Vulnerability Assessment, an analyst-centric (manual) methodology that has been used successfully to evaluate many production middleware systems. AvA's results are language-independent, provide a comprehensive assessment attack vector in the middleware, and it is based on the Common Weakness Enumeration (CWE) system, a widely-use labeling of security weaknesses. Our results are contrasted against a previous manual vulnerability assessment of the CrossBroker grid resource manager, and corroborate which middleware components should be assessed and why.

Keywords: Vulnerability Assessment, Security, Weakness, Attack Vector, Cloud, Grid, Middleware.

* This research has been supported by the MEC-MICINN Spain under contract TIN2007-64974 and by Department of Homeland Security grant FA8750-10-2-0030.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 278–294, 2013.

1 Introduction

Vulnerability assessment is a security task that is insufficiently addressed in most existing Grid and Cloud projects, and even in Supervisory Control and Data Acquisition (SCADA) [1] systems it is an afterthought. Such projects use middleware software which usually bases its security on mechanisms such as authentication, authorization, and delegation. These mechanisms have been studied in depth and carry out control of key resources, but they are not enough to assure that all middleware resources are safe. However, middleware systems usually do not undergo a thorough vulnerability assessment during their life cycle, resulting in overlooked security flaws. One possible solution would be to use existing automated tools such as Coverity Prevent [2] or HP Fortify SCA [3] to analize source code for previously known threats, but even the best of these tools may find only a small percentage of the serious vulnerabilities [4].

A thorough vulnerability assessment requires a systematic approach that focuses on the key resources to be protected and allows for a detailed analysis of those parts of the code related to those resources and their trust relationships. *First Principles Vulnerability Assessment (FPVA)* [5] was designed to address these requirements. FPVA have been successfully applied to many large and widely-used middleware systems, such as Condor [6], a high-throughput scheduling system; Storage Resource Broker (SRB) [7], a data grid management system; Crossbroker [8], a Grid resource management for interactive and parallel applications, among others [9].

FPVA starts with an architectural analysis. This step identifies key structural components in a middleware system, including modules, threads, processes, and hosts. The second step is a resource analysis, which identifies the key resources accessed by each component, and the operations supported on those resources. Privilege analysis is the next step; it identifies the trust assumptions about each component, answering such questions as how are they protected or who can access them. A complex but crucial part of trust assumptions and privilege analysis is evaluating trust delegation. By combining the information from the first two steps, we determine what operations a component will execute on behalf of another. The results of these steps are documented in diagrams that provide a roadmap for the last stage of the analysis, the manual middleware source code inspection. This top-down, architecture-driven analysis, can also help to identify more complex vulnerabilities that are based on the interaction of multiple system components and are not amenable to local code analysis.

In our use of FPVA, we have noticed that there is a gap between the three initial steps and the manual source code inspection. The security practitioner must provide expertise about the kind of security problems that the systems may present during the last stage of the analysis (e.g. depending on the language used the analyst might look for different kinds of vulnerabilities) and be creative enough to discover new vulnerabilities. Hence, this gap directly affects the quality of the vulnerability assessment, because security flaws may be overlooked due to either incomplete analyst knowledge or insufficient time for an in-depth analysis. We have observed that the security practitioner's knowledge is similar to that

described by several vulnerability classification systems, such as the Common Weakness Enumeration (CWE) [10], Seven Pernicious Kingdoms [11], OWASP Top Ten [12], and Microsoft SDL [13]. So, much of this information can be codified in the form of rules, metrics, and scores to be applied automatically. The proposed *Attack Vector Analyzer (AvA)* presented in this paper automates the generation of hints for middleware vulnerabilities based on codified expert knowledge. AvA's results include a prioritized alert list of likely weaknesses, which can lead to exploitable vulnerabilities. Results are based on a detailed joint analysis of complex of the relationship between middleware components, which could be potentially attractive targets for the attackers.

This paper discusses how to address the FPVA "gap" without looking at the middleware source code, or parsing it, and the automatic AvA tool for systematically indicating which middleware components should be assessed and why, before the analyst shifts to the source code inspection.

The remainder of this paper is structured as follows. Section 2 introduces Attack Vector Analyzer and its components. Section 3 discusses a case study: AvA applied to CrossBroker, and its results compared to the ones obtained by a manual assessment of the same middleware. Related work is introduced in Section 4, and conclusions and future work are shown in Section 5.

2 The Attack Vector Analyzer

Before proceeding with the description of AvA, first we describe relevant aspects of the FPVA methodology. One of these aspects is the FPVA aim of concentrating the analyst's attention on the components and resources of the middleware system that are most likely to have critical vulnerabilities. The artifacts produced during the initial steps of FPVA include the architectural analysis that produces diagram of the structure of the system and the interactions among the different components and with the end users. In this diagram, the Attack Surface of the system can be defined.

The **Attack Surface** is the set of interfaces from which an attack might start; it tells security practitioners where to start looking for the attacker's initial behavior. The resource analysis step produces a description of each resource and its value as an end target (such as a database with personnel or proprietary information) or as an intermediate target (such as a file that stores access-permissions). These resources are the target of an exploit. We then define the **Impact Surface** as the set of locations in the code where exploits or vulnerabilities might be possible. Finally, the artifact produced by the privilege analysis step is a further labeling of the previous diagrams with trust levels and labeling of interactions with delegation information. In this diagram the Attack Vectors can be identified. An **Attack Vector** is a path from a point in the attack surface to a point in the impact surface.

Despite all the information gathered during the intial steps of FPVA, finding actual vulnerabilities in system during the component analysis depends on the implementation details of each component and the analyst's expertise.

As previously disscused, much of this knowledge can be found in several existing vulnerability classifications [10–13], and that, in consequence, can be codified to help automatically indicate which attack vectors of a given system should be analyzed and why. We have developed a methodology that follows this approach, and have implemented this methodology in a prototype tool called Attack Vector Analyzer (AvA) for demostrating how the FPVA gap can be reduced.

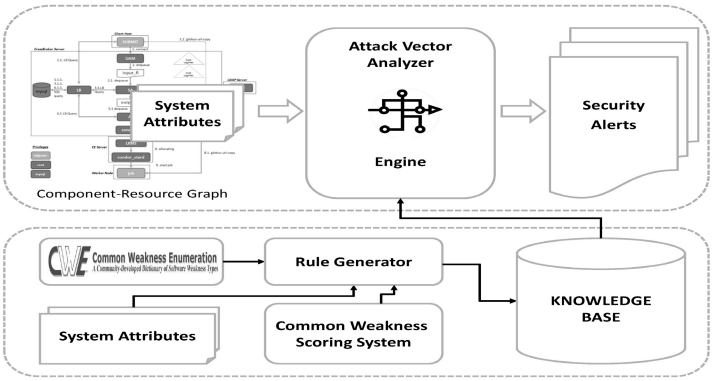


Fig. 1. Attack Vector Analyzer Architecture

In the following subsections, we describe our automated approach, depicted in Figure 1. It shows the components of the AvA architecture, composed of the (attack vector) analyzer engine, which receives as inputs a single modified version of the FPVA diagrams called the component-resource graph, and a knowledge base (KB) with codified rules. The KB is based on three elements: the most current knowledge about vulnerabilities (the CWE), a scoring system for these weaknesses (the CWSS), and outstanding information that characterizes middlewares in terms of safety (the system attributes). The outcome of the analyzer engine will be security alerts on what components of the system should be analyzed.

2.1 Building the Knowledge Base

In this section we describe the process followed for defining the propositions included in the AvA knowledge base (KB) based on the information provided by the Common Weakness Enumeration [10] system, the Common Weakness Scoring [14] system, and the system attributes extracted from the experience using FPVA. The information depicted in Figure 2 is used to build the set of rules that will guide the analysis of a target system.

Common Weakness Enumeration. CWE is a community initiative of security practitioners, used to describe common software weaknesses that can occur in software's architecture, design, code or implementation, and that can lead to exploitable security vulnerabilities. CWE could be roughly described as a three tiered approach, the first tier consisting of the full CWE list (hundreds of

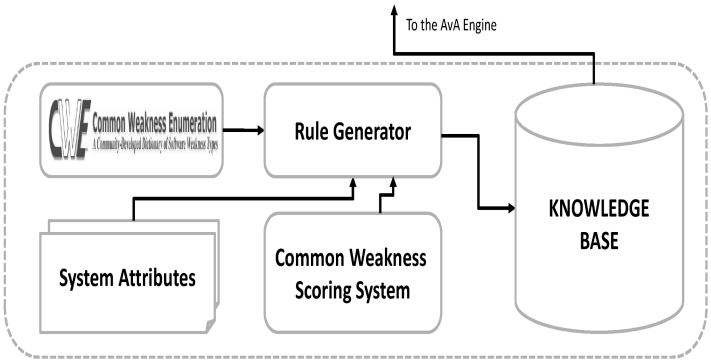


Fig. 2. AvA's Knowledge Base

nodes); the second tier consisting of groupings of individual CWEs, called the Development View; and the third tier consisting of high level groupings (pillars) of the intermediate nodes to define strategic classes of vulnerabilities, called the Research View. The Attack Vector Analyzer is based on the CWE Research View.

Common Weakness Scoring System. CWSS is a part of the CWE project, providing a scoring mechanism for weaknesses. CWSS scoring CWE's with 18

Table 1. AvA's System Attributes

Name	Description
Owner	The owner's component
User	The user's component
User-Admin Interface	Is the component part of the user or admin interface
Sanitize	Determines if the component performs data sanitizing operations
Transform Data	Determines if the component performs data transforming operations
Transfer Data	Determines if the component performs data transfering operations
Trust	Determines if the component performs trustworthy operations
Server Interaction	Determines if the component performs a DB, LDAP, etc., server operations
Timeout	Determines if the component performs timeout operations
Max/Min	Determines if the component performs data restriction operations
Third-party	Determines if the component performs local/remote third-party operations
Spoofing	Determines if the component performs operations against spoofing
Tampering	Determines if the component performs operations against tampering
Encryption	Determines if the component performs encryption operations
Attachment	Determines if the component performs attachment operations
Error Handling	Determines if the component performs operations against unexpected error
Client-Server	Determines if the component is installed on client or server host
Web	Determines if the component is a web service or application
Log-Backup	Determines if the component performs Log and/or Backup operations

different factors in three metric groups: (1) the Base Finding group, which captures the inherent risk of the weakness, confidence in the accuracy of the finding, and strength of controls; (2) the Attack Surface group, which captures the barriers that an attacker must cross to exploit the weakness; and (3) the Environmental group, which includes factors that may be specific to a particular operational context, such as business impact, likelihood of exploit, and existence of external controls.

System Attributes. System attributes [15, 16] are based on the information provided by several FPVA diagrams, developer team interviews, and user, administrator, and API documentation. The attributes included in AvA, shown in Table 1, have been derived from our experience with several different middleware systems, including Condor, SRB, MyProxy [17], gLExec [18], VOMS-admin [19], and CrossBroker.

Rule Generation. We use the CWE, CWSS, and system attributes to define assessment rules (propositions). First, we gathered twelve relevant elements from CWE's:

1. The weakness identificator
2. The weakness name
3. The weakness description
4. The weakness extended description
5. The programming language in which the weakness may occur
6. The consequence scope, which identifies an individual consequence that may be associated to the weakness
7. The consequence technical impact, which describes the technical impacts that can arise if an attacker attempts to exploit the weakness
8. The consequence notes, which provides additional commentary about its consequence
9. The mitigation description, which contains a single method for mitigating the weakness
10. The mapped node names, which identifies the name of the entry to which this weakness is being mapped in other taxonomies or classifications
11. The relationship, which contains a note regarding the relationships between CWE entries.
12. The observed example description, which presents an unambiguous correlation between the example being described and the weakness which it is meant to exemplify.

Second, a comprehensive search over the above twelve elements is conducted on the 682 CWE weaknesses to find the set of weaknesses related to each system attribute. For example, for the system attribute "Owner", 150 different weaknesses related to resource ownership were found; among these weaknesses we have the "CWE-282: Improper Ownership Management". It says "The software assigns the wrong *ownership*, or does not properly verify the *ownership*, of an object or resource". Similarly, we found between 20 to 150 relationships between each of the remaining system attributes and the weaknesses.

Table 2. CWSS Metric groups

Base Finding	Attack Surface	Environment
Technical Impact (TI)	Required Privilege (RP)	Business Impact (BI)
Acquired Privilege (AP)	Required Privilege Layer (RL)	Likelihood of Discovery (DI)
Acquired Privilege Layer (AL)	Access Vector (AV)	Likelihood of Exploit (EX)
Internal Control Effectiveness (IC)	Authentication Strength (AS)	External Control Effectiveness (EC)
	Authentication Instances (AI)	Prevalence (P)
	Level of Interaction (IN)	
	Deployment Scope (SC)	

Table 3. TI Scoring (CWSS adaptation)

Technical Impact	Critical	High	Medium	Low	None	Default	Unknown	Not Applicable
Score	1.0	0.9	0.6	0.3	0	0.6	0.5	0.3

Once the system attributes have been related to the weaknesses, we use a customized version of CWSS for producing the final set of rules. Rules consist of defining logical propositions for each value of each system attribute, to obtain a quantitative measurement of how the attribute influences to each of its related weaknesses.

More precisely, we verify this influence through the three metrics groups of the customized CWSS scoring system, linking up the system attributes with its relevant factors, and scoring accordingly to them. The relevant factors for the customized CWSS metrics groups are shown in Table 2. Thus, each factor has a list of possible values, and its corresponding score, e.g., the technical impact (TI) factor is shown in the Table 3.

Finally, we illustrate the kind of propositions in the KB with the next examples.

– Example I. Rule for the Owner attribute

  If "Owner" == "Administrator" then:
  "TI" == "Critical", "AP" =="Administrator"
  "AL"=="Enterprise", "AV"=="Private Network"

– Example II. Rule for the User-Admin Interface attribute

  If "User-Admin Interface" == "Yes" then:
  "TI" == "High", "AV"=="Local", "IN"=="Automated"
  "DI"=="High", "EX"=="High", "AS"=="Moderate"

It can be appreciated in both examples that the "Owner" and "User-Admin Interface" attributes are related with particular CWSS factors such as the TI factor, which are influenced by their corresponding values. In the first example, the TI factor answers with the "Critical" value when the "Owner" attribute

corresponds with the "Administrator" value. On the other hand, for the "User-Admin Interface" rule example, the TI factor assumes a "High" value, because the component being assessed is part of the attack surface. Then, with the knowledge base of rules stated, we proceed to describe the analysis process of the attack vectors of a middleware system.

2.2 Analyzing Attack Vectors

This section describes how the AvA engine analysis works on the attack vectors, the inputs required, and the security alerts produced. Below, we present the component-resource graph, which is the input of the AvA engine, and then we proceed with the AvA engine, as we can see depicted in Figure 3.

Component-Resource Graph. with the stated objective of reducing the gap between the outcomes from the initial FPVA stages, and the manual FPVA component code analysis, we have defined a structure called *Component-Resource graph* [20] for representing the results of these initial stages in a more suitable and readable format, which also includes relevant information about middleware components such as the system attributes. A component-resource graph is aimed to depict all attack vectors between middleware components, given that most of the generated FPVA diagrams describe particular operations of the middleware, such as submitting a job in a workload management system. Thereby, the order in which an attack vector is built is also quite clear because every edge in the diagrams is labeled with a number indicating when the interaction represented by the edge takes place, and also indicating if a node in the diagrams belong either to the attack or the impact surface.

To represent a component-resource graph we have chosen the GraphML format [21]. Basically, a component-resource graph is an XML file composed by the information gathered from the FPVA diagrams. Once the component-resource graph is correctly depicted in the graphml format, we proceed to apply the analysis process on it.

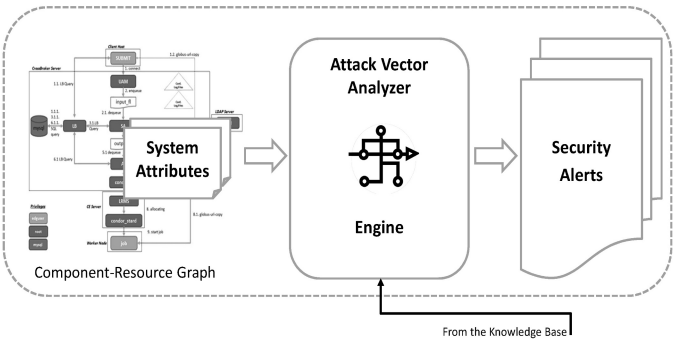


Fig. 3. AvA Engine

The AvA Engine Analysis. algorithm 1 shows the AvA engine analysis process. It begins reading the component-resource graph, this step allows the AvA engine to identify and to load the attack vectors of the middleware been assessed. Then, the AvA knowledge base is read, in order to load both the rules and the weaknesses - system attributes relationships. The next step of the engine is to start traversing each attack vector, component by component. For each component, its system attributes are fetched, and then assessed accordingly to the KB-rules, taking into account our customized CWSS system. Hence, the weaknesses related to the system attributes are assessed too. As a result of this step, a mark is obtained for each attribute associated to every weakness in every component of all identified attack vectors in the component-resource graph. For example, for a weakness CWE-X with associated attributes X_0, X_1, and X_2 (such as: owner, user, tampering, etc.) and an attack vector composed by components C_0, C_1, C_2, the results shown in table 4 can be obtained. After this first assessment, with the objective of obtaining a single score for each weakness associated to the attack vector, the algorithm applies the following steps:

(1) assign to each attribute the minimum score obtained by any component of the attack vector (line 12). Using the same example of table 4, the score of attribute X_0 for the weakness CWE-X will be $min(SC_{00}, SC_{10}, SC_{20})$. We have choosen the minimun value because it means that in that component at least, a weakness mitigation has been implemented; (2) once the minimum scores are computed then proceed to weigh them accordingly to the CWE research view (line 13) because the system attributes might have a different impact for each weakness depending of one pillar or another, i.e., a system attribute such as owner has a high weight regarding the "CWE-693 Protection Mechanism Failure" pillar, while the encryption attribute has a low weight. On the contrary, for the "CWE-330 Use of Insufficiently Random Values" pillar the encryption attribute has a high weight, while the owner has a low weight; at last, (3) the maximum weighed score for the weakness is computed, bearing in mind how was the score of its child weaknesses (line 14). It means that child weaknesses provide more information to the top-level weaknesses. Once all the weaknesses are processed, the last step is to sort them into the eleven pillars of the CWE research view accordingly to their maximum weighed score. Finally, the security alerts for the assessed attack vector are delivered as a hierarchical list of weighed weaknesses for each CWE pillar. Thus, we are systematically providing comprehensive information to the security practitioner, pointing out not only which vulnerabilities should be analyzed, but also why we should pay attention to them in the assessed attack vector.

Table 4. Individual marks

CWE-X			
Component \ Attribute	X_0	X_1	X_2
C_0	SC_{00}	SC_{01}	SC_{02}
C_1	SC_{10}	SC_{11}	SC_{12}
C_2	SC_{20}	SC_{21}	SC_{22}

Algorithm 1.

```
1. Read the Component-Resource graph
2.     Load the Attack Vectors
3. Read the Knowledge Base
4.     Load the Rules
5.     Load the Weaknesses
6. For each Attack Vector
7.     For each Component
8.         Fetch the system attributes
9.         Parse the system attributes with KB-rules
10.        Assess the weaknesses related
11.    For each Weakness
12.        Compute the minimum score components
13.        Weigh the computed score for the weakness
14.        Compute the max weighed score based on CWE
15.    For each Pillar at the CWE research view
16.        Sort weaknesses in order of max weighed score
17.        Write the sorted security alert lists
```

3 Case Study

This case study shows the benefits of AvA approach by using it on a grid middleware system and then verifying the results against the previous FPVA assessment on the same middleware. CrossBroker is a Grid resource management system for interactive and parallel appplications used in various european projects, including Crossgrid [22] the Interactive European Grid [23], and it is being used by the Spanish Grid Initiative. CrossBroker was built by extending the functionality provided in LCG [24] and gLite WMS [25].

3.1 FPVA Applied to CrossBroker

The manual vulnerability assessment following the FPVA guidelines on CrossBroker identified serious and complex vulnerabilities affecting high value assets. A completed and detailed information about them can be found on previous work [15], which is out of scope for this paper. Below, we introduce a summarized description of the CrossBroker FPVA found vulnerabilities:

Vulnerability 1: If CrossBroker is used in an environment where the user can control certain attributes of the jdl submission file, but the executable to run must be selected from a white list of valid executables, then there exists a flaw that allows the user to run arbitrary code as the execute user beyond the white listed executables. ***Cause:*** *Code injection, Improper data validation, Incorrect authorization.* ***Component:*** *submit, network server.*

Vulnerability 2: Certain types of user's job submitted to CrossBroker are not protected from manipulation from other user's jobs. ***Cause:*** *Incorrect privileges, Missing authentication, Multiple unique privilege domains.* ***Component:*** *scheduling agent.*

Vulnerability 3: Remote resources are prone to a hijacking through CrossBroker. If Computing Elements use a firewall/NAT traversal solution

to allow access to grid site elements, attackers will build an independent high throughput computing system without Crossbroker interactions and restrictions. ***Cause:*** *Missing authorization, Misconfiguration.* ***Component:*** *scheduling agent.*

Vulnerability 4: The CrossBroker is prone to a Denial of Service vulnerability. As a result of this attack, Crossbroker will not be able to process the submission of the user jobs, being necessary to stop and restart the Crossbroker host. ***Cause:*** *Improper error handling, Inability to handle missing/invalid field or value.* ***Component:*** *submit, logging and bookkeeping, mysql.*

Up to now, the security practitioners who applied FPVA on several middleware provided their own expertise and creativity about different kind of security problems over the key structural components identified on FPVA artifacts, without further guidance or information. The goal of the following subsection is to demonstrate that the AvA concepts can be used to increase the effectiveness of the FPVA component analysis, with a more comprehensive guidance to automatically indicate where and why the components should be assessed.

3.2 AvA Applied to CrossBroker

The validation consists of applying the AvA assessment process to the CrossBroker component-resource graph (Figure 4), and look for those weaknesses in the security alerts that are related with the vulnerabilities found manually, and those weaknesses that can depict likely new vulnerabilities that were not found initially. CrossBroker has several attack vectors that can be observed from its graph, and for the sake of simplicity we just introduce two different attack vectors with completely different impact surfaces. It is worth to state that the number of attack vectors depends on the number of attack and impact surfaces components we have identified using a graph editor tool.

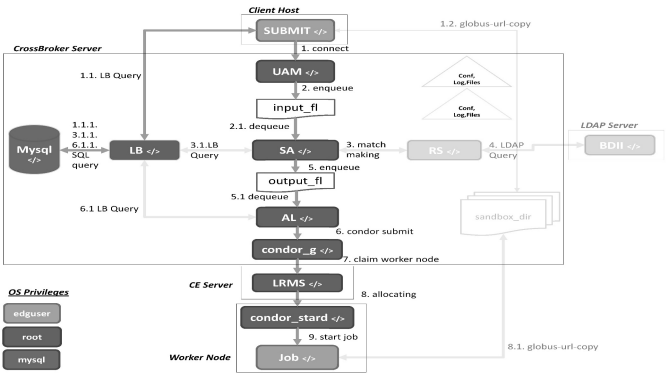


Fig. 4. CrossBroker component-resource graph with attack vectors I-II highlighted

Attack Vector I. The attack vector I is composed by ten components starting with the *submit component* which is the attack surface component in this case, passing through the *network server, input queue, scheduling agent, output queue, application launcher, condor daemon I, local resource manager, condor daemon II*, until achieve its impact surface the *job component*. To visualize the results of asssessing the attack vector I, the AvA security alerts were clustered in accordance to every hierarchical pillar of the CWE research view, and regarding to the max score of the weaknesses and the total number of them belonging to every pillar, as we can see in Figure 5. Therefore, for the "CWE-664" pillar composed of 150 weaknesses, considering the characterization of the attack vector components, and the whole assessment process, we found that around 20 % of weaknesses in the whole pillar can derive into one of three vulnerabilities found with FPVA, which are distributed as follows: 11 weaknesses related to "Vulnerability 3", 17 weaknesses related to "Vulnerability 2", and one weakness related to "Vulnerability 1". In this attack vector there are no weaknesses matches for "Vulnerability 4", due that its underlying cause belong to other middleware components. Itemizing, from the 15 first weaknesses with the highest scores, nine are related to three of the vulnerabilities found manually, as for example the "CWE-862: Missing Authorization" whose description is *"The software does not perform an authorization check when an actor attempts to access a resource or perform an action"*, and reviewing again the CrossBroker vulnerabilities summary along with the CWE-664 pillar description *"The software does not maintain or incorrectly maintains control over a resource throughout its lifetime of creation, use, and release"*. Thus, it can be seen the straight relationship between the weakness-pillar and the "Vulnerability 3" because although the AvA analysis has taken into account those system attributes related to authentication and authorization underlying mechanisms for the attack surface, the attacker is able to handle at will the impact surface.

The AvA analisys corroborated this relationship, due that there is no more control found for the same system attributes on any other attack vector

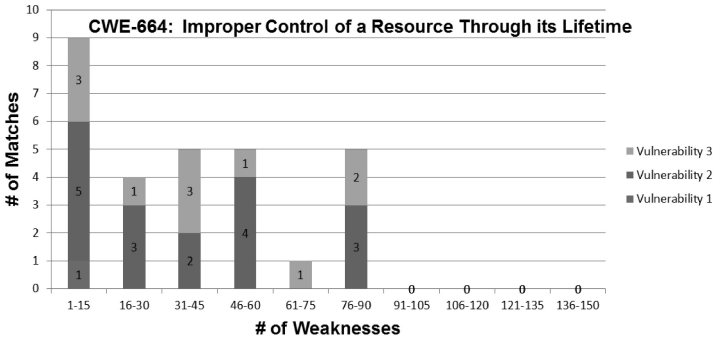


Fig. 5. Security Alerts for attack vector I clustered by CWE-664 pillar

component after a job is submitted in CrossBroker. For some of the remaining CWE pillars their security alerts also hinted related weaknesses to the same vulnerabilities, which gives the attack vector I results consistency, and some others security alerts hinted a likely new vulnerability related to certificate issues.

Attack Vector II. Since the attack vector II has only three components starting with the *submit component* which is again the attack surface component, the *logging and bookkeeping component*, and its impact surface the *mysql component*, then we put together all its AvA security alerts to visualize the results from a global perspective regarding the whole CWE research view, as we can see in Figure 6, instead of an individual pillar visualization as in the attack vector I. By inspecting the security alerts for all the pillars, the assessment of the attack vector II with the AvA analyzer hinted 19 weaknesses, all of them related with the "Vulnerability 4" found with FPVA, which are distributed as follows: 8 weaknesses in "CWE-703 Improper Check or Handling of Exceptional Conditions" , 4 weaknesses in "CWE-664 Improper Control of a Resource Through its Lifetime", 3 weaknesses in "CWE-691 Insufficient Control Flow Management" , 1 weakness in "CWE-710 Coding Standards Violation" , 1 weakness in "CWE-682 Incorrect Calculation" , 1 weakness in "CWE-118 Improper Access of Indexable Resource", 1 weakness in "CWE-435 Interaction Error", and 0 weaknesses for the rest of pillars. Just like in attack vector I for "Vulnerability 4", it happens that there are no weaknesses matching for "Vulnerabilities 1, 2, and 3", due that their underlying causes belong to other middleware components. If we look inside the pillars with hinted weaknesses, we found that the sum of all the issues, such as improper check for unusual or exceptional conditions, unexpected status code or return value, return of wrong status code, missing report of error condition, uncontrolled resource comsuption, improper resource shutdown or release, insufficient control of network message volume, uncontrolled recursion, improper validation, or incorrect control flow scoping, among others are indeed strongly related to the "Vulnerability 4" causes. In summary, to focus the security alerts clearly, we visualized from either individually or globally point of view the hinted weaknesses by the AvA analyzer

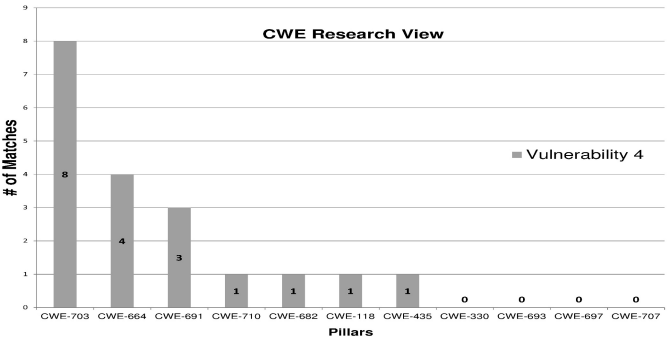


Fig. 6. Security Alerts for attack vector II clustered by entire CWE research view

in both of the attack vectors illustrated, which are strongly related with the vulnerabilities found previously with FPVA.

4 Related Work

Vulnerability Assessment of middleware systems is a field that has attracted the interest of both research and commercial communities, due to the rapid growth of the use of distributed and high performance computing, as well as the increasingly rapid growth of threats. Accordingly, in this section we introduce those vulnerability assessment projects that are most related to the AvA approach, such as the Microsoft Threat Modeling [13], the Open Vulnerability and Assessment Language [26] project, and the vulnerability cause graphs [27].

4.1 Microsoft Threat Modeling

The methodology that has the most in common with the AvA approach is *Microsoft Threat Modeling*. It is aimed at identifying and rating the most likely threats affecting applications, based on understanding their architecture and implementation during the entire development life cycle. While Microsoft's methodology is the closest to AvA approach, there is a key difference: after developing the architectural overview of the application, the Microsoft methodology applies a list of pre-defined and known possible threats, and tries to see if the application is vulnerable to these threats. As a consequence only known vulnerabilities on individual components may be detected, and the vulnerabilities detected may not refer to high value assets. With AvA, the component evaluation is performed only on the critical parts of the system, and we may be able to hint vulnerabilities based on a list of weaknesses, particularly those resulting from the interaction of complex relationships between attack vector components. In addition, Microsoft threats identification suggest a brainstorming with the developers and test teams, these interactions could lead to a biased analysis and may result in threats going undetected.

4.2 The Open Vulnerabilities and Assessment Language (OVAL)

OVAL is an international, information security, community standard to promote open and publicly available security content, and to standardize the transfer of this information across the spectrum of security tools and services. OVAL includes a language used to encode system details, and an assortment of content repositories held throughout the community. The repositories are collections of publicly available and open content that utilize the language. In short it means that OVAL is an open language to express checks for determining whether software vulnerabilities and configuration issues, programs, and patches exist on a system. It is based primarily on known vulnerabilities identified in Common Vulnerabilities and Exposures (CVE) [28], a dictionary of standardized names and descriptions for publicly known information security vulnerabilities and

exposures developed by the MITRE Community and stakeholders. In contrast to OVAL, our effort is neither based on the specific CVE dictionary nor alleged vulnerabilities according to machine states, instead we claim that AvA approach works with CWE and CWSS systems, and with nonspecific software vulnerabilities, also AvA approach is based on FPVA stages, thereby AvA gathers more meaningful information for the assessment process.

4.3 Vulnerability Cause Graphs

It is based on a thorough analysis of vulnerabilities and their causes, similar to root cause analysis. The results are represented as a graph, which Byers et al. [27] called vulnerability cause graph. Vulnerability cause graphs provide the basis for improving software development best practices in a structured manner. The structure of the vulnerability cause graph and the analysis of each individual cause are used to determine which activities need to be present in the software development process in order to prevent specific vulnerabilities. In a vulnerability cause graph, vertices with no successors are known as vulnerabilities, and represent classes of potential vulnerabilities in software being developed. Vertices with successors are known as causes, and represent conditions or events that may lead to vulnerabilities. In our case, the most noticeable difference is that we want to know whether a vulnerability may exist and why, instead Byers' work knows the vulnerabilities and looks for their causes.

5 Conclusions

In this paper we described a novel approach for hinting Grid and Cloud middleware vulnerabilities. The proposed methodology was implemented in a prototype tool, called the Attack Vector Analyzer (AvA). The absence of a formal method that attempts to systematically use the information gathered by the First Principles Vulnerability Assessment (FPVA) methodology, and the knowledge found on the Common Weakness Enumeration (CWE), motivated us to develop the AvA prototype, which validates the AvA approach. AvA demonstrates effectiveness for filling the gap between different steps of the FPVA methodology; and provides significant guidance for security practitioners to determine which middleware components should be assessed and why. We corroborated AvA's efectiveness with the assessment of the Grid middleware CrossBroker. In order to get comprehensive and accurate results, the CrossBroker's security alerts produced by AvA were analyzed from two perspectives, the first one, considering them through each CWE pillar, and the second one, considering them through the whole CWE research view. Thus, it was possible to correlate previous vulnerabilities found manually with several attack vector weaknesses, and automatically identify the most likely vulnerabilities.

AvA will positively impact security practitioners empirical research during the source code inspection, and consequently its quality and accuracy of

vulnerability assessment. Our methodology approach has produced several key accomplishments that distinguish it from formal related vulnerability assessment works. A list of our accomplishments include:

- Our assessment methodology has the important characteristic that it focuses on complex interrelationships among component, and not only on single components.
- The development of a well defined knowledge base based on rules, which allows to match system attributes into multiple weaknesses, and quantifying attack vector weaknesses according to complex component interrelationships.
- A systematic guidance provided for the last FPVA analysis stage, automated by a software tool. The AvA's results provide significant guidance to security practitioner. These results are not sensitive to source code analysis, which makes results language independent.

Future work around AvA approach will involve considering the use of machine learning techniques for improvements of the whole knowledge base architecture, where rules could change in function of new acquired data at middleware runtime.

References

1. Sommestad, T., Ericsson, G.N., Nordlander, J.: Scada system cyber security - a comparison of standards. In: Power and Energy Society General Meeting IEEE, pp. 1–8 (July 2010)
2. Coverity Prevent, `http://www.coverity.com`
3. Fortify Source Code Analyzer, `http://www.fortify.com`
4. Kupsch, J., Miller, B.: Manual vs. automated vulnerability assessment: A case study. In: International Workshop on Managing Insider Security Threats, vol. 469, pp. 83–97 (June 2009)
5. Kupsch, J., Miller, B., Heymann, E., Cesar, E.: First principles vulnerability assessment, mist project. tech. rep., UAB & UW (September 2009)
6. Condor Project, `http://www.cs.wisc.edu/condor`
7. Storage Resource Broker, `http://www.sdsc.edu/srb/`
8. Fernandez del Castillo, E.: Scheduling for Interactive and Parallel Applications on Grid. PhD thesis, Universitat Autònoma de Barcelona (2008)
9. MIST Group: Middleware security and testing web site, `http://www.cs.wisc.edu/mist`
10. The Common Weakness Enumeration, `http://cwe.mitre.org/`
11. McGraw, G., Tsipenyuk, K., Chess, B.: Seven pernicious kingdoms: A taxonomy of software security errors. IEEE Security and Privacy 3, 81–84 (2005)
12. The open web application security project (owasp), `https://www.owasp.org/`
13. Swiderski, F., Snyder, W.: Threat Modeling. Microsoft Press (2004)
14. The Common Weakness Scoring System, `http://cwe.mitre.org/cwss/`
15. Serrano Latorre, J.D., Heymann, E., Cesar, E.: Manual vs automated vulnerability assessment on grid middleware. III Congreso Espanol de Informatica (CEDI 2010) (September 2010)

16. Serrano Latorre, J.D., Heymann, E., Cesar, E.: Developing new automatic vulnerability strategies for hpc systems. In: Latinamerican Conference on High Performance Computing (CLCAR), pp. 166–173 (August 2010)
17. MyProxy, http://grid.ncsa.illinois.edu/myproxy
18. gLExec - Gluing grid computing jobs to the Unix world, https://www.nikhef.nl/
19. The virtual organization membership service (voms), http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html
20. Serrano Latorre, J.D., Heymann, E., Cesar, E., Miller, B.: Vulnerability assessment enhancement for middleware. In: 5th Iberian Grid Infrastructure Conference (IBERGRID) (June 2011)
21. The GraphML File Format, http://graphml.graphdrawing.org/
22. Crossgrid EU Project, http://www.eu-crossgrid.org/
23. Interactive European Grid Project, http://grid.ifca.es/inteugrid_ifca.htm
24. Baud, J.-P.B., Caey, J., Lemaitre, S., Nicholson, C., Smith, D., Stewart, G.: Lcg data management: From edg to egee (2005)
25. Andreetto, P., et al.: Practical approaches to grid workload and resource management in the egee project. In: Proceedings of the International Computing in High Energy and Nuclear Physics, pp. 899–902 (2004)
26. OVAL - Open Vulnerability and Assessment Language, http://oval.mitre.org/
27. Byers, D., Ardi, S., Shahmehri, N., Duma, C.: Modeling software vulnerabilities with vulnerability cause graphs. In: 22nd IEEE International Conference on Software Maintenance (ICSM 2006), pp. 411–422 (2006)
28. The Common Vulnerability and Exposures, http://cve.mitre.org/

VulLocator: Automatically Locating Vulnerable Code in Binary Programs

Yingjun Zhang[1], Kai Chen[2,⋆], and Yifeng Lian[1]

[1] Institute of Software, Chinese Academy of Sciences
[2] Institute of Information Engineering, Chinese Academy of Sciences
chenkai010@gmail.com

Abstract. It usually takes rather long time to generate patches for vulnerabilities. For example, an analysis on 21 recent Microsoft patches shows that it usually takes 115 days on average to generate and release a patch. The longer it takes to generate a patch, the higher the risk a vulnerable system needs to take. In patch generation process, perhaps the core part is to find the vulnerable code in software from zero-day attacks or crash reports. However, this is not easy since there are millions of instructions in an ordinary execution path. In this paper, we present VulLocator, a system that aims at automatically locating vulnerable code in software without requiring any source code. VulLocator could analyze different types of vulnerabilities including stack/heap/integer overflow, double free, memory corruption, format string and division by zero. By generating vulnerability dependence tree, it decreases the number of instructions that need to be analyzed (from millions of instructions to dozens of instructions). VulLocator could also generate a sample patch for temporarily defending against attacks. Analysts could also benefit from the information given by VulLocator to generate more fine-grained patches. Several experiments with real-world exploits are made on VulLocator. The results show that VulLocator could successfully find the vulnerable code in binary programs both effectively and efficiently.

Keywords: Vulnerability locator, Vulnerability dependence tree, Patch generation.

1 Introduction

Vulnerabilities are a big threat to Internet. Attackers can invade into a system using vulnerabilities, especially the unknown vulnerabilities. Worms make use of vulnerabilities to spread and cause damage. For example, the economic damage caused by the Code Red worms is more than $2 billion [1]. The number of software vulnerabilities keeps increasing in recent years. But their patches are released in rather long time. An analysis on 21 recent Microsoft patches (from MS11-087 to MS12-007) shows that it usually takes 115 days on average to

⋆ Corresponding author.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 295–308, 2013.

generate and release a patch (the detailed data is shown in Appendix A). No need to say other small software vendors.

Patch generation is not an easy job. Firstly, software becomes more and more complex. Analysts should analyze millions of instructions carefully. Thus, it is very hard to locate the vulnerable code in software. Secondly, vulnerabilities have different types. Analysts need to be familiar with all of them. Thirdly, since most programs do not open their sources, only developers could make patches. So it is very hard for a third-party to fix vulnerabilities and release patches. These are the reasons why a patch is usually released long after being reported.

People have done much work to solve this problem. 1) Some methods analyze only one type of vulnerability (e.g., buffer overflow) and generate patches for it. AutoPaG [2] and PatchGen [3] could generate patches for previously unknown vulnerabilities if a working out-of-bound exploit is available. But these methods usually patch only one type of vulnerability. 2) Some methods like ClearView [4,5] study from normal executions to learn invariants to patch. But these methods do not try to find the real reason of the vulnerability. 3) Differential slicing [6] could find the difference between two similar traces. It helps analysts to identify causal execution differences for security applications. But it needs a normal execution trace to compare. Sometimes, it is not easy to generate such a trace. An arbitrary trace will have so many differences that it is still difficult for analysts to make patches.

In this paper, we propose a method to automatically locate vulnerable code in binary programs, which reduces the long delay in software patch generation. By generating vulnerability dependence tree, we decrease the number of instructions that need to be analyzed. No source code is needed in this process. Thus analysts from third-party could make patches themselves. Moreover, sample patches could be generated for temporarily defending against attacks. Although they may not work correctly like real patches, we wish the vulnerable code information given by VulLocator could help analysts to generate the real patch. Different types of vulnerabilities are used to make evaluations including stack/heap/integer overflow, double free, memory corruption, format string and division by zero. In summary, we make the following contributions.

- We propose a method to locate the vulnerable code in software without any need for source code. Several types of vulnerabilities (e.g., overflow, double free, memory corruption, format string and division by zero) are supported.
- We propose *vulnerability dependence tree*, which decreases the number of instructions that need to be analyzed.
- We implement a system called VulLocator and make several experiments on it. The results show that VulLocator could automatically locate vulnerable code both effectively and efficiently..

The rest of the paper is organized as follows. We first present related work in Section 2 and compare VulLocator with existing work. Then, we present the overview of VulLocator in Section 3. In Section 4 and Section 5, we give detailed implementation of VulLocator. We make some experiments in Section 6 and make conclusion in Section 7.

2 Relate Work

Our work is mainly related to three bodies of work as follows.

1) Patch Generation. Genetic programming approach is used to generate suitable patches [7,8]. Kranakis [9] and Sidiroglou [10] proposed a framework for patch generation. It places the function that has the problem of stack buffer overflow to heap for better control. But these methods do not find the reason of the vulnerabilities. HOLMES [11] gives scores to paths to indicate the probability of vulnerability and finds the most likely position of the vulnerability. Sweeper [12] detects and patches the vulnerability in software by taint analysis and dynamic backward slicing. ClearView [4] observes normal executions to learn invariants to patch errors in deployed software. PASAN [13] can detect control-hijacking attacks automatically and generate corresponding patches. AutoPaG [2] and PatchGen [3] is able to catch on-the-fly the out-of-bound violation, and then automatically generate patches without any human intervention. These methods usually focus on one type of vulnerabilities while we try to find more types of vulnerabilities. Most methods also need source code to generate patch, which is not suitable for third-party analysts.

2) Software Analysis Method. Differential slicing [6] could help to locate vulnerable code in software by analyzing normal execution trace and the failing execution trace. By comparing them, it could identify causal execution differences for security applications. But a suitable normal execution trace is not easy to get. An arbitrary trace will have so many differences that it is still difficult for analysts to locate vulnerable code. Slicing method is proposed by Mark Weiser, which can analyze the relationship between variables in program. Typical slicing methods include data flow slicing [14] and graph-based slicing [15]. We also slice the program to generate vulnerability dependence tree. Different from traditional slicing methods, we add edge lengths to the tree to show the relevance between instructions and vulnerabilities.

3) Vulnerability Prevention. Some methods instrument the program source code to capture all run-time memory errors such as FOC [16]. TaintCheck [17] performs a dynamic taint analysis so that it can raise an alert when the tainted data is executed. But these methods do not investigate the vulnerability behind the attack, high performance overhead is imposed. VSEF [18] only monitors and instruments those instructions that are related to the exploited vulnerability. Similar approach is also taken by DACODA [19], Vigilante [20], Argos [21]. The aim of this class is to prevent vulnerabilities, but is not to locate vulnerable code, which is the main focus of our system.

3 Overview

Figure 1 shows the overall architecture of VulLocator, which comprises four phases: attack detection, vulnerability dependence tree (VDT) generation, vulnerability code location and sample patch generation.

The attack detection phase contains two steps. First, the program executes by feeding with attack input data. When the program runs, VulLocator records the execution trace. Then VulLocator detects the attacks by using taint analysis or checking whether the program crashes. The last instruction in the trace is referred to as *exploited instruction*.

After an attack is detected, slicing method is used to find the related instructions of vulnerability in the execution trace. Then VulLocator generates vulnerability dependence tree. Different from traditional slicing methods, we add edge lengths to the tree to show the relevance between instructions and the *exploited instruction*.

On the basis of VDT, VulLocator locates vulnerable code by using edge lengths. In this way, not all instructions in the trace need to be analyzed. In practice, this process is combined with the second phase. When VulLocator finds the code may be vulnerable, it stops construing VDT and tries to generate a sample patch.

The last phase is to generate a sample patch. The patch serves as a temporary solution for defending against attacks. It could also be used as suggestion for analysts to generate a real vulnerability.

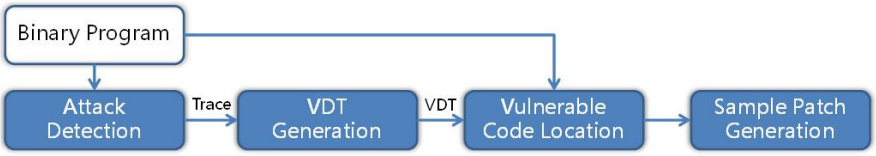


Fig. 1. an Overview of VulLocator

Scope. The aim of the paper is to find the vulnerable code in software without requiring any source code. It decreases the number of instructions which need to be analyzed. In this process, VulLocator generates a sample patch to verify the correctness of the vulnerable code. Note that the sample patch could not replace the real patch. This is reasonable in the real world. In most cases, only software vendors could release official patches.

4 Attack Detection

4.1 Execution Trace Recording

When the program runs, VulLocator records the execution trace of the failing run. Since it is not easy to know the runtime information of indirect index by static analysis, we record the corresponding concrete address of the memory operation instructions with indirect index dynamically. So do indirect jump/call instructions. When we meet branch instructions, we may need lots of computation to get the branch direction. Sometimes, the taken branch cannot be computed if the operand of the branch depends on input. Thus, we record every branch instruction and the arm that it takes.

In real analysis process, we only record the last 100000 instructions currently. This is because vulnerabilities are almost always close to the instructions at which an exploited vulnerability is detected. If we could not find the vulnerable code from the recorded instructions, we can re-run the program and record more instructions. In this way, we do not waste much time on the recording process and save a lot of disk space. When a vulnerability is exploited by using more than one instruction and one of the instructions is after the exploited instruction, our method could still detect first vulnerable instruction before the exploited instruction.

4.2 Exploited Vulnerability Detection

People have done much work to detect or prevent from vulnerability exploits in recent years such as address space layout randomization (ASLR) [22], instruction set randomization [23] and taint analysis [20]. These methods have different pros and cons. In our current implementation, we employ a simple method to detect exploited vulnerability. If any one of the following conditions is met, we think that a vulnerability is exploited: 1) Instruction pointer *eip* is data dependent on input data; 2) Memory index is data dependent on input data; 3) The system default exception handler is triggered. In our current implementation, 'UnhandledExceptionFilter()' is used as the default exception handler. 4) Divisor is zero. Note that condition 3 catches almost all exploited vulnerabilities that crash the program. The last instruction in the trace is referred to as *exploited instruction*.

We instrument instructions on the basis of Pin [24]. It is a dynamic binary instrumentation framework for the IA-32 and x86-64 instruction-set architectures. Taint analysis is used to catch condition 1 and 2. We set file inputs as taint source. We check whether condition 1 and 2 are met by checking the taint status of register *eip* and memory index. Condition 3 can be checked by instrumenting the first instruction of 'UnhandledExceptionFilter()'. Condition 4 can be checked by instrumenting the 'div/idiv' instructions. We could catch different types of vulnerabilities by using different rules. We will also use more sophisticated rules to catch more exploited vulnerabilities in the future.

5 Vulnerable Code Location

We first generate VDT by performing dynamic slicing and taint analysis on the execution trace. After locating the vulnerable code, VulLocator generates a sample patch.

5.1 Vulnerability Dependence Tree Generation

Since there are lots of instructions in the execution trace, it is not easy to find vulnerable code from it directly. We try to choose some instructions that are related to the *exploited instruction*. In this way, analysts could locate the vulnerable code and make patches easily.

Definition 1. [**Vulnerability Dependence Tree**]**:** Vulnerability dependence tree is represented as $VDT(V, E)$. V stands for nodes in VDT and E stands for edges in VDT. For any v_1 and $v_2 \in V$, if there is an edge from v_1 to v_2, then 1) $def(v_2) \in use(v_1)$ and 2) there is no instruction v_3 in the execution trace between v_2 and v_1 which meets $def(v_3) \in use(v_1)$. The edge could be referred to as $e : v_1 \rightarrow v_2$. The root of VDT is the exploited instruction. def means the variables which a statement defines and use means the variables that a statement uses.

Definition 2. [**Edge Length**]**:** For an edge $e : v_1 \rightarrow v_2$, its length indicates the number of branch instructions between v_2 and v_1 in the trace. Those branch instructions should be dependent on input data.

Definition 3. [**Node Depth**]**:** For a node $v \in V$, its node depth is the edge length to the root of VDG.

```
   Input: Execution Trace t
   Output: VDG v
 1 INS i = t.last; // i points to the exploited instruction in t
 2 WorkList wl = ∅;
 3 v.addNote(i); // add the exploited instruction into VDG
 4 wl.add(use(i)); // use(i) consists of the variables that i uses.
 5 int branchNum = 0;
 6 while i ≠ NULL and wl ≠ ∅ do
 7     if i is a branch instruction which depends on input data then
 8         branchNum++; i = i.prev; continue;
 9     end
10     if def(i) ∈ wl then
11         wl.remove(def(i)); wl.add(use(i));
12         INS j = v.node(def(i)); // v.node(var) returns the node that defines var
13         int edgeLen = branchNum - v.len(root, j); // v.len(root,j) returns the
           edge length between nodes root and j
14         v.addNode(i);
15         v.addEdge(j, i, edgeLen); // add new edges to v
16     end
17     i = i.prev; // i points to the previous instruction in t
18 end
```

Algorithm 1. VDG Generation Algorithm

Algorithm 1 shows the pseudo-code to generate a VDG v. It scans the logged trace from exploited instruction (Line 1) and maintains a worklist wl (Line 2). The root of v is the exploited instruction itself (Line 3). At first, wl includes the variables used in exploited instruction (Line 4). We use variable $branchNum$ to record the number of branches that depend on input in the trace (Line 5). Note that we count $branchNum$ in the backward direction (Line 7-9). When a variable var in wl is defined in instruction i (Line 10), we remove var from wl and add use(i) to wl (Line 11). We also add new nodes (Line 14) and edges to v

(Line 15). Since the algorithm scans the trace at most once, it is of order $O(n)$. n is the length of the trace. In practice, we do not need to analyze all instructions in the trace since part of the trace is enough for locating vulnerable code.

The code below show an example. The entry point is line 9.

```
1:  mov  eax, [esi] ;subprocedure    14: je   L20;if 'x', goto L20
2:  cmp  eax, 0     ;copy till 0     15: mov  [edx],eax ;get input
3:  je   L8         ;jump to L8      16: inc  ecx
4:  mov  [edi], eax ;copy string     17: cmp  ecx, 0x10
5:  inc  edi        ;increase des    18: je   L20;at most 0x10
6:  inc  esi        ;increase src    19: jmp  L11;get more chars
7:  jmp  L1         ;jump to L1      20: lea  esi, [ebp-0x18]
8:  ret                              21: lea  edi, [ebp-0x2c]
9:  mov  [ebp-0x8], 0x10             22: call L1 ;copy chars
10: xor  ecx, ecx                    23: mov  eax, [ebp-0x1c]
11: lea  edx, [ebp+ecx-0x18]         24: mov  esi, [eax]
12: call getch      ;getch()         25: lea  edi, [ebp-0x2c]
13: cmp  eax, 0x78 ;0x78 is 'x'      26: call L1
```

Figure 2 shows the logged execution trace and the corresponding VDT for the example. The nodes with two circles are branch nodes. The nodes with single circle are assignment nodes. The number '0x10' in the upper part of Figure 2(a) means that nodes pattern '11, 13, 15' appears 0x10 times. So does the number '0x11'. By using Algorithm 1, VulLocator generates VDT as shown in Figure 2(b). We show the worklist beside each node in the analysis process.

At first, we detect that *eax* is data dependent on input data at statement 24. So we construct the VDT by using node 24 as the root. Then we add *eax* to the worklist according to Algorithm 1 (line 4). We find *eax* is defined in statement 23. Then we remove *eax* from the worklist and add *ebp* and [*ebp*-0x1c] to it (line 11 in Algorithm 1). Since there is no branch node between statement 23 and 24, the edge length between nodes 23 and 24 is 0 (line 13 in Algorithm 1). We add node 23 to the VDT and add length 0 between nodes 23 and 24 (line 14, 15 in Algorithm 1). Then we continue to construct the VDT. Note that the edge length between node 4 and 1 is 1 because *eax* in statement 2 is data dependent on input data. The worklist of node 9 contains 'ebp'. We do not consider stack pointer variables including 'ebp' and 'esp' since they are usually not error-prone. Analysis process finishes at node 9. The numbers beside edges are edge lengths. Not all nodes in VDT are vulnerable. Thus, we need to find the instruction that is most likely to be vulnerable code.

5.2 Vulnerable Code Location and Sample Patch Generation

The nodes in VDT are related to the exploited instructions, but not all of them are vulnerable code. We want to locate vulnerable code from the patching process. Note that the method to patch a vulnerability is not unique. Our basic ideas are as follows. 1) The patched program should not be exploited when fed

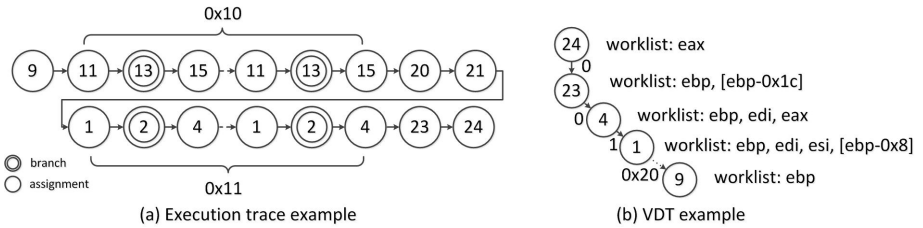


Fig. 2. Execution trace of the example and its corresponding VDT

with the same malicious input. 2) The patched program should run normally when fed with other normal inputs. In other words, the patch will impact the original program as little as possible. According to these basic ideas, we generate some rules to locate the vulnerable code.

R1) Choose the node in VDT with 'smaller' depth. 'Smaller' depth is a value compared with previous node depth. This rule could locate the vulnerability in a small code region. It could also make VulLocator to check the code close to the exploited instruction first. In the VDT, it chooses the nodes closer to the root.

R2) Choose the instruction that changes the value of a variable much more. These instructions will be more likely to let the program run abnormally. In real analysis, one could choose how much a variable is changed. He could use absolute value or relative value, or combine the two values together.

R3) The patched code should execute as little as possible. For example, we try to patch the code in caller procedure instead of callee procedure. In other words, if a procedure p is called by different procedures and p does not always crash by using the same input, we may patch the caller procedure that triggers the crash.

R1 and R2 could help to locate the vulnerable code region, and R3 could help to tune the patch position slightly to find the most suitable one. VulLocator also generates a sample patch for temporarily defending against attacks. Note that the sample patch cannot replace the real patch. Sometimes, a program with the sample patch will also crash. But it could prevent from the attack that exploits the vulnerability. It could also help an analyst to generate a real patch. Since VulLocator aims to handle different types of vulnerabilities, it is not easy to fix all vulnerabilities perfectly. Thus, we use a simple method to avoid the attack. Our basic idea is not to execute the vulnerable code or to make the vulnerable code run normally. Sometimes, the rules are conflictive when we select a vulnerable instruction. For example, one instruction changes the value of a variable much more, but it is far from the root. Another instruction is close to the root, but it only changes the value a little. Our strategy is to check them one by one from the node that is closer to the root. This is simple but effective. For an instruction that is not vulnerable, even if it is removed, the program may still crash. We are also try to keep the original functionality of the program.

Recall the example in Figure 2, node 4 changes '[ebp-0x1c]' (this could be gotten dynamically, in the 0x11th execution of the loop) and its depth is 0.

We choose it as the candidate vulnerable code (R1 and R2). Since node 4 is in procedure L1 and it is called from other instruction (Line 22/26), we set the caller instruction as the vulnerable code (R3). To fix the vulnerability, we use a simple method. We want to change the branch direction at L3. However, we cannot change the code directly since the branch instruction executes many times (0x11 times in the trace in Figure 2(a)) and this will make the program run abnormally. Thus, VulLocator adds code '`mov [ebp-0xc], 0`' (this could be gotten dynamically) before L22. After patching, the program runs 'normally' when fed with both malicious input and normal input. 'Normally' here means no crash. Then it stops the arbitrary code execution. If we want to generate a perfect patch that keeps the program having the exactly same functionality, we still need people's participation.

There is another two problems need to be considered. One problem is that the memory address may change in different execution cases. Thus, we should use relative address. For example, if the memory is in stack, we use the address offset to the stack base. Another problem is that the patch may not be suitable for all malicious inputs. We could test whether the other malicious input might trigger the vulnerability.

6 Evaluation

This section describes the experiments to evaluate VulLocator. We implemented it on the basis of Pin [24] to perform static analysis and dynamic analysis. Our experiments were conducted on a single-processor machine with a dual-core 1.86GHz Intel processor and 2GB of RAM.

6.1 Effectiveness and Performance

In this subsection, VulLocator is used to locate vulnerable code of several programs including Dizzy 1.12, ZipItFast 3, etc. These programs have different types of vulnerabilities such as stack/heap/integer overflow, double free, memory corrupt, format string and division by zero. We use the exploit input from Exploit-DB [25] to evaluate the effectiveness of our method. We detect the exploited vulnerability by using the method shown in Subsection 4.2. Table 1 shows the results.

Column 1 and column 2 show the program name and vulnerability type. From the table, we find that VulLocator could locate vulnerable code of several types. Column 3 shows the number of all instructions in the execution trace. Column 4 shows the number of all memory instructions and branch instructions in the execution trace, which is about half the number of all instructions. It is still too large for people to analyze directly. VDT size is shown in column 5. It is much smaller than the trace size. Analysts could easily get vulnerability information from VDT. Using the generated patch, VulLocator prevents from all the exploits (column 6). We also test whether the patched program could run normally when fed with abnormal input. The results (column 7) show that not all the patched

Table 1. Evaluation Results on Effectiveness

Program	Vul Type	T.Len(A)	T.Len(S)	VDT.S	Pv	C.E	C.N
Dizzy 1.12	Stack Overflow	25360285	11831550	13	√	N	N
ZipItFast 3	Heap Overflow	547164959	210754229	10	√	N	N
MS Paint 5.1	Integer Overflow	66682422	30828159	31	√	N	N
FXSCOVER 5.2	Double Free	162868592	80391484	22	√	N	N
VLC 1.1	Memory Corrupt	214299644	94774038	6	√	Y	N
RadASM 2.2.1.5	Format String	379353073	151896256	19	√	N	N
Zortam MP3 1.5	Division by Zero	1685231785	1205914686	8	√	N	N

T.Len(A): Trace Length (All); T.Len(S): Trace Length (Memory/branch operation); VDT.S: VDT Size; Pv: Prevented; C.E: Crash (abnormal input); C.N: Crash (normal input)

programs work normally (e.g. VLC). We will talk about this in Section 6.2. We also use 20 normal inputs to test whether the patches impact the normal run of those programs. The results show that all the programs run normally (column 8).

We record the time and memory usage in the analysis process including trace logging, VDT generation and vulnerable code locating. Figure 3 shows the time and memory usage. Compared with 115 days on average to generate a windows patch (Appendix A), it is much more efficient. We also measure the performance impact of our generated patches on the programs as a whole. Figure 4 shows the results. Since we use three rules to make the patch impact on original program as little as possible, we find our patch only imposes very little overhead.

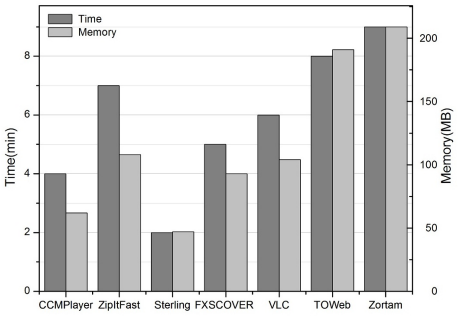


Fig. 3. Execution time and memory usage of VulLocator

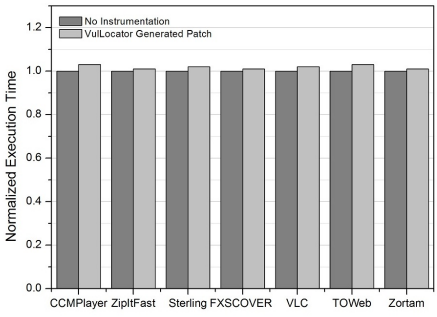


Fig. 4. Performance impact of our generated patch

6.2 Case Study

In this subsection, we use a case to illustrate our method. We choose VLC 1.1 as an example here for two reasons. Firstly, we have used a buffer overflow vulnerability as an example in previous section. So we use another type of vulnerability

here. Secondly, VulLocator does not give a successful patch. We want to know the reason.

As shown in Figure 5, VulLocator generates a VDT. The left part in the figure shows part of the corresponding execution trace of the program. At first, VulLocator finds an error in '0x50C0C6D0'. It adds the first node to the VDT and adds 'edx' to the worklist. By analyzing the trace, it adds another four nodes to the VDT and computes the edge lengths. When it finds the edge length between '0x50C0C3E5' and '0x50C0C262' is 0x37, it tries to locate vulnerable code in the former four nodes. The four nodes have the same depth. VulLocator finds the branch instruction at '0x50C0C4D3' that depends on input data. In this failing trace, eax=0xFFFFFFFF. To fix this problem, VulLocator adds 'xor eax, eax' before this branch instruction. We use the same input again and find that no error happens at '0x50C0C6D0'. However, there is another error in the later execution. VulLocator fails to generate a patch in this case because there are some logic errors in this situation. This problem needs to be analyzed by human manually. We try to use other methods such as AutoPaG [2] to fix this problem, but they also fail. Although VulLocator fails to patch, it points out the vulnerable code in the program. This information could help a lot to fix the error.

Fig. 5. a VLC example

6.3 Discussion

Detection of Vulnerability Exploiting. Currently, we detect whether a vulnerability is exploited by checking the following four conditions. 1) *eip* is data dependent on input data. 2) Memory index is data dependent on input data. 3) System default exception handler is triggered. 4) Divisor is zero. If any one of the four conditions is met, we consider that a vulnerability is exploited. These four conditions can detect most of the vulnerabilities, especially the arbitrary code execution vulnerabilities. However, there are still some types of vulnerabilities that cannot be detected. We could use more complex methods like address space layout randomization (ASLR) [22], instruction set randomization [23] and taint analysis [20] in the future.

False Positive and False Negative of Detector. False positive and false negative may happen in the exploited vulnerability detection phase. For example, if the program itself allows user to change 'eip' or 'arbitrary memory', VulLocator will report a false alarm. In this situation, we can detect attacks by only using condition 3 (exception handler monitoring). False negative may happen when an error happens without exception. We could use more sophisticated detectors to solve this problem.

Correctness of Patch. Since there are many different types of vulnerabilities including logic errors (e.g., VLC), the sample patch generated by VulLocator may not work correctly like real patches. It cannot replace the real patch. However, we wish the vulnerable code information given by VulLocator could help analysts to generate the real patch. We will pay more attention to patch generation process in the future.

7 Conclusion

In this paper, we design and develop a proof-of-concept prototype called VulLocator to find vulnerable code from a working exploit which may be previously unknown. By generating vulnerable dependence tree, VulLocator decreases the number of instructions which need to be analyzed. Furthermore, VulLocator automatically generates a sample patch. The patch could not only be used for temporarily preventing from attacks, but also for analysts to generate more fine-grained patches. VulLocator does not need any source code. It can locate the vulnerable code for different vulnerability types including stack/heap/integer overflow, double free, memory corruption, format string and division by zero. We use some real exploits to make evaluation. The results show that VulLocator could find the vulnerable code both efficiently and effectively. We will test more vulnerability types in the future and we will also try to generate more fine-grained patches.

Acknowledgements. The authors would like to thank the anonymous reviewers for their constructive feedback. This material is based upon work supported in part by the National Natural Science Foundation of China under grant no. 61100226 and the National High Technology Research and Development Program (863 Program) of China under grant no. SQ2013GX02D01211 and 2011AA01A203 and the Natural Science Foundation of Beijing under grant no. 4122085.

A Time-Lines of 21 Recent Microsoft Patches

An analysis on 21 recent Microsoft patches (from MS11-087 to MS12-007) shows that it usually takes 115 days on average to generate and release a patch.

Table 2. The time-lines of 21 recent Microsoft patches

Advisory	CVE#	V.P	P.R	Int	Advisory	CVE#	V.P	P.R	Int
MS12-007	2012-0007	11/09/2011	01/10/2012	62	MS12-006	2011-3389	09/05/2011	01/10/2012	127
MS12-005	2012-0013	11/09/2011	01/10/2012	62	MS12-004	2012-0003	11/09/2011	01/10/2012	62
MS12-004	2012-0004	11/09/2011	01/10/2012	62	MS12-003	2012-0005	11/09/2011	01/10/2012	62
MS12-002	2012-0009	11/09/2011	01/10/2012	62	MS12-001	2012-0001	11/09/2011	01/10/2012	62
MS11-100	2011-3414	09/09/2011	12/29/2011	111	MS11-100	2011-3415	09/09/2011	12/29/2011	111
MS11-100	2011-3416	09/09/2011	12/29/2011	111	MS11-100	2011-3417	09/09/2011	12/29/2011	111
MS11-099	2011-1992	05/09/2011	12/13/2011	217	MS11-099	2011-2019	05/09/2011	12/13/2011	217
MS11-099	2011-3404	09/09/2011	12/13/2011	95	MS11-098	2011-2018	05/09/2011	12/13/2011	217
MS11-097	2011-3408	09/09/2011	12/13/2011	95	MS11-096	2011-3403	09/09/2011	12/13/2011	95
MS11-095	2011-3406	09/09/2011	12/13/2011	95	MS11-094	2011-3396	09/09/2011	12/13/2011	95
MS11-094	2011-3413	09/09/2011	12/13/2011	95	MS11-093	2011-3400	09/09/2011	12/13/2011	95
MS11-092	2011-3401	09/09/2011	12/13/2011	95	MS11-091	2011-1508	03/23/2011	12/13/2011	264
MS11-091	2011-3410	09/09/2011	12/13/2011	95	MS11-091	2011-3411	09/09/2011	12/13/2011	95
MS11-091	2011-3412	09/09/2011	12/13/2011	95	MS11-090	2011-3397	09/09/2011	12/13/2011	95
MS11-089	2011-1983	05/09/2011	12/13/2011	217	MS11-088	2011-2010	05/09/2011	12/13/2011	217
MS11-087	2011-3402	09/09/2011	12/13/2011	95					

V.P: Vulnerability Phrased; P.R: Patch Released; Int: Interval (day)

References

1. CNN: Cost of 'code red' rising (2001), `http://articles.cnn.com/2001-08-08/tech/code.red.II_1_russ-cooper-code-red-ii-internal-networks?_s=PM:TECH`
2. Lin, Z., Jiang, X., Xu, D., Mao, B., Xie, L.: Autopag: towards automated software patch generation with source code root cause identification and repair. In: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, pp. 329–340. ACM (2007)
3. Chen, K., Lian, Y., Zhang, Y.: Automatically generating patch in binary programs using attribute-based taint analysis. Information and Communications Security, 367–382 (2010)
4. Perkins, J., et al.: Automatically patching errors in deployed software. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 87–102. ACM (2009)
5. Weimer, W., Nguyen, T., Le Goues, C., Forrest, S.: Automatically finding patches using genetic programming. In: Proceedings of the 31st International Conference on Software Engineering, pp. 364–374. IEEE Computer Society (2009)
6. Johnson, N., Caballero, J., Chen, K., McCamant, S., Poosankam, P., Reynaud, D., Song, D.: Differential slicing: Identifying causal execution differences for security applications. In: 2011 IEEE Symposium on Security and Privacy (SP), pp. 347–362. IEEE (2011)
7. Forrest, S., Nguyen, T., Weimer, W., Le Goues, C.: A genetic programming approach to automated software repair. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 947–954. ACM (2009)
8. Nguyen, T., Weimer, W., Le Goues, C., Forrest, S.: Using execution paths to evolve software patches. In: Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops, pp. 152–153. IEEE Computer Society (2009)
9. Kranakis, E., Haroutunian, E., Shahbazian, E.: The case for self-healing software. Aspects of Network and Information Security 47 (2008)

10. Sidiroglou, S., Keromytis, A.: Countering network worms through automatic patch generation. IEEE Security & Privacy 3(6), 41–49 (2005)
11. Chilimbi, T., Liblit, B., Mehra, K., Nori, A., Vaswani, K.: Holmes: Effective statistical debugging via efficient path profiling. In: Proceedings of the IEEE 31st International Conference on Software Engineering, pp. 34–44. IEEE Computer Society (2009)
12. Tucek, J., Newsome, J., Lu, S., Huang, C., Xanthos, S., Brumley, D., Zhou, Y., Song, D.: Sweeper: A lightweight end-to-end system for defending against fast worms. ACM SIGOPS Operating Systems Review 41(3), 128 (2007)
13. Smirnov, A., Chiueh, T.: Automatic patch generation for buffer overflow attacks. In: The Third International Symposium on Information Assurance and Security, pp. 165–170 (2007)
14. Weiser, M.: Program slicing. IEEE Transaction on Software Engineering, 352–357 (1984)
15. Ferrante, J., Ottenstein, K.J., Warren, J.D.: The program dependence graph and its use in optimization. ACM Transactions on Programming Languages and Systems (TOPLAS) 9(3), 319–349 (1987)
16. Rinard, M., Cadar, C., Dumitran, D., Roy, D., Leu, T., Beebee Jr., W.: Enhancing server availability and security through failure-oblivious computing. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation, vol. 6, p. 21. USENIX Association (2004)
17. Newsome, J., Song, D.: Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In: Proceedings of the 12th Annual Network and Distributed System Security Symposium (2005)
18. Newsome, J., Brumley, D., Song, D.: Vulnerability-specific execution filtering for exploit prevention on commodity software. In: Proceedings of the 13th Symposium on Network and Distributed System Security (NDSS) (2006)
19. Crandall, J., Su, Z., Wu, S., Chong, F.: On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 235–248. ACM, New York (2005)
20. Costa, M., Crowcroft, J., Castro, M., Rowstron, A., Zhou, L., Zhang, L., Barham, P.: Vigilante: end-to-end containment of internet worms. In: Proceedings of the Twentieth ACM Symposium on Operating Systems Principles, pp. 133–147 (2005)
21. Portokalidis, G., Slowinska, A., Bos, H.: Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. In: Proceedings of the 2006 EuroSys Conference, pp. 15–27 (2006)
22. Baratloo, A., Singh, N., Tsai, T.: Transparent run-time defense against stack smashing attacks. In: Proceedings of the USENIX Annual Technical Conference, pp. 251–262 (2000)
23. Kc, G., Keromytis, A., Prevelakis, V.: Countering code-injection attacks with instruction-set randomization. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 272–280. ACM (2003)
24. Luk, C., et al.: Pin: building customized program analysis tools with dynamic instrumentation. In: Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 190–200. ACM (2005)
25. Exploit DB: Exploit database (2012), `http://www.exploit-db.com`

Software Protection with Obfuscation and Encryption

Vivek Balachandran and Sabu Emmanuel

School of Computer Engineering, Nanyang Technological University
Singapore
vivek2@e.ntu.edu.sg, asemmanuel@ntu.edu.sg

Abstract. Software code released to the user has the risk of reverse engineering attacks. Software obfuscation is one of the techniques used to make the reverse engineering of software programs hard. In this paper, we propose an obfuscation algorithm, which is applied to the assembly code generated by the compiler. Our method uses both obfuscation and encryption, which complement each other thus making reverse engineering harder. The main idea of the algorithm is to hide the control flow information in the data area in encrypted form and removing the control flow instructions from the program. During execution time, these instructions are reconstructed, thereby, preserving the semantics of the program. The stored control flow information is decrypted at runtime and used by self modifying code to reconstruct the control flow instructions. Experimental results indicate that the algorithm performs well against automated attacks.

1 Introduction

The 2010 annual study on software piracy [1] conducted by Business Software Alliance (BSA) and IDC shows that the total loss for the software industry is about 59 billion US dollars. This figure has nearly doubled in real terms from the year 2003. In many of the intellectual property thefts reported, some employ software reverse engineering techniques. In 1992 Atari Games v. Nintendo [2], Sega v. Accolade [6] in 1991, Sony v. Connectix [3] in 2000 and Blizzard v. bnetd [4,5] in 2002 are some of the law suits involving software thefts using reverse engineering.

In addition, reverse engineering a binary program, to a higher level abstraction poses the threat of exposing vulnerabilities of the program. An adversary may exploit it for his advantage. Inserting Trojans, viruses and worms, denial of service are common attacks on programs. The threat by these attacks can be lethal; the recent stuxnet attack [7] is an example. Stuxnet virus was designed to cause damage to Siemens industrial equipment. It caused damage to the machinery at Irans uranium enrichment facility. In this era of computerization, where software plays important role in security and business, security measures against reverse engineering demands high importance.

One of the approaches to make software reverse engineering harder for an attacker is obfuscation. Obfuscation [8,19] is the process of converting a program into a semantically equivalent but hard to understand form. In an ideal case, obfuscation should be so strong that it is easier for an attacker to develop the program from scratch than reverse engineering it.

Recent papers on obfuscation [9-13] concentrate more on obscuring the control flow of binary program. Different concepts, like signals [9], dynamic code mutation [10, 11],

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 309–320, 2013.

control flow flattening [12], are used in attaining binary level control flow obfuscation. Disrupting the control flow of the program makes it harder for an adversary to understand the logical flow of the program code. The basic idea is to remove the control flow instructions from the code and store it in a different module, signal handler, stack or function, and during runtime this module is invoked and control flow is re-established. An attacker can retrieve the control flow addresses stored in the extra modules and custom script his own automated de-obfuscation method in conjunction with reverse engineering tools like IDAPro. We address this problem in our method by encrypting the stored control flow addresses.

In obfuscation based on signals described in [9], trap instructions are used to replace control flow instructions like *call, jump* and *return* in the binary program. During execution these trap instructions raises a signal triggering the programmed signal handler, which directs the system control to the original target address. One of the disadvantages of this method is that an adversary can find the control flow by analyzing the signal handlers code section. Protecting the new module, the signal handler, is also a concern for this method.

In [10], a self modifying code based algorithm is proposed.The control flow instructions like *jmp* are replaced with normal instructions like *mov*. Obfuscation algorithm inserts self modifying code in the program which converts the *mov* instruction back to *jmp* at runtime. A disadvantage of this method is that the target address of the jmp instruction is stored in the destination field of mov instruction which is visible when the code is disassembled.

In the obfuscation method based on dynamically mutating code proposed in [11], parts of procedures are removed and a stub is placed at the entry point of the procedure. The stub invokes an editing engine which inserts the missing instructions during runtime. The extra module, the editing engine, is a disadvantage as it draws the attention of an attacker. This module has the information needed to de-obfuscate the program and hence extra protection mechanism should be implemented to protect the editing engine.

Control flow flattening is another control obfuscation method [12] to confuse the disassembler about the execution sequence of the procedure. The idea is that, all the basic blocks will be assigned with the same predecessor and successor block. Performance overhead in terms of space and time is high for this method. Instruction disassembly error is also less for control flow flattening.

In [16], a method which combines obfuscation and encryption is discussed. The basic method is to replace the control flow jumps with a function call to another procedure (M-process). M-process acts as a jump table and is encrypted in parts and decrypted on demand. The advantage of this method is that only a small part of the M-process will be open at a time. Given the knowledge of the algorithm, an adversary can decrypt the jump table procedure statically. The encryption keys are stored in M-process and the part which is open has the key to the next encrypted part. Thus with the knowledge of the algorithm an attacker can decrypt the M-process statically.

In this paper, we discuss a new binary obfuscation technique, which uses obfuscation and encryption hand in hand to provide better security to the binary programs. We use dynamic code mutation to attain control flow obfuscation. The control flow instructions of the program are replaced by ordinary instructions and the control flow addresses are

stored in data area. The obfuscated program is corrected dynamically at run time with the help of control flow addresses stored in data area. Encryption is used to protect the stored control flow addresses. The control flow addresses are decrypted on demand and is re-encrypted after use, by random keys generated dynamically during runtime. Because of the obfuscations, unreachable code region will look like valid code area to the attacker. Junk decrypt functions with wrong keys are inserted here, thereby confusing an attacker who is trying to find a key.

The paper is organized as follows. Our algorithm is discussed in detail in section 2. Section 3 discusses the experimental evaluation of our method, discussing the performace of our algorithm against automated attacks.

2 Proposed Method

Our algorithm takes an assembly program as input and gives the binary program as output. The algorithm has two phases. The first phase is obfuscation. In this phase control flow obfuscation is achieved using self modifying code obfuscation technique. This obfuscation will change the semantics of the program which is corrected dynamically by self modifying code. The second phase of the algorithm is encryption, which encrypt the information required for self modification of the program at runtime. In the first pass we analyze the program to find the suitable instructions for obfuscation. Obfuscation, the first phase of our algorithm, as shown in algorithm 1, is described in detail in the following subsection:

Algorithm 1. Obfuscation pass-1

Require: Assembly program as input
Ensure: Control flow obfuscation

```
D ← New Data Area
Func_List ← Find User Defined Functions
for each Function in Func_List do
    Jmp_List ← Find Jump Instructions to be Obfuscated
    for each Jump in Jump_List do
        Tgt_Add ← Extract Target Address
        Store Target Address in D
        S ← Size of Jump Instruction
        N ← Normal Instruction of Size S
        Replace Jump Instruction with N
        Add Re − construction Instructions
        Add Re − obfuscation Instructions
    end for
end for
```

2.1 Obfuscation

Create New Data Area. The first step is to create a new data area to store the target address of the jump instructions that are going to be obfuscated. The assembly code for

this has to be added to the input program, during obfuscation time. This data area is also used to store the encryption keys, used to encrypt and decrypt the target addresses.

Find Functions to Obfuscate. Once the data area is created the algorithm will scan the assembly program to find suitable functions for obfuscations. A program will have a set of system procedures and external library calls associated with it. There is no point in obfuscating these functions as an attacker can always get the unobfuscated binary version of these functions. Hence, in our implementation we consider the user defined functions as our targeted functions for obfuscation.

Finding Instructions to Camouflage. An important step of the algorithm is to identify the instructions that have to be camouflaged. The trivial method is, randomly picking instructions from the code area. But, in our method the jump instructions are chosen to be camouflaged for the following reasons.

Jump instructions decide the control flow of a procedure in the program. By obscuring the jump instructions in the procedure we are thus obfuscating the control flow of the program. Instructions which give information about the control flow of the program will help the adversary to easily understand the logic of the program. Another motivation for considering jump instructions to be camouflaged is the scope it provides for inserting junk bytes in the program as explained in [14].

Storing Target Address to Data Area. Once the jump instruction to be obfuscated is decided, the next step is to extract the target address from the jump instruction. In the data area created, space will be allotted to store the target address. The assembly program is modified in such a manner that the target address is then stored in the data area.

Obfuscating the Jump Instructions. The jump instruction can be obfuscated after storing the target address to the data area. The size of the jump instruction is calculated and the jump instruction is replaced by one or more instructions to fill up the void. The easiest approach is to replace the jump instruction with *nop* no operation, instructions. If the jump instruction is of size 2 bytes then, the jump instruction is replaced by two *nop* instructions. We can use other normal instructions like *mov, add, mul,* etc. The key factor is that the camouflaging instruction(s) should have size equal to or more than of that of the jump instruction. This is to ensure that there are enough bytes in the program that can be modified to re-construct the jump instruction during runtime.

By replacing jump instruction with other normal instruction, the program loses its control flow information. When an automated disassembler tries to disassemble the program, it assumes the control flows just to the next address location after the normal instruction. In figure 1, the camouflaged *jmp* instruction is at address location A1 in basic block B1. The *jmp* instruction is camouflaged into *mov* instruction and the reconstruction instructions are added before the camouflaged instruction

Adding Re-construction Instructions. Camouflaging the instructions in the program as explained in the previous section changes the semantics of the program. Running this program in this form gives erroneous results and most probably crashes the program.

And hence, the program has to be changed back to its original form before it gets executed. In our method we do this dynamically at runtime with the help of self modifying code.

Reconstruction instructions which reconstruct jump instruction at runtime are inserted in a block that precedes the jump instruction. The block in which the reconstruction instructions are added should be a dominator block. Block A is a dominator block to block B if and only if block A precedes block B in all execution paths. The insertion of reconstruction instructions are shown in figure 1.

In the example shown in figure 1, the *jmp* instruction is replaced by *mov* instruction. The opcode of jmp instruction is 0xE9 and that of mov instruction 0xB8. We add an instruction to XOR the address location of mov instruction with 0x00000051. This changes the instruction to *jmp offset 0*. Now the next step is to add the address offset stored in the data area to the instruction. We add an instruction to add the value in the global variable to the instruction address. Now the exact *jmp* instruction is created at the address location of *mov* instruction.

In figure 1, the camouflaged *jmp* instruction is at address location A1 in basic block B1. The *jmp* instruction is camouflaged into *mov* instruction and the reconstruction instructions are added before the camouflaged instruction

Adding Re-obfuscation Instructions. The addition of re-construction instructions makes sure that the obfuscated program is semantically equivalent to the original program. Now, after the execution of the re-construction instructions the camouflaged instruction is in its original form. An adversary tracks the image of the program at regular intervals will be able to find the de-obfuscated instructions.

To address this problem, we introduce the concept of dynamic re-obfuscation. The jump instruction is dynamically camouflaged at runtime. This can be achieved by adding extra re-obfuscation instructions in the succeeding blocks of the jump instruction to camouflage it back to ordinary instructions. Note that, the re-obfuscation instruction should be inserted in all the successor blocks as the execution path is chosen dynamically at runtime.

In the example shown in figure 1, re-obfuscation is done by XOR-ing the *jmp* instruction with 0x00000051 to get the instruction: *mov eax, 0*

According to the control flow of the example in figure 1, the basic block B3 follows after the execution of the jmp instruction. The re-obfuscation instructions for the program are hence added in the beginning of the basic block B3.

Randomization of Re-construction Instructions. Randomization of the reconstruction instructions are used so that the attacker wont be able to infer any knowledge by searching for particular pattern of instructions. In the example shown in the paper we use *xor* and *add*. We can use various combinations of arithmetic and logical operations to achieve the same result. The selection of the set of instructions happens at random during obfuscation time. So, an adversary cannot look at specific instructions alone to sort out obfuscation points.

Randomization of Re-construction Locations. The only condition for the block in which the reconstruction instructions are added is that the block should dominate [17]

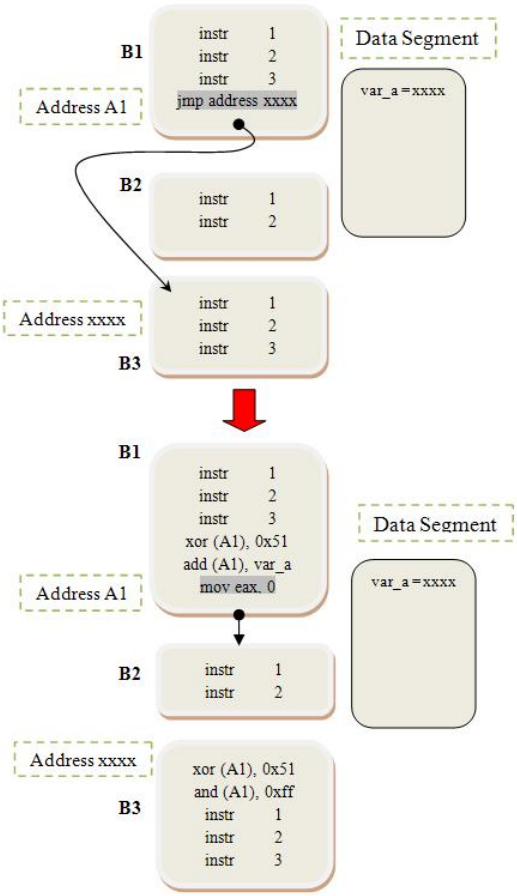


Fig. 1. Obfuscation of jump instruction

the instruction to be camouflaged. The reconstruction instruction can be added in any of the dominators [17] of the camouflaged instruction. The selection of the block from the list of dominators happens at random during obfuscation time.

2.2 Encryption

Encryption is the second phase of our algorithm. In this phase the stored target addresses in the newly created data area will get encrypted. In this pass the target addresses in the newly created data area are encrypted using randomly generated symmetric keys at obfuscation time. The symmetric keys are stored in the data area and decryption and re-encryption functions are added to the assembly program to decipher the target addresses on demand. The output of this pass in the algorithm is the final obfuscated binary program. Algorithm 2, is the algorithm for second pass. The detailed steps of the algorithm are as follows

Algorithm 2. Obfuscation pass-2

Require: Assembly program from pass-1, D- Data area created in pass 1
Ensure: Control flow obfuscation with encryption
for *each* Tgt_Add *in* D **do**
$K \leftarrow Create\ Random\ Key$
Add K *to* D
$Encrypt\ Tgt\_Add\ with\ Key$
$Instr \leftarrow Call\ Decrypt$
$Insert\ Instr\ before\ Camouflaged\ Instruction$
$Instr \leftarrow Call\ ENcrypt$
$Insert\ Instr\ after\ Camouflaged\ Instruction$
$Jmp\_List \leftarrow Find\ Jump\ Instructions\ to\ be\ Obfuscated$
end for
Do Randomization
Assemble to Binary Program

Create and Store Random Symmetric Key. For each target address stored in the newly created data area, we assign one symmetric key. The symmetric key is randomly assigned during obfuscation time. These keys are used to encrypt the target addresses stored in the newly created data area. The keys are then stored along with the encrypted target addresses in the newly created data area. The position of the key and the target address in the data area are randomly assigned.

Encrypting the Target Address. Once the keys are generated, they are used for encrypting the target address in the newly created data area. During obfuscation time, the target addresses get encrypted using the corresponding key. The encryption method we use is XOR. The target address is XOR-ed with the key and is then stored in the data area as shown in Figure 5. The target addresses does not share keys, each target address use a separate key, and hence the strength of the encryption is as good [18]. Another advantage of using XOR is its simplicity.

Adding Decryption Method. With the encrypted target address in the newly created data area if you try to run the program, it will crash. This is because the correct target address is required for reconstructing the camouflaged jump instruction. Therefore the target address should be decrypted to its original values before the reconstruction instructions use it. The instruction to call the decryption function is thus added before the camouflaged instruction as shown in figure 2, during obfuscation time. The decryption function just takes one input parameter, which is the location of key, in the data area.

3.1)Runtime decryption method: We have designed decryption algorithm in such a manner that not much information about the decrypted data is understood by analyzing the decryption function. The input to the decryption function is a location in the data area, where the key to be used for decryption is stored. The decryption algorithm reads the key from the data area. Then, each element in the newly created data area is XOR-ed using the key. Since the decryption and encryption functions are both XOR, this will decrypt the target address. So, the specific target address to which the key was associated gets decrypted. All other target addresses and keys stored in the newly created data area changes due to the XOR-ing.

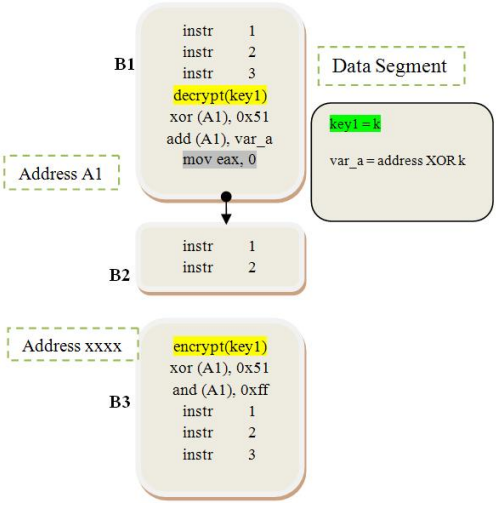


Fig. 2. Assembly program after obfuscation

One of the advantages of this method is that an attacker will not be able to know which specific target address the key is associated to. Another advantage is that during a decryption process every other target address gets modified along with their keys. Since, every element in the data area is XOR-ed; the other keys are also XOR-ed. Hence both the keys and the target addresses are XOR-ed with the current key. This changes both of them in such a way that the new keys can decrypt their target addresses when used by the decryption algorithm. Algorithm for decryption is shown in algorithm 3.

Algorithm 3. Decryption

Require: Key location - K, Data Area - D
Ensure: Decryption of the target address
$KEY \leftarrow D\,[K]$
for $each\ X in\ D$ **do**
$X \leftarrow XxorKEY$
end for

Adding Encryption Method. Call to encryption function is added to the successive blocks of the camouflaged instruction as shown in figure 2, during obfuscation time. During program execution, decryption of the target address happens before the execution of the reconstructed camouflaged jump instruction. At the point of jump instruction, the target address is decrypted and is in the true form. So after the execution of the jump instruction, the re-obfuscation instructions in the successive blocks camouflage the jump instruction again. Just after this the encryption function is called. The encryption function randomly generates a new key and encrypts the target address with the new key, at runtime.

4.1)Runtime encryption method: Encryption method randomly generates a key at runtime. This key is used to encrypt the newly created data area, entirely. In our case, the new key is XOR-ed with every element in the newly created data area. This will change the keys and the target addresses dynamically. Algorithm 4, shows the encryption algorithm.

Algorithm 4. Encryption

Require: Data Area - D
Ensure: Decryption of the target address
$KEY \leftarrow Random\ ()$
for $each\ X in\ D$ **do**
$X \leftarrow XxorKEY$
end for

Randomization in Data Area. We use the newly created data area to store both the keys and the target addresses. There is no specific pattern or location for storage of keys and the target addresses. Completely random locations are assigned for each key and target address. Looking just at the pattern of storage one cannot conclude which is a key and which is target address. Similarly, one cannot find the relationship between a key and a target address by just looking at the data storage pattern.

Junk Decrypt and Encrypt Call Insertions. The argument to the decryption function call is the location to a key stored in the data area. So, if an adversary sees a decrypt call in the program he can infer that the argument used in that call is a location for the key. This is not desirable. Hence, decryption and encryption calls with wrong key locations, are inserted in the program. These insertions will be done at unreachable code area so that it will not affect the semantics of the program.

3 Experimental Evaluation

In this section we evaluate the performance of the proposed algorithm against automated attacks. We tested the potency of our algorithm with IDAPro 6.2 [15], and measured the instruction disassembly error and control flow error caused by the obfuscation. The increase in the size and time complexity due to the addition of self modifying code and encryption/decryption functions are also measured. Microsoft Visual Studio 10.0 is used to generate the assembly programs for obfuscation.

Table 1. Experimental Evaluation

Programs	Instr. disas. error			Control flow error			Time overhead			Space overhead		
	T_{total}	T_{disasm}	CF_{instr}	CFG_{before}	CFG_{after}	CF_{CFG}	T_{before}	T_{after}	$Time_{ovh}$	S_{before}	S_{after}	S_{ovh}
Qsort	199	28	85.9%	11	5	54.4%	310	420	1.35	44	44.5	1.01
Mergesort	397	114	71.3%	29	10	65.5%	770	1100	1.42	74	75	1.01
Huffman Encoding	1904	449	73.6%	191	69	63.9%	1320	1550	1.17	60	66.5	1.10
Gauss Jordan	348	30	91.4%	40	16	60%	890	1090	1.22	67	69	1.02
Mean			78.2%			63.1%			1.26			1.04

3.1 Instruction Disassembly Error

We evaluate the instruction disassembly error with confusion factor. Confusion factor is the fraction of instruction address that the disassembler fails to identify [14]. T_{total} is the total number of actual instruction addresses before obfuscation and T_{disasm} is the total number of instruction addresses properly recognized by the disassembler, then the confusion factor is defined by the following,

$$CF_{instr} = \frac{|T_{total} - T_{disasm}|}{T_{total}} \quad (1)$$

Table 1 shows the instruction confusion factor of assembly programs generated by Microsoft Visual Studio and the obfuscated assembly. The average instruction disassembly error of the test programs is 78.2%. This means that the disassembler succeeds in recovering only 21.8% of the instructions properly, on an average.

3.2 Control Flow Disassembly Errors

We calculated the number of conditional and unconditional jump instructions in the program before and after the obfuscation. If CFG_{before} is the total number of conditional and unconditional jump instructions in the program and CFG_{after} is the total number of jump instructions in the obfuscated program. CF_{cfg} is the confusion factor in the control flow of the program,

$$CF_{cfg} = \frac{|CFG_{before} - CFG_{after}|}{CFG_{before}} \quad (2)$$

The ratio gives the control flow confusion caused by the obfuscation as shown in Table 1.

3.3 Time Overhead

Obfuscation will have effect on the time complexity of the program. With the insertion of new instructions more instructions are computed during runtime. In this section we will discuss the increase in the time complexity due to obfuscation. We evaluate the effect of obfuscation on execution speed with $Time_{ovh}$ defined as,

$$Time_{ovh} = \frac{Time_{after}}{Time_{before}} \quad (3)$$

T_{before} refers to the execution time of the original file and T_{after} refers to that of the obfuscated code. An average time overhead of 1.26 is caused by our algorithm.

3.4 Program Size Overhead

Obfuscation will have effect on the size of the program. $Space_{ovh}$ defines the increase in the size of the program.

$$Space_{ovh} = \frac{Space_{after}}{Space_{before}} \quad (4)$$

The average increase of binary programs after obfuscation is 1.04 times of the original size as shown in Table 1.

3.5 Comparison with Other Algorithms

The performance of our algorithm is compared with three algorithms, namely signal-based obfuscation [9] (SBC), self modifying code based algorithm [10] (SMC) and M-Process based obfuscation [16]. Figure 3, shows the comparison on the basis of instruction disassembly error, control flow error and space and time efficiency of our algorithm with the other two. Our algorithm has better instruction disassembly error, control flow error, space efficiency and comparable time efficiency.

Our method has better instruction disassembly error and control flow error and space efficiency than the other two algorithms. Time overhead of our algorithm is comparable with the other algorithms. We compared our algorithm with M-Process based obfuscation because it is the only other obfuscation algorithm, we found, which uses encryption and obfuscation hand in hand. Our algorithm has a better size and time overhead compared to [16]. The measures of instruction disassembly error and control flow error were not presented in [16] and hence we compare our algorithm with [16] for space and time efficiency.

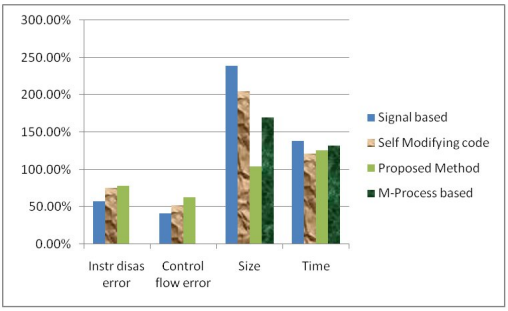


Fig. 3. Comparison with other algorithms

4 Conclusion

In this paper we proposed an algorithm to increase the difficulty in reverse engineering binary programs by combining both software obfuscation and encryption so that it gives better security. The control flow information from the program are removed from the code area and stored in the data area and reconstructed dynamically on demand. The control flow information is stored in an encrypted form in the data area making it hard for an attacker to infer about the control flow by statically analyzing the data area. The concept of adding junk bytes and randomization are used to make the disassembly process harder. The evaluation results show that the proposed method is effective in confusing professional disassemblers like IDAPro better than the competing algorithms. An average instruction disassembly error of 78% and control flow error of 63% are obtained by our method with the overhead in time of 1.26 and space of 1.04.

References

1. BSA Global Software Piracy Study, http://portal.bsa.org/globalpiracy2010/ (last accessed June 12, 2012)
2. Digital Law Online, Reverse Engineering, http://digital-law-online.info/lpdi1.0/treatise25.html (last accessed June 12, 2012)
3. PR Newswire, http://www.prnewswire.com/news-releases/siia-files-six-new-software-piracy-lawsuits-against-fraudulent-online-vendors-across-the-country-69854267.html (last accessed June 12, 2012)
4. Blizzard, www.blizzard.com (last accessed June 12, 2012)
5. Bnetd, Wikipedia, http://en.wikipedia.org/wiki/Bnetd (last accessed June 12, 2012)
6. Digital Law Online, http://digital-law-online.info/cases/24PQ2D1561.html (last accessed June 12, 2012)
7. StuxNet, Wikipedia, http://en.wikipedia.org/wiki/Stuxnet (last accessed June 12, 2012)
8. Collberg, C., Thomborson, C., Low, D.: A taxonomy of obfuscating transformations. Technical Report 148, University of Auckland (1997), http://www.cs.arizona.edu/collberg/Research/Publications/CollbergThomborsonLow97a/LETTER.pdf (last accessed June 12, 2012)
9. Popov, I.V., Debray, S.K., Andrews, G.R.: Binary obfuscation using signals. In: USENIX Security Symposium, pp. 1–16 (2007)
10. Shan, L., Emmanuel, S.: Mobile agent protection with self-modifying code. Journal of Signal Processing Systems 65, 105–116 (2010)
11. Madou, M., Anckaert, B., Moseley, P., Debray, S., De Sutter, B., De Bosschere, K.: Software protection through dynamic code mutation. In: Information Security Applications, pp. 194–206 (2006)
12. Wang, C., Davidson, J., Hill, J., Knight, J.: Protection of software-based survivability mechanisms. In: Dependable Systems and Networks, pp. 193–202 (2001)
13. Balachandran, V., Emmanuel, S.: Software code obfuscation by hiding control flow information in stack. In: IEEE Workshop on Information Forensics and Security (2011)
14. Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassembly. In: ACM Conference on Computer and Communications Security, pp. 290–299 (2003)
15. Hex-Rays, www.hex-rays.com/ (last accessed June 12, 2012)
16. Ge, J., Chaudhari, S.: Control flow based obfuscation. In: ACM Digital Rights Management (2005)
17. Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools, 2nd edn. Addison Wesley (2006)
18. Stallings, W., Brown, L.: Computer Security: Principle and Practice, 2nd edn. Prentice Hall (2007)
19. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: ACM Symposium on Principles of Programming Languages, vol. 25, pp. 184–196 (1998)

Secure Content Delivery in DRM System with Consumer Privacy

Dheerendra Mishra and Sourav Mukhopadhyay

Department of Mathematics
Indian Institute of Technology Kharagpur, India
{dheerendra,sourav}@maths.iitkgp.ernet.in

Abstract. Unauthorized access and illegal content distribution cause huge revenue loss to the rights holders. Digital Rights Management (DRM) is a system, which is developed to prevent illegal content consumption. To ensure authorized content consumption, most of the existing DRM systems lose consumer privacy. However, a scalable DRM system should maintain secure and flexible content distribution which can protect privacy without losing accountability. In this article, we present a secure content distribution mechanism in which involve parties mutually authenticate each other and establish secure session which offers promise for secure content delivery. Moreover, content delivery mechanism does not reveal consumers' preferences and system learns nothing except what must be learned to achieve accountability, which ensures consumers' privacy.

Keywords: Digital rights management, mutual authentication, elliptic curve cryptography, pairing-free identity based cryptosystem, privacy.

1 Introduction

Digital Rights Management (DRM) has emerged as a solution for the digital content copyright protection in the response to the piracy threats. It broadly refers to the set of policies, techniques and tools which manages the access control on the digital contents [1]. In DRM, content is encrypted using standard content encryption techniques and the access of these contents is controlled using the digital license [2]. A remote user achieves the license in the system from the license server through public channel like Internet. A system should adopts a secure and authorized content distribution framework such that only authorized consumer license request should proceeds and can restrict adversary attacks. Moreover, in traditional DRM systems, consumers content consumption information (preferences) revels during the license distribution and content consumption tracking. In general, Preference reveals consumer's habit which he/she do not want to make public. This information can be collected and used for future business or personal benefit. Many papers proposed models and standards for distributing digital content, a few are, Open Mobile Alliance (OMA), Moving Picture Experts Group (MPEG), Internet Streaming Media Alliance (ISMA), Coral, Digital

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 321–335, 2013.

Media Project (DMP), Internet Streaming Media Alliance (ISMA), Advanced Access Control System (AACS), etc., with different approaches, implementations, names and ways to specify the content usage rules. However, DRM is being used to accumulate consumers' tracking habits, surfing habits, personal information, and technical data; frequently use and disclose of this data for secondary purposes without informing to the consumer, elevates serious privacy concern. In general, it is noticed that DRM systems are privacy encroaching and many times violate privacy [3]. The privacy threats discourage the consumer using DRM. The maximization of privacy protection gives significant commercial benefits where these benefits may enhance the public image of DRM system. It leads to increase in the consumer ratio and makes many more individuals comfortable taking part in digital business.

Related Work. The traditional DRM systems are the client-server model, i.e., the system involves single distributor [2, 4–7]. Existing systems may not be sufficient to provide flexible and scalable business models that can handle geographical diversity. However, multi-distributor DRM system provides an efficient and flexible content distribution mechanism that supports existing business models and flexible enough to extend adopting future business model [8]. The multi-distributor DRM architecture has been used in [8–13] which is an alternative to the traditional two party DRM system. It provides the way to implement different strategies in diverse geographical regions. In addition, it achieves the flexibility where user can select the distributor of his choice who offers better services, promotions and discounts. The articles in [11–13] describe secure key management scheme which restricts insider as well as outsider attacks. However, these schemes [8–13] do not offer promise for privacy rights management. The scheme [9] provides the transparent content distribution mechanism while [8, 14] gives rights violation detection mechanism. Schemes [9, 8] protect consumer's privacy during rights violation detection but fail to support during license acquisition process. Thomas et al. [10] provided a secure multimedia content distribution mechanism that hides consumer's preferences during content download but fails to conceal privacy during license acquisition.

Articles [15–21] present privacy preserving digital rights management system. Anonymous-cash-based scheme in [17], content key is issued to the requester without verifying his/her authenticity which provides full anonymity to the consumer during content access. But, anonymous content distribution makes rights violation detection impossible. Scheme [15] and blind decryption mechanics in [17] verify the authenticity of the user. Although in both the schemes, it is most likely that the same key can be acquired by two or more users. This makes the identification of traitor impractical. The schemes [16, 18] issue the license only for authorized user such that user's privacy remain preserved during the license acquisition but both the system does not hide purchaser's total expenses over the digital goods. Win et al. [19] presented a privacy protection mechanism that protects the consumer preference and hides consumer payment information using anonymous payment system [22]. However, tracking of traitor is not possible in this model, as system issues token anonymously to the user and

license server only verifies the authentication of the anonymous token instead of requester. The schemes [20, 21] presents strong privacy protection mechanism which does not violate accountability requirement. Schemes [15–19, 23] provides solution to privacy rights management, although do not present secure content distribution framework. The schemes [24, 25] propose authentication mechanism that ensure authorized and secure communication between DRM principals but do not address privacy issue.

Our Contribution. In this paper, we focus on enhancing the functionality of DRM system by making content distribution flexible and secure. Proposed content distribution mechanism is suitable for more innovative and scalable electronic commerce. We consider a DRM model with multi-distributors instead of single-distributor. A local distributor can better explore potentially unknown market for the owner and make strategies according to the local market requirement.

Proposed system manages transparent and secure content distribution. To achieve transparency, owner separates the content key usages information and key and provides these to independent authorities, respectively license server and distributor. In otherworld, content key usage information (*which key is for which content*) is with the distributor and content key is with license server. To achieve security, involve parties authenticate each other and established a common session key for each session. Further, we present an efficient privacy protection mechanism without violating accountability parameters. Privacy mechanism conceals consumers' preferences from DRM principals such that they do not come to know "which content a user is achieving in the system".

The rest of the paper is organized as follows: In Section 2, we describe the basic framework of DRM system and give brief background of elliptic curve group. Section 3 presents proposed content distribution framework. Section 4 provides the solution to privacy rights management. We analyze our scheme in Section 5. Comparison of our scheme with some existing schemes is presented in Section 6. Finally, in Section 7, we draw a conclusion.

2 Preliminaries

2.1 DRM Framework

Existing DRM systems have different frameworks in terms of implementation, work-flows, usage rules, component, etc. However, most of the systems have the similar architecture. A review of general DRM system is presented in [6]. In general, DRM systems involve four core component: owner, distributor, license server and consumer [1]. Each component is the collection of some servers, logical devices and applications. In DRM, owner holds the rights of digital contents. He encrypts the unprotected contents and provides the protected content with content information to the distributor and content keys with usage rules to the license server. Distributor facilitates the distribution of protected content freely and license server provides the digital license to the authorized consumers.

2.2 Background of Elliptic Curve Group [26, 27]

Let F/F_q denotes an elliptic curve E over a prime finite field F_q, defined by an equation

$$y^2 = x^3 + ax + b, \quad \text{for } a, b \in F_q$$

and with the discriminate

$$\Delta = 4a^3 + 27b^2 \neq 0$$

The points on E/F_q together jointly an extra point Θ called the point at infinity form a group

$$G = \{(x, y) : x, y \in F_q; (x, y) \in E\} \cup \{\Theta\},$$

For a given point $P = (x_P, y_P)$, where x_P is called the X - coordinate of P , and y_P is called the Y - coordinate of P. The group addition operation in G is defined below.

(i) $P + \Theta = \Theta + P = P$ for all $P \in G$.
(ii) If $P = (x_P, y_P) \in G$, then $-P = (x_P, -y_P)$ and $(x_P, y_P) + (x_P, -y_P) = \Theta$.
(iii) If $P = (x_P, y_P) \in G$ and $Q = (x_Q, y_Q)$, where $P \neq Q$, then $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_P - x_Q \pmod p$, $y_3 = \lambda(x_P - x_Q) - y_P \pmod p$ and $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$

The scalar multiplication on the group G is defined as $k \cdot P = P + P + \ldots +$ $P(k$ times). The following problem defined over G is assumed to be intractable within polynomial time. The security of CL-PKC in $e(\cdot, \cdot)$ based on the hardness of following computational problems:

Discrete Logarithm Problem: For a given generator P of G_1 and $Q \in G_1$, find an element $a \in Z_q^*$ such that $aP = Q$.

Computational Diffie-Hellman (CDH) Problem: Let P be a generator of G_1. Given $\langle P, aP, bP \rangle \in G_1$ compute abP for $a, b \in Z_q^*$.

3 Protocol

3.1 Overview of Multi-distributor DRM System

Multi-distributor DRM system involves namely the system owner (O), license server (L), distributors ($D_1, D_2, \cdots, D_n$), and consumer (C). This system works under following assumptions:

- System accommodates multiple distributors where the deployment of distributors depends on the requirement of system and diversity of geographical areas.
- Each entity have public private key pair in the system.

– Involve parties mutual authenticate each other and established a session key.
– Distributor receives the payment from the consumer while license server generates the license over the consumers' request.
– Owner receives royalty from the the distributors and usages license statistic from the license server.

3.2 Content Encryption

Owner appoints the distributors, namely $D_1, D_2, \ldots, D_n$ according to system requirement in the diverse geographical regions to facilitate flexible content distribution. Let the system have r contents, namely $M_1, M_2, M_3, \cdots, M_r$ with contents identities $\mathsf{ID}_{M_1}, \mathsf{ID}_{M_2}, \mathsf{ID}_{M_3}, \ldots, \mathsf{ID}_{M_r}$. Then, the content distribution mechanism is as follows:

- O generates distinct symmetric keys $K_1, K_2, K_3, \ldots, K_r$ and encrypts all the contents $M_1, M_2, M_3, \ldots, M_r$ with unique keys $K_1, K_2, K_3, \ldots, K_r$ by using suitable symmetric key encryption algorithm and gets

$$E_{\mathsf{sym}}(M_i|K_i),\ i = 1, 2, \ldots, r).$$

  O also assigns pseudonym unique number U to all the blind version of the keys, say U_{K_t} is the pseudonym identity of key K_t, where pseudonym identity describe the relationship between key and content identity, i.e., which key is for which content.
- O establishes a secure communication with L and provides the content keys $\{K_i :\ i = 1, 2, \ldots, r\}$ and their name mapping with pseudonym identities $\{(K_i, U_{K_i}) :\ i = 1, 2, \ldots, r\}$.
- O sends the protected contents with their name mapping with pseudonym identity $\{(U_{K_i}, \mathsf{ID}_{M_i});\ i = 1, 2, \ldots, r\}$ to all $D_i,\ i = 1, 2, \ldots, n$.

3.3 Private Key Generation

Private key generater (PKG) creates its set up and generates the partial private keys to license server, distributors and consumers. The process works in the following three steps:

- Set up.
- Private key extraction
- Consumers' registration.

Setup: PKG chooses an arbitrary generator $P \in G$, selects a master key $\mathsf{mk} \in Z_q^*$ and sets $\mathsf{PK} = \mathsf{mk}P$. It chooses hash functions $H_1 : \{0,1\}^* \to Z_q^*$, $H_2 : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^k \to \{0,1\}^n$, and $H : \{0,1\}^* \times \{0,1\}^* \times G^* \times G^* \to \{0,1\}^k$. Then, PKG publishes system parameters $\langle E/F_q, G, k, P, \mathsf{PK}, H_1, H_2, H\rangle$ and Keeps master key mk secret.

Private Key Extraction: L and $\{D_1, D_2, \ldots, D_n\}$ submit their public identities $ID_L, ID_{D_1}, ID_{D_2}, \ldots, ID_{D_n}$ to the PKG, respectively. PKG verifies the proof of identities. If verification succeeds, then generates the partial private keys as:

- Generate $x_L, x_{D_1}, x_{D_2}, \ldots, x_{D_n} \in Z_q^*$.
- Compute $X_L = x_L P$ and $X_{D_i} = x_{D_i} P$, for $i = 1, 2, \ldots, n$.
- Compute $h_L = H_1(ID_L||X_L)$ and $h_{D_i} = H_1(ID_{D_i}||X_{D_i})$, for $i = 1, 2, \ldots, n$.
- By using its master key mk, PKG generates the partial private keys $Y_L = x_L + \mathsf{mk}h_L$ and $Y_{D_i} = x_{D_i} + \mathsf{mk}x_{D_i}$, for $i = 1, 2, \ldots, n$. Then, PKG delivers these partial keys to license server and distributors through a secure channel.

On receiving their partial private keys L and D_i can verify their partial keys as follows:

$$Y_L P = X_L + H_1(ID_L||X_L)\mathsf{PK} \text{ and } Y_{D_i} P = X_{D_i} + H_1(ID_{D_i}||X_{D_i})\mathsf{PK}$$

Consumer's Registration: For the registration, C submits his public identities ID_C to the PKG. Then, PKG verifies the proof of C's identity. If verification succeeds, then include C in its database and generates the partial private key for C as:

- Generate $x_C \in Z_q^*$.
- Compute $X_C = x_C P$, $h_C = H_1(ID_C||X_C)$ and the partial private keys $Y_C = x_C + \mathsf{mk}h_C$. Then, PKG delivers C's partial keys through a secure channel. On receiving the partial private keys, C can verify it as:

$$Y_C P = X_C + H_1(ID_C||X_C)\mathsf{PK}$$

3.4 License Acquisition

Consumer visits distributor's website, selects some content M_t, extract its identity ID_{M_t} and downloads its encrypted file $E_{\mathsf{sym}}(M_t|K_t)$ from the media server. However, the encrypted content can not be played without the valid license. To achieve the license following steps are required:

Step 1. C selects some distributor D and establishes a secure session with D. Once the session is established, C can perform any number of transection to achieve the content. The process is as follows:

- C chooses a random value $c \in Z_q^*$, computes $T_C = cP$, $T'_C = cY_C P$ and sends $\langle ID_C, T_C, T'_C, X_C, t_1 \rangle$ to D, where t_1 is the current date and time of C.
- On receiving the user message, D compute $t_2 - t_1 \leq \triangle t$, where t_2 is the message receiving time of server and $\triangle t$ is the valid time delay in message transmission. If time delay in message transmission is valid, then D selects a random value $d \in Z_q^*$ and gets $T_D = dP$.
- D computes $dT_C = dcP$, $T'_D = dY_D P$ and K_{DC} as:

$$K_{DC} = Y_D[T'_C + d(X_C + H_1(ID_C||X_C)\mathsf{pk})] = cY_D Y_C P + dY_D Y_C P.$$

Then, compute session key sk and and message authentication code mac as:

$$sk = H(ID_C||ID_D||dcP||K_{DC}||t_1||t_3)$$
$$\mathsf{mac} = H_2(ID_C||ID_D||sk||t_1||t_3),$$

where t_3 is the time and date when distributor send the message $\langle ID_D, T_D, T'_D, X_D, \mathsf{mac}, t_3\rangle$ to C.

- On receiving the message, C computes $t_4 - t_3 \leq \triangle t$, where t_4 is the message receiving date and time of consumer's system.
- If time delay in message transaction is valid, then C computes $cT_D = cdP$ and K_{CD} as:

$$K_{CD} = Y_C[T'_D + c(X_D + H_1(ID_D||X_D)\mathsf{pk})] = dY_CY_DP + cY_CY_DP.$$

Then, compute session key sk^* and and message authentication code mac^* as:

$$sk^* = H(ID_C||ID_D||cdP||K_{CD}||t_1||t_3)$$
$$\mathsf{mac}^* = H_2(ID_C||ID_D||sk||t_1||t_3).$$

Then, C checks the condition $\mathsf{mac}^* =? \mathsf{mac}$. If the condition hold, C also send mac^* to D. On receiving the message, D verifies $\mathsf{mac} =? \mathsf{mac}^*$.

Step 2. If mutual authentication succeeds, C encrypts required content identity id_M using the session key sk and submits his/her license request with payment to D. Then, D receives the payment from C and generates a tuple T, where

$$T = (U_{K_t}, \mathsf{ID}_C, E_{\mathsf{sym}}(\mathsf{ID}_{M_t}|\mathsf{sk}), \mathsf{time}).$$

Step3. D establish a secure session with L. Once the session is established D can communicate securely with L. The process is as:

- D chooses a random value $d \in Z_q^*$, computes $T_D = dP$ and sends $\langle ID_D, T_D, T'_D, X_D, t_4\rangle$ to D, where t_4 is the current date and time of D.
- On receiving the user's message, D compute $t_5 - t_4 \leq \triangle t$, where t_5 is the message receiving time of L. If time delay in message transmission is valid, then L selects a random value $l \in Z_q^*$ and gets $T_L = lP$.
- L computes $lT_d = ldP$, $T'_L = lY_LP$ and K_{LD} as:

$$K_{LD} = Y_L[T'_D + l(X_D + H_1(ID_D||X_D)\mathsf{pk})] = dY_LY_DP + lY_LY_DP.$$

Then, compute session sk' and message authentication code mac' as:

$$sk' = H(ID_L||ID_D||ldP||K_{LD}||t_4||t_6)$$
$$\mathsf{mac}' = H_2(ID_L||ID_D||sk'||t_4||t_6),$$

where t_6 is the L current date and time. Then, L sends $\langle ID_L, T_L, T'_L, X_L, \mathsf{mac}, t_6\rangle$ to D.

- On receiving the message, D computes $t_7 - t_6 \leq \triangle t$, where t_7 is time when D receives the message.
- If time delay in message transaction is valid, then D computes $dT_L = dlP$ and K_{DL} as:

$$K_{DL} = Y_D[T'_L + d(X_L + H_1(ID_L||X_L)\mathsf{pk})] = lY_DY_LP + dY_DY_LP.$$

Then, compute sk'^* and mac'^* as:

$$sk'^* = H(ID_L||ID_D||ldP||K_{DL}||t_4||t_6)$$
$$\mathsf{mac}'^* = H_2(ID_L||ID_D||sk'^*||t_4||t_6).$$

Then, D checks the condition $\mathsf{mac}'^* =? \mathsf{mac}'$. If the condition hold, D also send mac'^* to L. On receiving the message, L verifies $\mathsf{mac}' =? \mathsf{mac}'^*$.

Step 4. If verification succeeds, D encrypts T using secret session key sk' and sends encrypted message to L.

Step 5. L decrypts the message using secret session key sk' and gets T. Then, L extracts the key number U_{K_t}, and consumer identity ID_C from T.

Step 6. L generates the license by using key K_t which have identity U_{K_t}. L encrypts C's identity ID_C with session key sk' and associates this encrypted identity with the license. Then, L establishes secure session with C as:

- L chooses a random value $l \in Z_q^*$, computes $T_L = lP$ and sends $\langle ID_L, T_L, T'_L, X_L, t_8\rangle$ to C, where t_8 is the current date and time of L.
- On receiving the user message, C compute $t_9 - t_8 \leq \triangle t$, where t_9 is the time when C receives the message. If time delay in message transmission is valid, then C selects a random value $c \in Z_q^*$ and gets $T_C = cP$.
- C computes $cT_L = clP$, and K_{CL} as

$$K_{CL} = Y_C[T'_L + c(X_L + H_1(ID_L||X_L)\mathsf{pk})] = lY_CY_LP + cY_CY_LP$$

Then, session key sk'' and mac'' will be as follows:

$$sk'' = H(ID_C||ID_L||clP||K_{CL}||t_8||t_{10})$$
$$\mathsf{mac}'' = H_2(ID_C||ID_L||sk''||t_8||t_{10}),$$

where t_{10} is the C's current date and time. Then, C sends $\langle ID_C, T_C, T'_C, X_C, \mathsf{mac}'', t_{10}\rangle$ to L.

- On receiving the message, L computes $t_{11} - t_{10} \leq \triangle t$, where t_{11} is the date and time when L receives the message.
- If time delay in message transaction is valid, then L computes $lT_C = lcP$ and K_{LC} as:

$$K_{LC} = Y_L[T'_C + l(X_C + H_1(ID_C||X_C)\mathsf{pk})] = cY_LY_CP + lY_LY_CP.$$

Then, compute sk''^* and mac''^* as:

$$sk''^* = H(ID_C||ID_L||clP||K_LC||t_8||t_{10})$$
$$\mathsf{mac}''^* = H_2(ID_C||ID_D||sk''^*||t_8||t_{10}).$$

Then, L checks the condition $\mathsf{mac}''^* =? \mathsf{mac}''$. If the condition holds, L also sends mac''^* to C.

– On receiving the message, C verifies $\mathsf{mac}'' =? \mathsf{mac}''^*$.

Step 7. L encrypts the license using session key sk'' and sends the encrypted license to C. On receiving the message, C decrypts the message using the session key sk'' and gets the license.

4 Privacy Rights Management

4.1 Principles

System works under following assumption:

- System does not maintain distinct price to the items. The price of two different categories items may be alike and price of same category item may be distinct.
- Consumer can visit the website and can access the protected content using anonymous IP-address. A consumer is allowed to use anonymous networks such as Tor [28] to hide his originating IP-address in the system.
- An anonymous consumer can achieve the name mapping between the content identity and content encrypted identity from the owner.
- Owner charge a fix amount to provide name mapping information so that malicious user can be restricted from collect the name mapping information after several rounds of request. Owner provides the token of charged amount, which a consumer can use during license acquisition and can get the discount.

4.2 High Level Description

Mechanism of privacy protection is as follows:

- O constructs a collusion free one way function $f(.)$, which is hard to revert. O encrypts the contents identities $\mathsf{ID}_{M_1}, \mathsf{ID}_{M_2}, \mathsf{ID}_{M_3}, \ldots, \mathsf{ID}_{M_r}$ using one way function $f(.)$, and gets $f(\mathsf{ID}_{M_1})$, $f(\mathsf{ID}_{M_2}), \ldots,$ $f(\mathsf{ID}_{M_r})$ and keeps $f(.)$ secret.
- O sends pseudonym key identity corresponding to the blind version of contents identities $f(\mathsf{ID}_{M_t})$, i.e., $\{(U_{K_i}, f(\mathsf{ID}_{M_i}));\ i = 1, 2, \ldots, r\}$ to all $D_i,\ i = 1, 2, \ldots, n$ via secure channel.

Since, contents are identified by their encrypted identities $f(\mathsf{ID}_{M_i}))$, $i = 1, 2, \ldots, r$. To achieve a content key, a name mapping between content identity and content encrypted identity is needed. The name mapping acquisition process is as:

- C visits the distributor's website and selects some content M_t and extracts its identity ID_{M_t}.

- C sends a message to O, which includes ID_{M_t} with the name mapping request.
- O charges a fix amount and encrypts the requested identity ID_{M_t} using secret one way function $f(.)$ and sends the name mapping $(\mathsf{ID}_{M_t}, f(\mathsf{ID}_{M_t}))$ to the anonymous requester.

Once the consumer gets the name mapping information between the content original identity and its encrypted identity, he/she comes to know what request needs to place to the distributor to achieve desire content. As, distributor identifies the content by their encrypted identities instead of original identity, consumer sends license request to the distributor with respect to encrypted content identity $f(\mathsf{ID}_{M_t})$ instead of ID_{M_t}. On receiving the consumer's request, distributor verifies the consumer authenticity, if consumer is authorized then distributor receives the payment and extracts U_{K_t} corresponding to $f(\mathsf{ID}_{M_t})$ from the key name mapping file and generates a tuple T as maintained in step 2 of section 3.4 and sends this tuple T to the license server. On receiving the message, license server generates the license and provides it to the consumer as discussed in the steps 3, 4, 5, 6 and 7 of section 3.4.

5 Analysis

We design our license distribution scheme by keeping in mind the following specific security objectives:

- **Transparent Content Distribution:** Owner assigns the work of payment collection and license distribution to independent authorities, namely distributor and license server respectively. Moreover, by separating the content key and its usages information (*which key is for which content*) such that each of the distributor has the key usage information and license server has the content keys with pseudonym identity. Where, pseudonym key identity does not reveal content key usage information, i.e., for which content encryption key is used. As a result, license server never comes to know *what key is for what content*. Therefore, the license server cannot issue the key without the distributor participation. In addition, owner receives the usages license statistics from the license server and royalties from the distributor. Owner can monitor the royalty flow with the help of usage license statistics.
- **Flexible Content Distribution:** Owner appoints multiple distributors in diverse geographical regions to facilitate content distribution. Proposed mechanism also reduced the congestion on license server by separating the payment authority from license server. Moreover, mechanism provides the freedom to the consumer to select the distributor of his/her own choice that provides easy access of content, promotions and discounts.
- **Contents' Security:** System is relatively secure from insider as well as from outsider attack as follow.
  - *Preventing Insider Attacks:* Digital content should not be exposed to unintentional parties with the help of an insider. In proposed scheme,

content key and its usage information (*which key is for which content*) is separated such that each of the distributor has keys usage information while license server has keys with pseudonym identity. Therefore, to play the content, the participation of license server and a distributor are needed. Only license server cannot issue the key without the distributor participation. As a result, mechanism reduces the possibilities of insider attack.

– *Minimizing Attacks by Outsiders:* To acquire the content key and its usage information, an outsider have to break the security of license server and at least one distributor, which makes system potentially secure against outsider attack.

5.1 Privacy Analysis

Proposed DRM system conceals consumers preferences from all involve principals in the system. Privacy during some key process is justify below:

- **Protected Content Download:** When a user visits the website or downloads the content, he/she can be tracked. In proposed system, user is allowed to use an anonymous connection (anonymous IP-address) during content download. Anonymous connection does not reveal originating IP-address and consumer can achieve anonymity during content download.
- **License Acquisition:** Distributor and license server know *who is requesting* but both do not know *what content consumer is requesting*. Anonymity of content identity hides the consumer's preference because of the following facts:
  – Consumer makes the license request corresponding to content encrypted identity instead of content original identity where the content identity encryption is done by using a strong one-way function which is hard to revert.
  – Distributor verifies the authenticity of user. The authentication process discloses the consumer's identity but the anonymity of license request protects consumer's privacy.
  – Payment does not disclose content identity/category because two different items may have the same price and also two items of different category may have the same price.
  – The license server identifies the decryption key by its pseudonym identity of content which reveals no information about the content original identity.
- **Total Expense:** How much total money a consumer is spending over the contents can be hidden by selecting different distributors for different transaction. Distributors work independently and consumer has the freedom to select any distributor of his choice. A distributor can only know about the payment that a consumer makes to him. In addition, license server does not know the items prices. Hence, by selecting different distributor for different transaction, a user can hide his overall expenses.

5.2 Security Analysis

In this section, we justify that proposed mutual authentication mechanism is secure against following attacks between any two parties i.e., consumer and license server, consumer and distributor, and distributor and license server. Instead of justifying the secure communication between all parties' pairs, we will show that used protocol is secure between any two parities, namely consumer and server, for the rest it will follow the same.

Known-Key Secrecy: If a session key between consumer and server is compromised, which does not mean to compromise of other session keys because every session key evolves arbitrary short-term random values c and s which are selected arbitrary independent for each session by C and S. In addition, every session key involves time stamps which are different for each session.

Replay Attack: Replay Attack is most common attack in authentication process. However, the common countermeasures are time-stamp scheme and random number scheme. In our scheme, we adopt both the mechanism the time-stamp and random number mechanism as a counter-measure. The messages in phase $C \rightarrow S$ and $S \rightarrow C$ are with time-stamp, therefore, replay attack could not work in any phase.

Perfect Forward Secrecy: If the long term keys of two parties compromise, one could compute $sY_CY_SP+cY_SY_CP$. However, an adversary could not compute session key because to compute session key, csP is needed and to compute csP for given $< cP, sP, P >$ is equivalent to Computational Diffie-Hellman (CDH) problem. Moreover, to compute c or s, from cP or sP where P is a primitive element, is equivalent to Discrete Logarithmic Problem on ECC.

PKG Forward Secrecy: If the adversary acquired the PKG's master key mk, it means that the adversary can compute the partial private keys of both consumer and server. Although, it could not be possible to compute the short-term keys with the help of master key, as both C and S randomly generate the short-term keys. Besides, to compute short-term keys c or s, from cP or sP where P is a primitive element, is equivalent to Discrete Logarithmic Problem on ECC. Therefore, an adversary cannot compute session key because to compute session key, one has to compute csP for given $< cP, sP, P >$, which is equivalent to Computational Diffie-Hellman problem.

Man in the Middle Attack: Consumer and server authenticate each other without knowing each other. An adversary or malicious PKG can try man in the middle attack by sending the forge message. However, to authenticate each other, server and consumer exchange mac and $mac*$. Where, to compute mac requires the knowledge of long term private keys (x_C, x_S) and short-term random values (c, s), which are not known to adversary.

Known Session-Specific Temporary Information Attack: If short term secret keys c and s are compromised, then adversary can compute csP. However,

to compute cY_SY_CP and sY_SY_CP for given $\langle c, s, P, sY_SP, cY_CP\rangle$, is equivalent to Computational Diffie-Hellman problem on ECC.

Passive Attack: Adversary can collect the information $\langle P, sP, cP, Y_CP, Y_SP, T_S, T_C\rangle$ which transmits via public channel. However, to compute $cY_SY_CP+sY_SY_CP$ and csP for given information $\langle cY_CP, sY_SP, T_S, T_C, P\rangle$ and $\langle P, sP, cP\rangle$ is equivalent to Computational Diffie-Hellman problem.

6 Comparison

The judgment of the proposed privacy protection scheme with existing privacy rights management DRM systems is illustrated in Table 1. Where, "Y", "N" and "$-$" denote that *system supports*, *system does not support*, and *system does not address* the issue, respectively. We verifies the schemes in the different scenarios where privacy can be threaten like during protected content download, license acquisition, and rights violation detection.

Table 1. Comparison of the proposed scheme with other privacy related works of DRM

Issue	Characteristic	[23]	[8]	[10]	[16]	[19]	[15]	[18]	[24]	[25]	Proposed scheme
Privacy	Content download	Y	N	Y	N	Y	N	–	N	N	Y
	Total expenses	Y	N	N	N	Y	N	–	N	N	Y
	License acquisition	Y	N	N	Y	Y	Y	Y	N	N	Y
Traitor tracing		N	Y	Y	Y	N	N	–	–	–	Y
Mutual authentication		N	N	N	N	N	N	*N*	Y	Y	Y

It is evident from the table 1 that the schemes [15], [18], [19] hide consumer's preferences but do not manage traitor identification mechanism while the schemes [8], [10] are able to detect rights violation in the system but fail to protect consumers' privacy. The scheme [23]and [16] provides privacy without violating accountability parameters, but do not secure content delivery mechanism . The proposed scheme maintains secure communication among the involve parties and achieves all necessary properties of privacy and accountability.

7 Conclusion

Proposed scheme provides a flexible, transparent and secure content distribution mechanism which protects consumer privacy. It provides the flexibility to consumer to select a distributor of his/her choice. It achieves transparency by separating the payment collection and license distribution rights to independent authorities. Proposed scheme also ensure authorized content distribution by using pairing-free identity based mutual authentication mechanism. Moreover, proposed scheme restricts the involve parties to know what exactly a consumer is buying, although it supports accountability parameters.

References

1. Ku, W., Chi, C.: Survey on the technological aspects of digital rights management. Information Security, 391–403 (2004)
2. Rosset, V., Filippin, C., Westphall, C.: A DRM architecture to distribute and protect digital contents using digital licenses. In: Proceeding of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop, AICT/SAPIR/ELETE 2005, pp. 422–427. IEEE (2005)
3. Cohen, J.: DRM and Privacy. Communications of the ACM 46(4), 46–49 (2003)
4. Jamkhedkar, P., Heileman, G.: Digital rights management architectures. Computers & Electrical Engineering 35(2), 376–394 (2009)
5. Zhang, Z.: Digital rights management ecosystem and its usage controls: A survey. JDCTA: International Journal of Digital Content Technology and its Applications 5(3), 255–272 (2011)
6. Liu, Q., Safavi-Naini, R., Sheppard, N.: Digital rights management for content distribution. In: Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003, vol. 21, pp. 49–58. Australian Computer Society, Inc. (2003)
7. Lee, J., Hwang, S., Jeong, S., Yoon, K., Park, C., Ryou, J.: A DRM framework for distributing digital contents through the Internet. ETRI Journal 25(6), 423–436 (2003)
8. Sachan, A., Emmanuel, S., Das, A., Kankanhalli, M.: Privacy preserving multiparty multilevel DRM architecture. In: 6th IEEE Consumer Communications and Networking Conference, CCNC 2009, pp. 1–5. IEEE (2009)
9. Hwang, S., Yoon, K., Jun, K., Lee, K.: Modeling and implementation of digital rights. Journal of Systems and Software 73(3), 533–549 (2004)
10. Thomas, T., Emmanuel, S., Das, A., Kankanhalli, M.: Secure multimedia content delivery with multiparty multilevel DRM architecture. In: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 85–90. ACM (2009)
11. Dutta, R., Mukhopadhyay, S., Dowling, T.: Key management in multi-distributor based DRM system with mobile clients using IBE. In: Second International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2009, pp. 597–602. IEEE (2009)
12. Dutta, R., Mishra, D., Mukhopadhyay, S.: Access policy based key management in multi-level multi-distributor DRM architecture. In: Joye, M., Mukhopadhyay, D., Tunstall, M. (eds.) InfoSecHiComNet 2011. LNCS, vol. 7011, pp. 57–71. Springer, Heidelberg (2011)
13. Dutta, R., Mishra, D., Mukhopadhyay, S.: Vector Space Access Structure and ID Based Distributed DRM Key Management. In: Abraham, A., Mauri, J.L., Buford, J.F., Suzuki, J., Thampi, S.M. (eds.) ACC 2011, Part IV. CCIS, vol. 193, pp. 223–232. Springer, Heidelberg (2011)
14. Sachan, A., Emmanuel, S.: DRM violation detection using consumer logs analysis. In: 2011 IEEE International Conference on Multimedia and Expo, ICME, pp. 1–6. IEEE (2011)
15. Chong, D., Deng, R.: Privacy-enhanced superdistribution of layered content with trusted access control. In: Proceedings of the ACM Workshop on Digital Rights Management, pp. 37–44. ACM (2006)

16. Yao, J., Lee, S., Nam, S.: Privacy preserving DRM solution with content classification and superdistribution. In: 6th IEEE Consumer Communications and Networking Conference, CCNC 2009, pp. 1–5. IEEE (2009)
17. Perlman, R., Kaufman, C., Perlner, R.: Privacy-preserving DRM. In: Proceedings of the 9th Symposium on Identity and Trust on the Internet, pp. 69–83. ACM (2010)
18. Yuan, J., Zhang, W., Zhao, F.: Content key acquisition protocols hiding the usage information in DRM system. In: 2011 IEEE 15th International Symposium on Consumer Electronics, ISCE, pp. 313–317 (2011)
19. Win, L., Thomas, T., Emmanuel, S.: A privacy preserving content distribution mechanism for DRM without trusted third parties. In: 2011 IEEE International Conference on Multimedia and Expo, ICME, pp. 1–6. IEEE (2011)
20. Mishra, D., Mukhopadhyay, S.: Towards a secure, transparent and privacy-preserving DRM system. In: Thampi, S.M., Zomaya, A.Y., Strufe, T., Alcaraz Calero, J.M., Thomas, T. (eds.) SNDS 2012. Communications in Computer and Information Science, vol. 335, pp. 304–313. Springer, Heidelberg (2012)
21. Mishra, D., Mukhopadhyay, S.: Privacy rights management in multiparty multilevel DRM system. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, pp. 625–631. ACM (2012)
22. Tsiounis, Y.: Anonymity & privacy: The internet cash example, Internet document (August 2001), `http://www.internetcash.com/fgo/0,1383,white02,00.html`
23. Win, L., Thomas, T., Emmanuel, S.: Privacy enabled digital rights management without trusted third party assumption. IEEE Transactions on Multimedia 14(3), 546–554 (2012)
24. Fan, K., Pei, Q., Mo, W., Zhao, X., Li, X.: A novel authentication mechanism for improving the creditability of DRM system. In: International Conference on Communication Technology, ICCT 2006, pp. 1–4. IEEE (2006)
25. Yang, Z., Fan, K., Lai, Y.: Trusted computing based mobile DRM authentication scheme. In: Fifth International Conference on Information Assurance and Security, IAS 2009, vol. 1, pp. 7–10. IEEE (2009)
26. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48(177), 203–209 (1987)
27. Hankerson, D., Menezes, A., Vanstone, S.: Guide to elliptic curve cryptography. Springer (2004)
28. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium, vol. 13, p. 21. USENIX Association (2004)

Systematic Construction and Comprehensive Evaluation of Kolmogorov-Smirnov Test Based Side-Channel Distinguishers

Hui Zhao[1], Yongbin Zhou[1,⋆], François-Xavier Standaert[2], and Hailong Zhang[1]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences,
89A, Mingzhuang Rd, Beijing, 100093, P.R. China
{zhaohui,zhouyongbin,zhanghailong}@iie.ac.cn
[2] UCL Crypto Group, Université catholique de Louvain, Belgium
fstandae@uclouvain.be

Abstract. Generic side-channel distinguishers aim at revealing the correct key embedded in cryptographic modules even when few assumptions can be made about their physical leakages. In this context, Kolmogorov-Smirnov Analysis (KSA) and Partial Kolmogorov-Smirnov analysis (PKS) were proposed respectively. Although both KSA and PKS are based on Kolmogorov-Smirnov (KS) test, they really differ a lot from each other in terms of construction strategies. Inspired by this, we construct nine new variants by combining their strategies in a systematic way. Furthermore, we explore the effectiveness and efficiency of all these twelve KS test based distinguishers under various simulated scenarios in a univariate setting within a unified comparison framework, and also investigate how these distinguishers behave in practical scenarios. For these purposes, we perform a series of attacks against both simulated traces and real traces. Success Rate (SR) is used to measure the efficiency of key recovery attacks in our evaluation. Our experimental results not only show how to choose the most suitable KS test based distinguisher in a particular scenario, but also clarify the practical meaning of all these KS test based distinguishers in practice.

Keywords: Side-Channel Analysis, Distinguisher, Kolmogorov-Smirnov Test, Construction, Evaluation.

1 Introduction

Side-channel attack aims at identifying the secret information embedded in a cryptographic device from its physical leakages. One of the most famous side-channel attacks is Differential Power Analysis (DPA), which was proposed by Kocher in his seminal work [1]. Generally, DPA employs some type of statistics (also referred to as *distinguisher*) to reveal the correct key hypothesis about the secret key or part of it within a set of candidates. In side-channel attacks, the

⋆ Corresponding author.

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 336–352, 2013.

most famous two distinguishers known are distance-of-means [1], Pearson correlation coefficient in Correlation Power Analysis (CPA) [3]. Meanwhile, other variants of these two distinguishers, such as Multi-bit DPA [2] and Partitioning Power Analysis (PPA) [4], are also proposed to enhance the performance of DPA and CPA respectively. Concerning these distinguishers, recent works [5,6] has shown that DPA, CPA and even PPA are in fact asymptotically equivalent to each other, given that they are provided with the same a priori information about the leakages. Therefore, these distinguishers are collectively called CPA-like distinguishers. Essentially, all these CPA-like distinguishers exploit linear dependency between key-dependent hypothetical power consumptions and physical leakages.

Even though CPA-like distinguishers are well capable of measuring linear dependency between hypothetical power consumptions and physical leakages, they become less efficient when the dependency is not strictly linear [10]. In light of this, Mutual Information Analysis (MIA) was proposed by Gierlichs in [7] to measure total dependency (both linear and nonlinear) between the hypothetical power consumptions and the physical leakages. Consequently, MIA is considered to be generic because it is capable of dealing with the total dependency. Although MIA is generic, it also bears some technical challenges. For example, the Probability Density Function (PDF) estimation in MIA is widely accepted to be a difficult problem [8,9]. Experiments in [10,11] confirmed that the PDF estimation methods have a decisive impact on the performance of MIA. Therefore, the performance of MIA depends on the accuracy of the estimation methods. Considering the PDF of MIA is hard to estimate accurately, Kolmogorov-Smirnov Analysis (KSA) [10] and Partial Kolmogorov-Smirnov analysis (PKS) [13] were independently proposed. KSA and PKS use Cumulative Density Function (CDF) estimation, instead of PDF estimation, to avoid explicit PDF estimation. Both KSA and PKS sound like promising alternatives for MIA, but which one is a better alternative for MIA in key recovery attacks?

On the one hand, although both KSA and PKS are based on Kolmogorov-Smirnov (KS) test, they differ a lot from each other in terms of construction strategies. One natural yet important question is that whether or not we can construct more efficient distinguishers via combining different construction strategies by both KSA and PKS. For all these KS test based distinguishers, how can we choose the most suitable distinguisher in a certain scenario? For all these KS test based distinguishers, to what extent do they pose severe threats on the implementations of cryptographic modules in practice? In order to answer these questions above, we will investigate the efficiency of all these KS test based distinguishers in a comprehensive comparison framework. Since it seems difficult to study the relationship of all KS test based distinguishers theoretically, we will explore the advantages and limitations of KS test based distinguishers experimentally.

Note that we only compare KS test based distinguishers in a univariate setting, due to the fact that PKS does not have multivariate extensions.

1.1 Our Contributions

The contributions of this paper are threefold. First, we systematically construct nine new variants of KS test based distinguishers via combining different construction strategies by both KSA and PKS. Second, we consider the impacts of leakage function, noise level and power model to twelve KS test based distinguishers and MIA in a comprehensive comparison framework. Experimental results show that how to choose the most suitable distinguisher in a certain scenario. Third, we also demonstrate the practical meaning of all these KS test based distinguishers in practice.

2 Preliminaries

In this section, we will first introduce KS test, and then briefly recall KSA distinguisher and PKS distinguisher.

2.1 Kolmogorov-Smirnov Test

In statistics, KS test is a nonparametric test whose main target is to determine if two distributions differ significantly. Assume that the random variable X has n samples. Its empirical CDF is $F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{A_i \leqslant x}$. $I_{A_i \leqslant x}$ is the indicator function, where its value is 1 when $A_i \leqslant x$; otherwise, it is 0. For a given CDF $F(x)$, formula (1) is used to test their similarity.

$$D_n = sup_x |F_n(x) - F(x)| \tag{1}$$

where sup_x is the supremum of the set of distances. Specifically, the largest distance between two distributions represents the similarity between them. On the other hand, p-value can also be used to measure the similarity of two distributions. The smaller of the p-value, the less similar between them.

2.2 KSA Distinguisher

KSA distinguisher is based on two-sample KS test. Its central idea is to measure the maximum distance between the global trace distribution L and the conditional trace distribution $L|M$, and then average the distances over the prediction space, where M denotes hypothetical power consumption model. Denote l the leakages, and m the hypothetical power consumption values. Denote Pr the probability. KSA is shown in the formula (2).

$$E_{m \epsilon M}(D_{KS}(Pr[L = l|M = m]||Pr[L = l])) \tag{2}$$

KSA can be extended to a normalized version (norm-KSA) that is shown in the formula (3).

$$E_{m \epsilon M}(\frac{1}{|L|M = m|} D_{KS}(Pr[L = l|M = m]||Pr[L = l]))) \tag{3}$$

Both KSA and norm-KSA will produce a large average difference when the key hypothesis is correct.

2.3 PKS Distinguisher

PKS distinguisher is based on single-sample KS test. Its central idea is to measure the p-value produced by comparing normal distribution and part of conditional trace distribution $L|M$. For convenience, leakages L and the hypothetical power consumptions M are usually processed by Z-score transformation in PKS. p is an empirical parameter in PKS from zero to one. N(0,1) represents standard normal distribution. PKS, a two-partial KS test distinguisher, is shown in the formula (6).

$$D_{KS_l} = P_{value}(D_{KS}(Pr[L = l|M \leq p]||N(0,1))) \tag{4}$$

$$D_{KS_r} = P_{value}(D_{KS}(Pr[L = l|M > p]||N(0,1))) \tag{5}$$

$$D_{PKS} = D_{KS_l} \times D_{KS_r} \tag{6}$$

PKS will return the smallest product of p-values when the key hypothesis is correct.

3 Systematic Construction of KS Test Based Side-Channel Distinguishers

From section 2, we learn that both KSA and PKS are based on KS test, and they are able to recover the correct key by partitioning the leakages correctly. However, KSA and PKS are really different from each other in terms of their construction strategies. Therefore, we will show how to construct other new variants of KS test based distinguishers by combining their different construction strategies in a systematic way. For this purpose, we will analyze the construction strategies using by KSA and PKS, and then we will present nine new variants of KS test based distinguishers.

3.1 Construction Strategies of KSA and PKS

In this subsection, we will compare the construction differences between KSA and PKS in four aspects: partition method, similarity measure used by KS test, assumption about leakages, and normalization.

Partition Method. In a partition attack [16], leakages are divided into several sets $p_k^1, p_k^2, ..., p_k^n$ according to each key hypothesis k. These sets are built according to a power model H. In this paper, partition method is classified as non-cumulative partition method and cumulative partition method. Examples of hypothetical leakages that can be used to partition 16-element leakages are shown in Table 1. Specifically, non-cumulative partition used by KSA is shown in the left part of Table 1, while cumulative partition used by PKS is shown in the right part of Table 1.

Similarity Measure Used by KS Test. Distance is used by KSA to measure the similarity of two distributions. In contrast, p-value is adopted in PKS to indicate whether or not partial leakages follow a normal distribution.

Table 1. Examples of non-cumulative partition (left) and cumulative partition (right)

partition	leakages
p_k^1	l_5
p_k^2	$l_2\ l_7\ l_9\ l_{16}$
p_k^3	$l_1\ l_4\ l_8\ l_{10}\ l_{11}\ l_{15}$
p_k^4	$l_3\ l_6\ l_{12}\ l_{13}$
p_k^5	l_{14}

partition	leakages
p_k^1	l_5
p_k^2	$l_5\ l_2\ l_7\ l_9\ l_{16}$
p_k^3	$l_5\ l_2\ l_7\ l_9\ l_{16}\ l_1\ l_4\ l_8\ l_{10}\ l_{11}\ l_{15}$
p_k^4	$l_5\ l_2\ l_7\ l_9\ l_{16}\ l_1\ l_4\ l_8\ l_{10}\ l_{11}\ l_{15}\ l_3\ l_6\ l_{12}\ l_{13}$
p_k^5	$l_5\ l_2\ l_7\ l_9\ l_{16}\ l_1\ l_4\ l_8\ l_{10}\ l_{11}\ l_{15}\ l_3\ l_6\ l_{12}\ l_{13}\ l_{14}$

Assumption about Leakages. PKS distinguisher considers that leakages follow a normal distribution, while KSA makes no assumption about leakages.

Normalization. [10] suggested that normalization could improve the performance of KSA. Our question is whether or not the normalization is always effective in some typical scenarios for KSA. We will also try to answer this question in this work.

3.2 Nine New Variants of KS Test Based Distinguishers

In subsection 3.1, we analyzed the construction strategies of both KSA and PKS. We find that KSA and PKS have different choices for a specific construction strategy. One natural yet pertinent question is that is it possible to construct other (more efficient) KS test based distinguisher by combining the construction methods of both KSA and PKS? To answer this question, we combine construction strategies using by both KSA and PKS to construct nine new variants of KS test based distinguishers, in a systematic way.

For convenience, we will label each strategy that was used by KSA and PKS. Denote A0 the non-cumulative partition, and A1 the cumulative partition. Denote B0 the expectation of distance as the similarity measure of KS test, and B1 the product of p-values as the similarity measure of KS test. Denote C0 the distinguisher that makes no assumption about leakage distribution, and C1 the distinguisher that assumes the leakage follows a normal distribution. Denote D0 that we perform normalization on a distinguisher, and D1 that we do not.

By combining these strategies systematically, one can, in total, construct sixteen ($16 = 2^4$) KS test based distinguishers. Among these sixteen distinguishers, three are existing and they are KSA (A0,B0,C0,D1), PKS (A1,B1,C1,D1) and norm-KSA (A0,B0,C0,D0). On the other hand, note that B1 and D0 conflict with each other, therefore four combinations (A1,C1,B1,D0; A1,C0,B1,D0; A0,C1,B1,D0; A0,C0,B1,D0) do not make any sense. Additionally, three combinations , which are (A0, B0, C1, D1), (A0, B0, C1, D0) and (A0,B1,C1,D1), fail to work in the key recovery attacks. We free the limitation of Z-score on hypothetical power consumptions of D-PKS (A1, B0, C1, D1), norm-D-PKS (A1, B0, C1, D0) and PKS (A1,B1,C1,D1) to form C-PKS (A1, B0, C1, D1), norm-C-PKS (A1, B0, C1, D0) and MPC-PKS (A1, B1, C1, D1). Finally, the remaining combinations are MP-KSA (A0, B1, C0, D1), C-KSA (A1, B0, C0, D1), norm-C-KSA (A1, B0, C0, D0) and MPC-KSA (A1, B1, C0, D1). Therefore, we only

Table 2. Nine new variants of KS test based distinguishers

Distinguisher	Equation
MP-KSA	$log_2(\prod_{m \epsilon M} P_{value}(D_{KS}(Pr[L = l \vert M = m] \vert\vert Pr[L = l])))$
C-KSA	$E_{m \epsilon M}(D_{KS}(Pr[L = l \vert M \leqslant m] \vert\vert Pr[L = l]))$
norm-C-KSA	$E_{m \epsilon M}(\frac{1}{\vert L \vert M = m \vert} D_{KS}(Pr[L = l \vert M \leqslant m] \vert\vert Pr[L = l]))$
MPC-KSA	$log_2(\prod_{m \epsilon M} P_{value}(D_{KS}(Pr[L = l \vert M \leqslant m] \vert\vert Pr[L = l])))$
D-PKS	$E(D_{KS}(Pr[L = l \vert M \leqslant p] \vert\vert N(0, 1)))$
norm-D-PKS	$E(\frac{1}{\vert L \vert M \leqslant p \vert} D_{KS}(Pr[L = l \vert M \leqslant p] \vert\vert N(0, 1)))$
C-PKS	$E_{m \epsilon M}(D_{KS}(Pr[L = l \vert M \leqslant m] \vert\vert N(0, 1)))$
norm-C-PKS	$E_{m \epsilon M}(\frac{1}{\vert L \vert M \leqslant m \vert} D_{KS}(Pr[L = l \vert M \leqslant m] \vert\vert N(0, 1)))$
MPC-PKS	$log_2(\prod_{m \epsilon M} P_{value}(\frac{1}{\vert L \vert M \leqslant m \vert} D_{KS}(Pr[L = l \vert M \leqslant m] \vert\vert N(0, 1))))$

construct nine ($9{=}2^4-4-3-3+3$) new variants. These nine new distinguishers are summarized in Table 2.

In these nine new variants of KS test based distinguishers, the distinguisher which contains B0 strategy will return the largest expected distance under the correct key hypothesis, while the distinguisher which contains B1 strategy will return the smallest product of p-values. Additionally, in order to avoid arithmetic underflow, one typically applies the logarithm to the distinguisher which contains B1 strategy.

4 A Comprehensive Evaluation of All Twelve KS Test Based Side-Channel Distinguishers

So far, we have constructed nine new variants of KS test based distinguishers. The performance of these distinguishers in a univariate setting has a huge impact on how to choose the most suitable distinguisher in a certain scenario. Consequently, we will evaluate the performance of all these distinguishers by amounting key recovery attacks, and analyze their effectiveness and efficiency by using Success Rate (SR) [15] in typical scenarios. On the one hand, we will evaluate the performance of these KS test based distinguishers in a unified comparison framework inspired by [12]. In this framework, we will evaluate the influence of different factors, such as leakage function, noise level and power model, on the

performance of each KS test based distinguisher. We will compare the attacking efficiency of these distinguishers in terms of SR. On the other hand, we will perform a series of attacks against the real traces from both OpenSCA and DPA Contest v2, respectively. With these practical attacks, we will demonstrate the practical meaning of all these KS test based distinguishers. Note that we do not compare the running cost for different distinguishers.

4.1 Simulated Experiments

In simulated scenarios, we choose the output of the first S-box of the first round AES operation as the target intermediate value. Three typical leakage functions, i.e. Hamming Weight (HW) leakage function, an Unevenly Weighted Sum of the Bits (UWSB) leakage function and highly nonlinear leakage function, are adopted to test the adaptability of KS test based distinguishers and MIA. Noise level in simulated leakages is measured by Signal-to-noise ratio (SNR). We particularly employ seven SNRs, i.e. 0.125, 1, 8, 16, 32, 64 and positive infinity, to test the influence of Gaussian noise on these distinguishers.

In each scenario, we perform key recovery attacks using all twelve KS test based distinguishers and MIA (MIA(HW,bins=9) and MIA(ID,bins=256)) as well. For each one of these sixteen kinds of attacks, we repeat it 300 times by uniformly choosing different plaintexts.

Our experiments are also carefully organized in order to make them understood more easily. Specifically, we divide the results of all these thirteen distinguishers into three groups, and denote these groups by A, B and C, respectively. Group A consists of four existing distinguishers and they are PKS, KSA, normKSA and MIA. For each scenario, we select the most efficient one from Group A, and the selected one is set to be a benchmark for this scenario. Next, the other new nine KS test based distinguishers are classified into two groups, according to their relative efficiency over the selected benchmark. Namely, for each scenario, those distinguishers that are more efficient than the benchmark are set into Group B, while the others that are less efficient than the benchmark are put into Group C.

Hamming Weight Leakage

In the following scenarios, we assume that the leakage of a cryptographic device consists of HW of target intermediate value and Gaussian noise. Under this reasonable assumption, we will investigate the performance of different distinguishers with two adversarial characterization abilities.

- An Adversary with a Perfect Power Model. Figure 1 shows the SR of twelve KS test based distinguishers and MIA using a HW model against HW leakage of the first AES S-box. When the SNR is 0.125, PKS(HW,p=0.618) in Figure 1(a) is used as the benchmark for Figure 1(d) and 1(g). Figure 1(d) shows that, C-KSA(HW), MPC-PKS(HW) and C-PKS(HW) are better than the benchmark, and C-KSA(HW) is the best distinguisher. Distinguishers in Figure 1(g) are less efficient than the benchmark, so we do not explain them in more

details. Due to fact that other SNRs can be analyzed in a similar way as that of SNR of 0.125, we do not explain them in more details.

In summary, C-KSA(HW) is the best choice in all twelve KS test based distinguishers when the SNRs are 0.125 and 1 respectively, while MP-KSA(HW) is the best choice when the SNR is 8 and positive infinity. Additionally, MPC-PKS(HW) is better than the benchmark when the SNRs are 0.125, 1 and 8 respectively. Another interesting observation is that norm-KSA(HW) is less efficient than KSA(HW).

- An Adversary with a Generic Power Model. Figure 2 shows the SR of twelve KS test based distinguishers and MIA using an Identity (ID) model against HW leakage of the first AES S-box. When the SNR is 0.125, PKS(ID,p=0.618) in Figure 2(a) is chosen as the benchmark in Figure 2(d) and 2(g). Figure 2(d) shows that C-KSA(ID), norm-C-KSA(ID), MPC-KSA(ID), C-PKS(ID), norm-C-PKS(ID) and MPC-PKS(ID) are more efficient than the benchmark, and they have similar performance. Distinguishers in Figure 2(g) are less efficient than the benchmark, so we do not explain them in more details. When the SNRs are 1, 8, and positive infinity, they can be analyzed in a similar way as that of SNR of 0.125.

In a word, although C-KSA(ID), norm-C-KSA(ID), MPC-KSA(ID), C-PKS(ID), norm-C-PKS(ID) and MPC-PKS(ID) are more efficient than the benchmark and they have similar performance under four noise levels, C-KSA(ID), norm-C-KSA(ID) and MPC-KSA(ID) are slightly more efficient than C-PKS(ID), norm-C-PKS(ID) and MPC-PKS(ID). Another interesting point is that KSA(ID), norm-KSA(ID) and MIA(ID) all fail to reveal the correct key, while both PKS(ID,p=0.25) and PKS(ID,p=0.618) succeeds to do that.

An Unevenly Weighted Sum of the Bits Leakage Scenario

In the following scenarios, we assume that the least significant bit dominates in the leakage function with a relative weight of 10 and other bits with a relative weight of 1, and we will investigate the performance of twelve KS test based distinguishers and MIA with two adversarial characterization abilities.

- An Adversary with an Imprecise Power Model. Figure 3 shows the SR of twelve KS test based distinguishers and MIA using a HW model against UWSB leakage of the first AES S-box. When the SNR is 0.125, PKS(HW,p=0.618) in Figure 3(a) will be chosen as the benchmark for Figure 3(d) and Figure 3(g). Figure 3(d) shows that C-KSA(HW) exhibits consistently better performance compared with the benchmark. Distinguishers in Figure 3(g) are less efficient than the benchmark, so we do not explain them in more details. When the SNRs are 1, 8, and positive infinity, they can be analyzed in a similar way as that of SNR of 0.125.

In summary, C-KSA(HW) is the best choice of all twelve KS test based distinguishers when the SNRs are 0.125, 1 and 8 respectively, while MIA(HW) is the best choice when the SNR goes into positive infinity. Additionally, MPC-KSA(HW) is no worse than the benchmark, and KSA(HW) is more efficient than norm-KSA(HW).

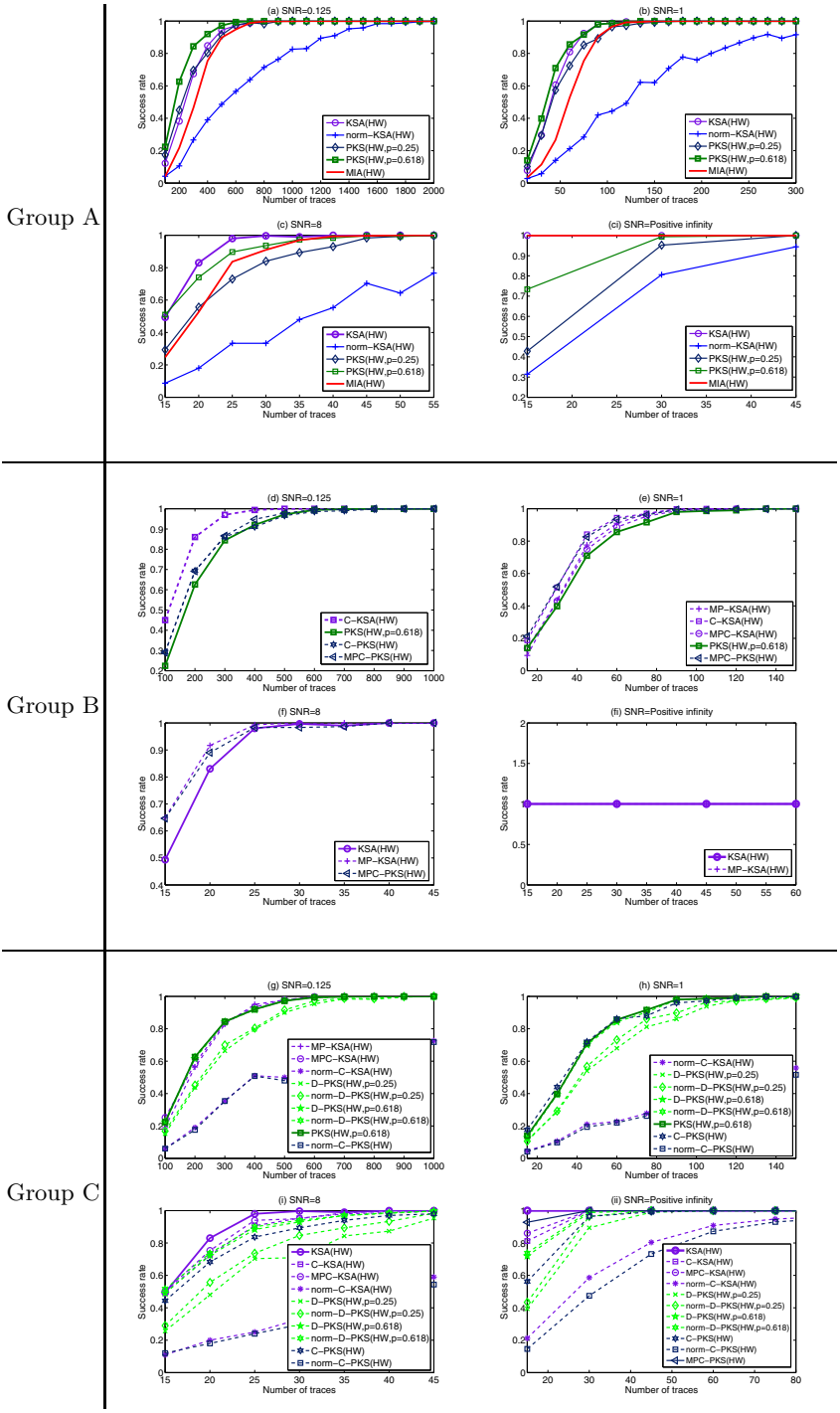


Fig. 1. SRs of different distinguishers against the first AES S-box in HW leakage

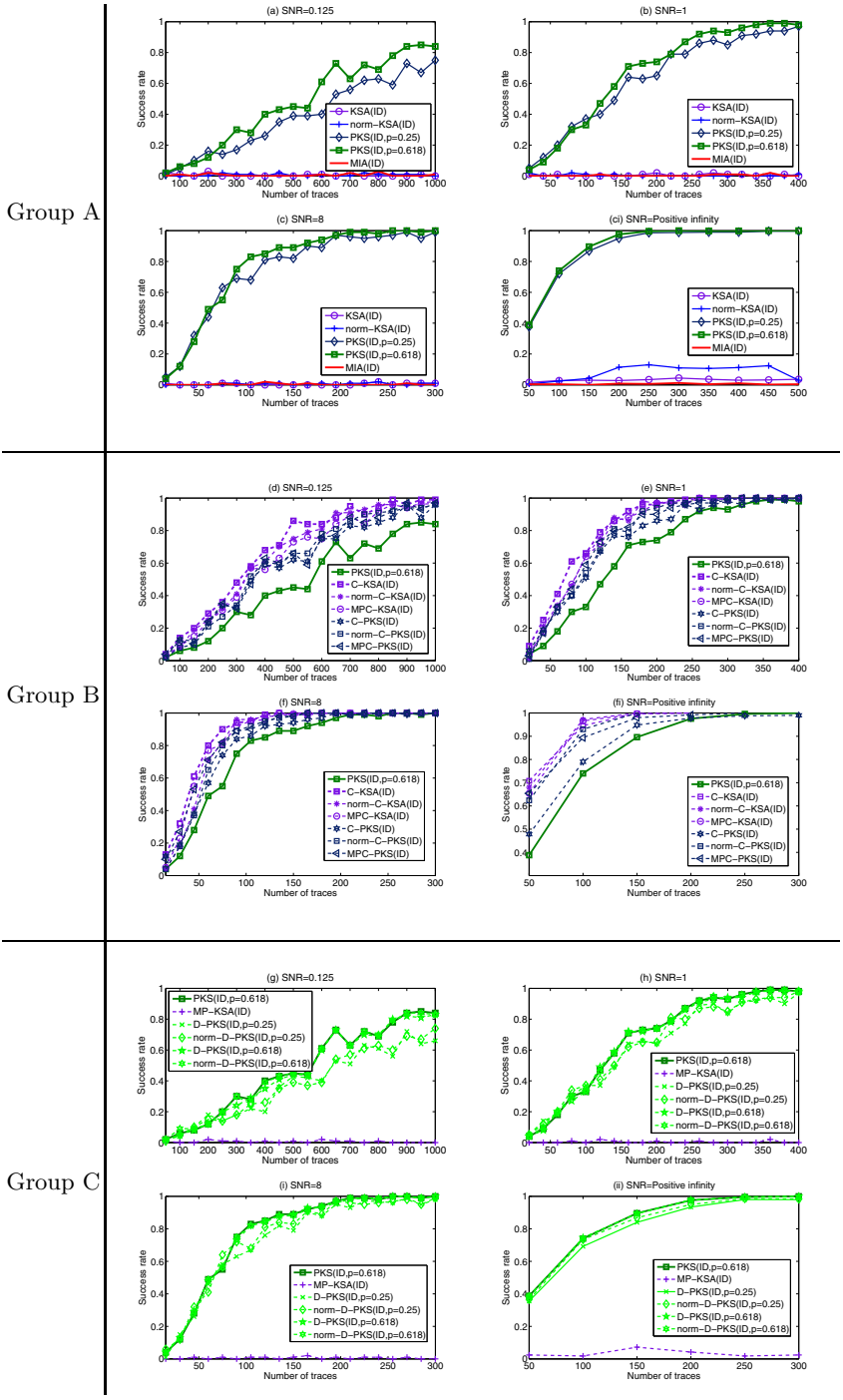


Fig. 2. SRs of different distinguishers against the first AES S-box in HW leakage

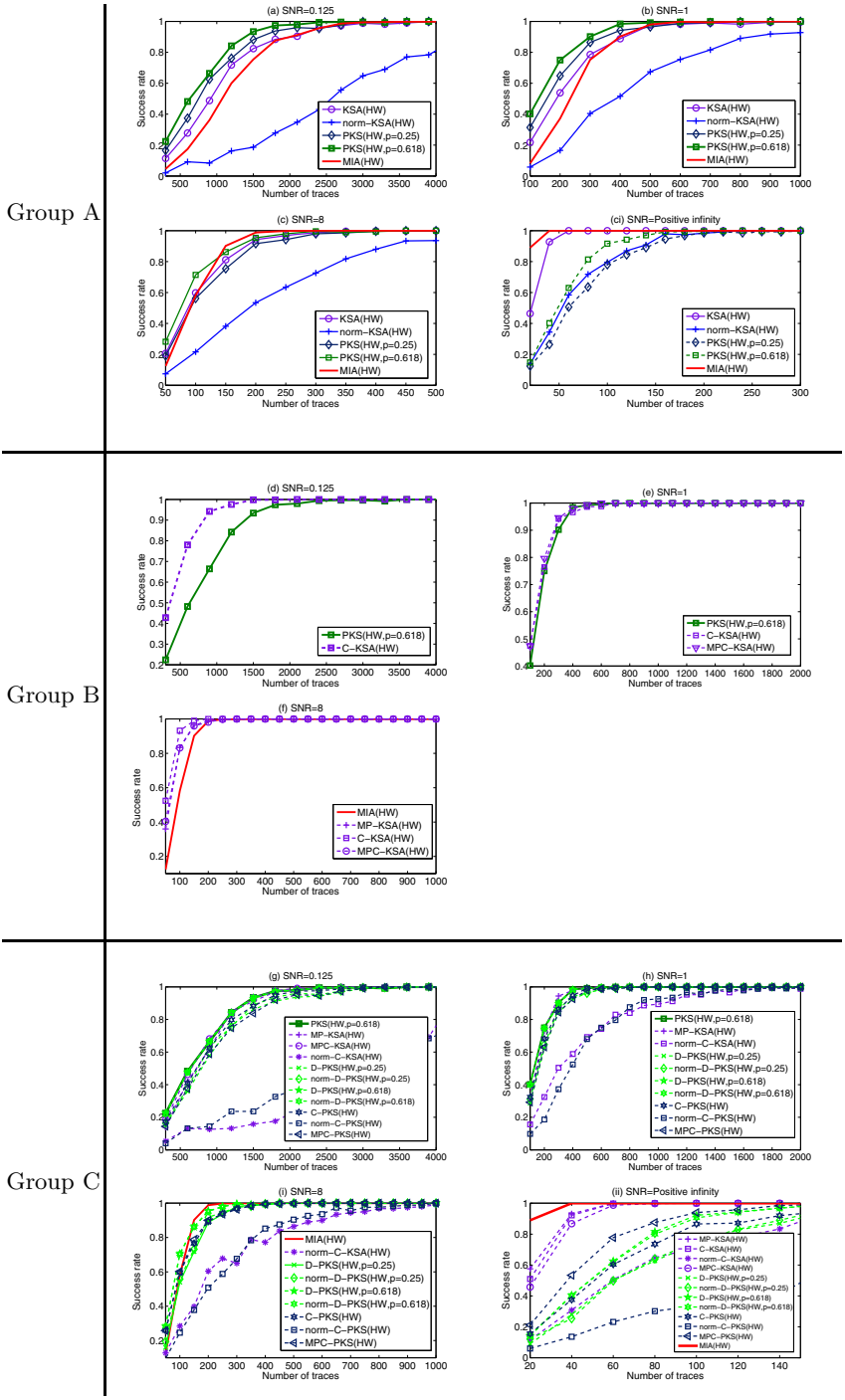


Fig. 3. SRs of different distinguishers against the first AES S-box in UWSB leakage

- An Adversary with a Generic Power Model. Due to the computation cost, we select the SNRs of 16, 32, 64 and positive infinity in this scenario. Figure 4 shows the SR of twelve KS test based distinguishers and MIA using an ID model against UWSB leakage of the first AES S-box. In Group A, Figure 4 shows that, KSA(ID), norm-KSA(ID), MIA(ID) and PKS(ID) all fail to recover the correct key with a relative small number of traces. Therefore, the benchmark for Group B and Group C is whether or not a distinguisher can recovery the correct key with a relative small number of traces. The distinguishers in Group B can recover the correct key with a trace number of 4,000, while the distinguishers in Group C fail to do that. For example, when the SNR is 16, C-KSA(ID), norm-C-KSA(ID) and MPC-KSA(ID) in Group B can recovery the correct key (see Figure 4(d)), while other new variants of KS test based distinguishers fail to do that with 4,000 power traces (see Figure 4(g)). When the SNRs are 32, 64, and positive infinity, they can be analyzed in a similar way as that of SNR of 16.

To sum up, C-KSA(ID), norm-C-KSA(ID) and MPC-KSA(ID) are more efficient than the benchmark, and C-KSA(ID) is the best choice when the SNRs are 16, 32, 64 and positive infinity.

Highly Nonlinear Leakage Scenario. In this scenario, the leakage function of cryptographic device is a highly nonlinear function. Without loss of generality, S-box is used in this leakage scenario [14]. Our experimental results show that twelve KS test based distinguishers and MIA all fail to recover the correct key in this scenario.

Note: When SNR goes into positive infinity, the performance of PKS with a fixed parameter may decrease with the increase of the trace number. This indicates that the parameter in PKS is critical to the performance of PKS, as is shown in [13].

4.2 Practical Experiments

In order to show how these twelve KS test based distinguishers behave in practical scenarios, we perform attacks against unprotected software AES implementation on 8-bit microcontroller (Case 1) and unprotected hardware AES implementation on Xilinx Vertex-5 FPGA (Case 2), respectively. These power traces are from OpenSCA and from DPA Contest v2, respectively.

In the view of an adversary, we will choose the power model according to our priori knowledge. Specifically, we will use HW model in Case 1, and Hamming distance (HD) model in Case 2. We will choose SR to evaluate the efficiency, by amounting key recovery attacks 300 times. In this part, the experiments are also organized exactly in the same way as that in our simulated experiments, except that we also perform CPA attacks. This means that we place CPA distinguisher in Group D. That is to say, in practical experiments, we will show the performance of traditional CPA distinguisher, which is widely believed to be well capable of characterizing linear leakages.

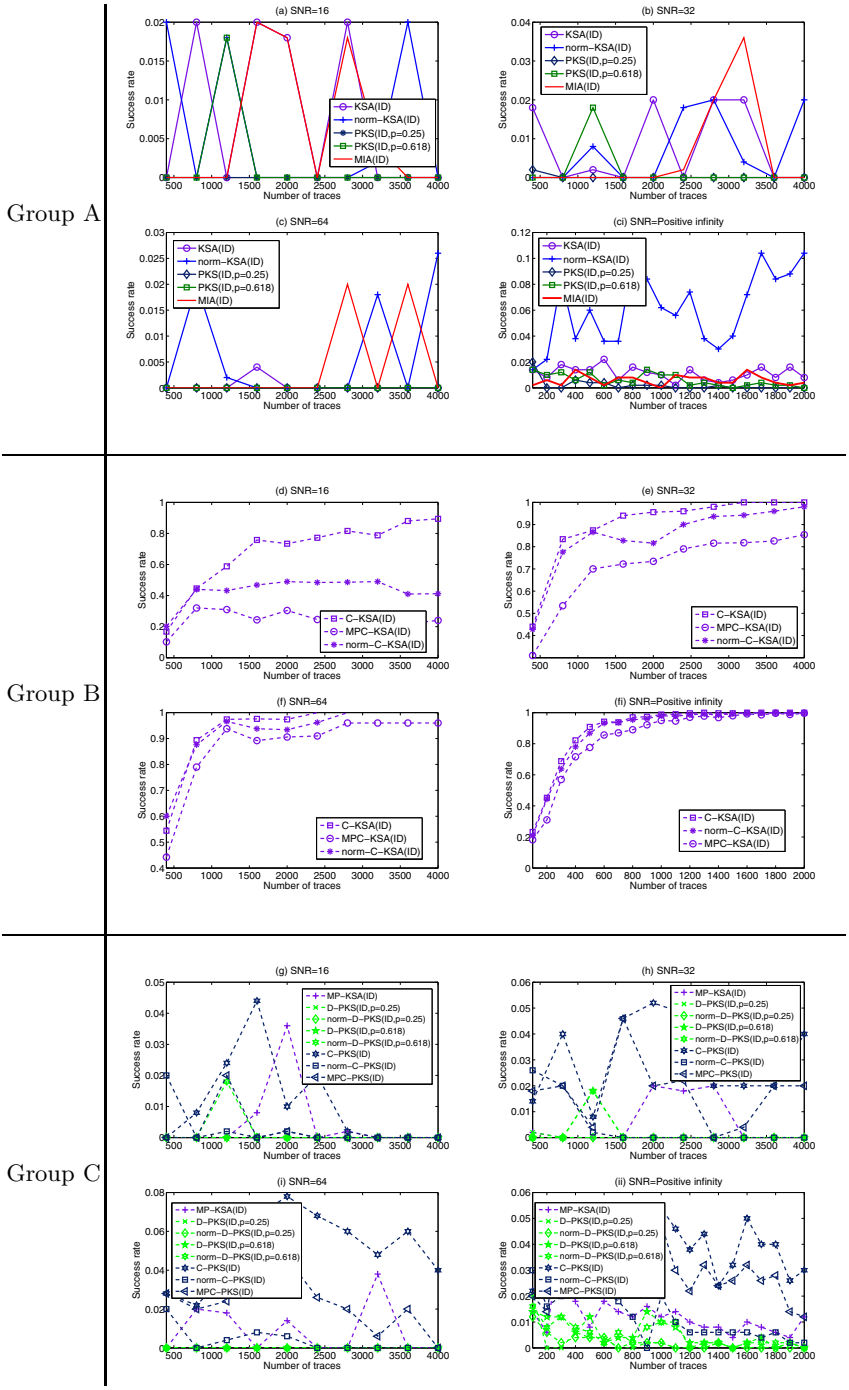


Fig. 4. SRs of different distinguishers against the first AES S-box in USWB leakage

Case 1: Unprotected Software AES Implementation Provided by OpenSCA

In this scenario, the output of the first S-box of the first round of AES operation is chosen as the target. In Group A, Figure 5 (a) shows that, KSA(HW) exhibits the best performance among three existing KS test based distinguishers, so KSA(HW) is used as the benchmark for Group B (see Figure 5(b)) and Group C (see Figure 5(c)). In Group B, Figure 5(b) shows that, MP-KSA(HW) is more efficient than the benchmark. In Group C, Figure 5(c) shows that, other new variants of KS test based distinguishers are less efficient than the benchmark. In Group D, Figure 5(d) shows that, MP-KSA(HW) is less efficient than CPA(HW). In summary, MP-KSA(HW) is the best choice in all these KS test

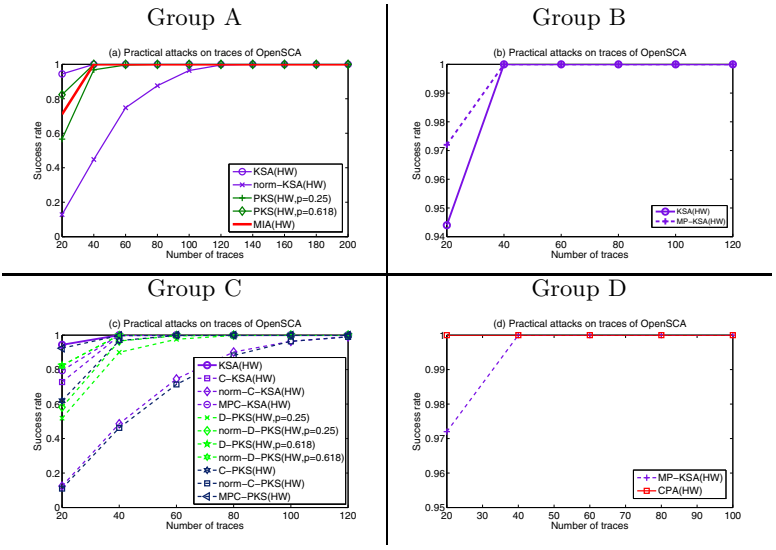


Fig. 5. SRs for twelve KS test based distinguishers, MIA and CPA with HW model in attacks against the first AES S-box

based distinguishers in this case. In the view of an adversary, CPA is an ideal distinguisher. This indicates that, when the leakage of a cryptographic device could be accurately characterized, CPA is the best choice compared with all KS test based distinguishers.

Case 2: Unprotected Hardware AES Implementation Provided by DPA Contest v2

In this scenario, the input of the first S-box of the last round of AES operation is chosen as the target. In Group A, Figure 6(a) shows that, both PKS(HD) and MIA(HD) (MIA(HD,bins=9)) can reveal the correct key, while KSA(HD) and norm-KSA(HD) fail to do that. The empirical parameter in PKS(HD) can largely improve the performance of PKS(HD). Therefore, PKS(HD,p=0.01) is

selected as the benchmark for finding the most promising variants in this case. In Group B, Figure 6(b) shows that, MPC-KSA(HD) and MPC-PKS(HD) outperform PKS(HD,p=0.01) in terms of achieving a partial success rate of 80%. In Group C, other KS test based distinguishers are less efficient than the benchmark, so we do not discuss them in more details. In Group D, Figure 6(d) shows that, MPC-KSA(HD) is even better than CPA(HD).

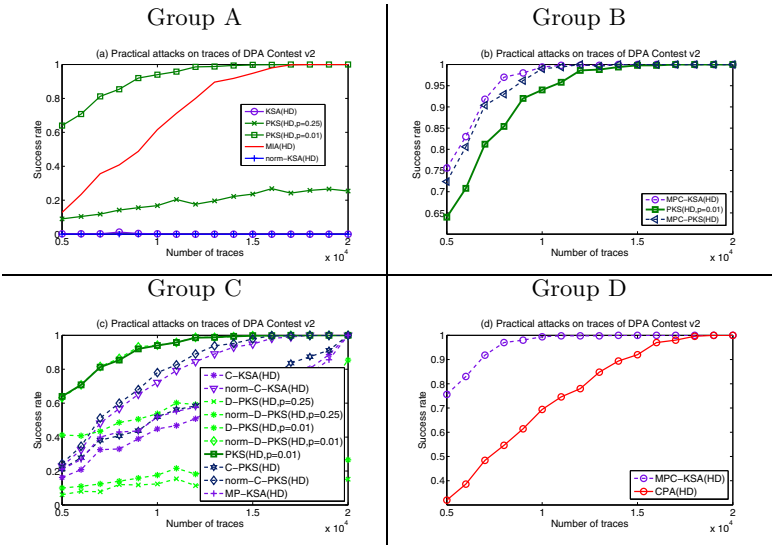


Fig. 6. SRs for twelve KS test based distinguishers, MIA and CPA with HD model in attacks against the first AES S-box of the last round

Table 3. Number of traces required to achieve partial SR of 80% on individual byte

	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	byte 8
MPC-KSA	5,300	6,100	5,700	9,800	9,600	5,500	4,800	6,800
CPA	12,500	10,000	6,900	7,000	12,700	6,000	5,900	7,400
	byte 9	byte 10	byte 11	byte 12	byte 13	byte 14	byte 15	byte 16
MPC-KSA	4,500	5,200	9,200	3,500	4,100	14,500	6,000	5,500
CPA	6,800	3,600	10,000	3,000	6,600	16,900	15,000	5,100

In order to enhance the understanding of whether or not MPC-KSA is a reasonable alternative for CPA, we perform attacks on all sixteen bytes of AES encryption. Table 3 shows the number of traces required to achieve partial SR of 80% of attacks on individual bytes. Although CPA is more efficient than MPC-KSA on four bytes (byte 4, byte 10, byte 12, byte 16), it is less efficient on other twelve bytes. For example, for byte 15, the number of required traces for MPC-KSA to achieve partial SR of 80% is 6,000, while that of CPA is 15,000. However, for byte 4, the number of required traces for MPC-KSA to achieve

partial SR of 80% is 9,800, while that of CPA is 7,000. Although MPC-KSA does not perform consistently better than CPA, it performs better than CPA on 75% of sixteen bytes. As a whole, MPC-KSA is more efficient than CPA in terms of the required number of traces to achieve the global SR of 80%. In summary, MPC-KSA is the best choice in this case. This experimental result indicates that, when the leakages of a cryptographic device could not been accurately characterized, MPC-KSA exhibits better performance than CPA in terms of SR, as the former is capable of measuring the total dependency between hypothetical power consumptions and physical leakages.

5 Conclusions

Distinguishers play an vital role in exploiting physical leakages in side-channel attacks. Due to the capability of dealing with both linear and nonlinear dependencies, generic side-channel distinguishers are being increasingly popular. Among those are KS test based distinguishers, such as KSA and PKS. In this paper, we constructed nine new variants of KS test based distinguishers via combining different construction strategies of KSA and PKS, and then explored the effectiveness and efficiency of twelve KS test based distinguishers and MIA in typical simulated scenarios and practical scenarios.

In a whole, we experimentally investigated the performance of KS test based distinguishers, and provided some helpful guides on how to choose a suitable distinguisher. One of the most interesting observations is that MPC-KSA is more efficient than CPA against unprotected hardware AES implementation on Xilinx Vertex-5 FPGA in DPA Contest v2. However, we did not provide any theoretical analysis yet about why this happens, which could be part of our future work.

Acknowledgments. This work was supported in part by National Natural Science Foundation of China (No. 61272478, 61073178, 60970135 and 61170282), Beijing Natural Science Foundation (No. 4112064), Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA06010701), and IIE Cryptography Research Project (No. Y2Z0011102).

References

1. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
2. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Examining Smart-Card Security under the Threat of Power Analysis Attacks. IEEE Trans. Comput. 51(5), 541–552 (2002)
3. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)

4. Le, T.-H., Clédière, J., Canovas, C., Robisson, B., Servière, C., Lacoume, J.-L.: A Proposition for Correlation Power Analysis Enhancement. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 174–186. Springer, Heidelberg (2006)
5. Mangard, S., Oswald, E., Standaert, F.-X.: All for one-one for all: Unifying univariate DPA attacks. IET Information Security 5(2), 100–110 (2011)
6. Doget, J., Prouff, E., Rivain, M., Standaert, F.-X.: Univariate side channel attacks and leakage modeling. Journal of Cryptographic Engineering 1(2), 123–144 (2011)
7. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis: A Generic Side-Channel Distinguisher. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
8. Moon, Y.-I., Rajagopalan, B., Lall, U.: Estimation of Mutual Information using Kernal Density Estimators. Physical Review E 52, 2318–2321 (1995)
9. Walters-Williams, J., Li, Y.: Estimation of mutual information: A survey. In: Wen, P., Li, Y., Polkowski, L., Yao, Y., Tsumoto, S., Wang, G. (eds.) RSKT 2009. LNCS, vol. 5589, pp. 389–396. Springer, Heidelberg (2009)
10. Veyrat-Charvillon, N., Standaert, F.-X.: Mutual Information Analysis: How, When and Why? In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 429–443. Springer, Heidelberg (2009)
11. Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F.-X., Veyrat-Charvillon, N.: Mutual Information Analysis: A Comprehensive Study. Journal of Cryptology 24(2), 269–291 (2011)
12. Whitnall, C., Oswald, E., Mather, L.: An Exploration of the Kolmogorov-Smirnov Test as Competitor to Mutual Information Analysis. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 234–251. Springer, Heidelberg (2011)
13. Liu, J.-Y., Zhou, Y.-B., Yang, S.-G., Feng, D.-G.: Generic Side-Channel Distinguisher Based on Kolmogorov-Smirnov Test: Explicit Contruction and Practical Evaluation. Chinese Journal of Electronics 21(3), 547–553 (2012)
14. Whitnall, C., Oswald, E.: A Fair Evaluation Framework for Comparing Side-Channel Distinguishers. Journal of Cryptographic Engineering 1(2), 145–160 (2011)
15. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
16. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition *vs.* Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)

Cryptanalysis of the OKH Authenticated Encryption Scheme

Peng Wang[1], Wenling Wu[2], and Liting Zhang[2]

[1] State Key Laboratory of Information Security
Institute of Information Engineering, Chinese Academy of Sciences
[2] Institution of Software, Chinese Academy of Sciences
wp@is.ac.cn,{wwl,zhangliting}@is.iscas.ac.cn

Abstract. Alomair proposed a new authenticated encryption scheme OKH at ACNS 2012, and proved its security, i.e. authenticity and privacy. Our research shows that it is not the case. We only need one query to break the authenticity of OKH with success probability of 1, and two queries to break the privacy of OKH with success probability of $1-1/2^n$, where n is the block-length of underlying blockcipher.

Keywords: authenticated encryption, universal hash function family, cryptanalysis.

1 Introduction

The basic requirement of *authenticated encryption* (AE) scheme is to achieve the security function of message authentication code (MAC) and that of encryption scheme at the same time, i.e. authenticity and privacy. Simply speaking, authenticity guarantees that the ciphertext is really delivered from the sender and not modified by the adversary during the transmission. Privacy guarantees that the adversary can not gain any information (except the length) about plaintext from the view of ciphertext. Due to its wide applications, during the past few years, considerable effort has been made to construct AE schemes, e.g. IAPM [7], OCB [11], CCM [12], EAX [2], CWC [8], GCM [9].

A straightforward method to construct AE schemes is by composing of an encryption scheme and a message authentication code (MAC). Three generic compositions are involved: Encrypt-and-MAC (E&M), MAC-then-Encrypt (MtE), and Encrypt-then-MAC (EtM). CCM [12], EAX [2], CWC [8] and GCM [9] can be viewed as composed (two-pass) schemes with refinement of using only one key. The other method is constructing integrated (one-pass) schemes, such as IAPM [7] and OCB [11].

Recently, Alomair proposed a new composed AE scheme OKH [1]. In this design, the author takes the E&M style because he observes that in both E&M and EtM schemes, the security requirements of authenticity can be relaxed, which can improve the efficiency of the overall construction.

Typical MACs are based on blockciphers, such as CBC-MAC [5], CMAC [10] and PMAC [4], but more efficient MACs are based on universal hash function

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 353–360, 2013.

families, in which the message is first compressed into a fixed-length string by a universal hash function and then encrypted to be the tag, e.g. UMAC [3], and MACs in CWC [8] and GCM [9]. The universal hash function family is a group of hash functions without any cryptographic requirement, but satisfying some combinatorial properties. The MAC in OKH is also based on a hash function family called the Odd Key Hash Family, but as mentioned by the author [1] it does not even satisfy the basic property of universal hash family, i.e. property of being universal[1].

Alomair proved the security of OKH by the reduction method with the assumption that the underlying blockcipher is a pseudorandom permutation (PRP). Unfortunately it is not true.

Our Contributions. In this paper we show that both authenticity and privacy of OKH do not hold in the usual security models. As to authenticity, we only need to query a special message to the encryption algorithm of OKH, and then forgery a new ciphertext and its tag that can pass the decryption algorithm of OKH successfully with probability of 1. As to authenticity, we only need to query two special messages to distinguish the ciphertexts from the random strings with probability of $1 - 1/2^n$, where n is the block-length of underlying blockcipher.

2 Description of OKH

2.1 Notations

- For a binary string M, $|M|$ denotes the length of M in bits.
- For a non-empty set $\mathcal{S}$, we denote by $s \xleftarrow{\$} \mathcal{S}$ the selection of a member of $\mathcal{S}$ uniformly at random and assigning it to s.
- A *blockcipher* is a function $E : \{0,1\}^{kl} \times \{0,1\}^{bl} \to \{0,1\}^{bl}$, where bl and kl are the block-length and key-length respectively, and $E_K(\cdot) = E(K, \cdot)$ is a permutation for all $K \in \{0,1\}^{kl}$.
- $X \oplus Y$ denotes the *exclusive or* (XOR) of two string X and Y. When the lengths of X and Y are not equal, we pad some 0s after the short one to make them equal and then do the usual XOR operation. E.g. $11 \oplus 1001 = 1100 \oplus 1001 = 0101$.
- We denote by $\cdot$ the *multiplication*, 0^n the n-bit string of all 0s, $\mathbb{Z}_{2^n} = \{0, 1, 2, \cdots, 2^{n-1}\}$ the set of all none-negative integers less than 2^n, and $\mathbb{Z}^*_{2^n}$ the set of all odd integers in $\mathbb{Z}_{2^n}$. Without confusion, we often use a string or a integer number interchangeably.

2.2 The Odd Key Hash Function Family

The Odd Key hash function family is a crucial component of OKH, which makes use of basic modular arithmetic operations within $\mathbb{Z}_{2^n}$. For an input message

[1] A hash function family $F = \{f_K | K \in \mathcal{K}\}$ is ε-almost-universal if $\#\{K | f_K(X) = f_K(Y)\}/\#\mathcal{K} \leq \varepsilon$ for any $X \neq Y$.

M with bit-length of multiples of n, partition it into a sequence of n-bit blocks, $M = M_1 M_2 \cdots M_l$, then the compressed image of M is given by

$$\text{OK-HASH}_{K_h}(M) = \sum_{i=1}^{l} K_i \cdot M_i \mod 2^n,$$

where the key $K_h = K_1 \cdots K_l$, $K_i \in \mathbb{Z}_{2^n}^*$, $i = 1, \cdots, l$.

Remark 1. As mentioned in [1], OK-HASH is not an almost universal hash function family, because $\text{OK-HASH}_{K_h}(0^{2n}) = \text{OK-HASH}_{K_h}(10^{n-1}10^{n-1})$ for any K_h.

2.3 OKH Authenticated Encryption

OKH is a nonce-based authenticated encryption scheme, combining an encryption scheme $\mathcal{SE} = (\mathcal{E}, \mathcal{D})$ and a special message authentication code OK-MAC based on the function family OK-HASH mentioned above. We can view OKH as an E&M composition AE scheme, and divide it into three components: OKH = (OKH.Key, OKH.Enc, OKH.Dec), where OKH.Key is a key generation algorithm, OKH.Enc is an encryption algorithm and OKH.Dec is a decryption algorithm.

Key Generation Algorithm. OKH has two keys, one for $\mathcal{SE}$, one for OK-MAC, denoted as K_e and K_h respectively, which are generated independently.

Encryption Algorithm. Both $\mathcal{SE}$ and OK-MAC only handle messages of full blocks. In order to treat arbitrary length messages, OKH.Enc pads the bit 1 and minimal bits of 0, making the length of the messages be multiples of block-length. We simply write the result of this procedure as $M10^*$. OKH uses $\mathcal{E}$ to get the ciphertext and OK-MAC to get the authentication tag:

$$\text{OKH.Enc}_{K_e,K_h}(N, M) = (\mathcal{E}_{K_e}(N, M), \text{OK-MAC}_{K_h,K_e}(N, M)),$$

where $\mathcal{E}$ encrypts the message block by block using the underlying blockcipher with different key $K_e \oplus (N||i)$,

$$\mathcal{E}_{K_e}(N, M) = E_{K_e \oplus (N||1)}(M_1)||E_{K_e \oplus (N||2)}(M_2)||\cdots||E_{K_e \oplus (N||l)}(M_l 10^*),$$

and

$$\text{OK-MAC}_{K_h,K_e}(N, M) = E_{K_e}(\text{OK-HASH}_{K_h}(M10^*) \oplus N),$$

when the length of input to the blockcipher E_{K_e} is less than one block, we pad some zeros to fill it.

Decryption Algorithm. OKH.Dec recovers the plaintext, and uses OK-MAC to regenerate the tag to decide whether to return the plaintext or not.

$$\text{OKH.Dec}_{K_e,K_h}(N,C,T) = \begin{cases} \perp & \text{if OK-MAC}_{K_h,K_e}(N,M) \neq T, \\ M & \text{else, where } M = \mathcal{D}_{K_e}(N,C), \end{cases}$$

where $\mathcal{D}$ is the inverse of $\mathcal{E}$,

$$\mathcal{D}_{K_e}(N,C) = D_{K_e \oplus (N||1)}(C_1)||D_{K_e \oplus (N||2)}(C_2)||\cdots||D_{K_e \oplus (N||l)}(C_l).$$

As a summery, we conclude OKH in pseudocodes as in fig. 1, and illustrate it in fig. 2.

Key Generation:

OKH.Key
$K_e \xleftarrow{\$} \{0,1\}^{kl}$
$K_i \xleftarrow{\$} \mathbb{Z}_{2^n}^*,\ i = 1,\cdots,l$

OKH Encryption:

$\text{OKH.Enc}_{K_e,K_h}(N,M)$
$C \leftarrow \mathcal{E}_{K_e}(N,M)$
$T \leftarrow \text{OK-MAC}_{K_h,K_e}(N,M)$
return (C,T)

OKH Decryption:

$\text{OKH.Dec}_{K_e,K_h}(N,C,T)$
$M \leftarrow \mathcal{D}_{K_e}(N,C)$
if $T \neq \text{OK-MAC}_{K_h,K_e}(N,M)$
then return $\perp$
return M

Fig. 1. The pseudocodes of OKH Authenticated Encryption

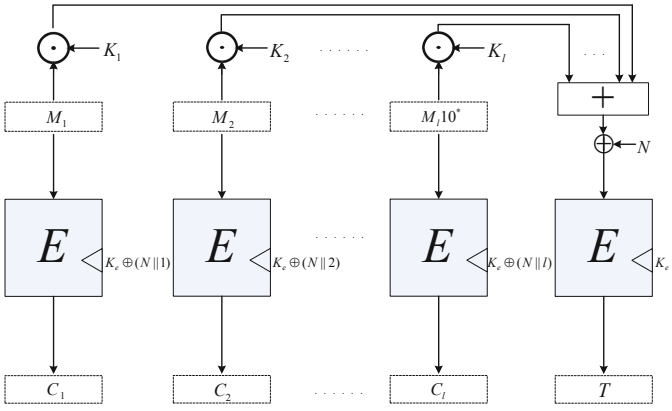


Fig. 2. The OKH Authenticated Encryption Scheme, when the block-length of the underlying blockcipher and that in the OK-HASH are equal

Remark 2. In the original description of OKH [1], the block-length of the underlying blockcipher and that in the OK-HASH may not be equal, the former is no less than the later. In the following discussion, we only consider the situation that the two lengths are equal (i.e. $bl = n$), just as illustrated in fig. 2.

3 Security Models

We adopt the standard security models as those mentioned in [1].

Authenticity Model. The adversary $\mathcal{A}$ is given oracle access to the encryption algorithm OKH.Enc. $\mathcal{A}$ queries OKH.Enc with a pair of nonce and message with restriction that he never repeats the nonce, or in other words he is nonce-respecting, observing the output. After some queries (current query may depend on past queries), he returns a triple of nonce, ciphertext and tag (N, C, T), which does not appear before in the previous answers to the queries. If (N, C, T) is valid, i.e. $\text{OKH.Dec}(N, C, T) \neq \perp$, we say that $\mathcal{A}$ makes a successful forgery. Formally, the advantage of $\mathcal{A}$ is defined by

$$\mathbf{Adv}^{auth}_{\text{OKH}}(\mathcal{A}) = \Pr[\mathcal{A}^{\text{OKH.Enc}(\cdot,\cdot)} \text{ forges}].$$

Privacy Model. The nonce-respecting adversary $\mathcal{B}$ is also given oracle access to the encryption algorithm OKH.Enc. $\mathcal{B}$ queries OKH.Enc with pairs of nonce and message, observing the outputs, trying to distinguish it from random bits. Formally, the advantage of $\mathcal{B}$ is defined by

$$\mathbf{Adv}^{priv}_{\text{OKH}}(\mathcal{B}) = |\Pr[\mathcal{B}^{\text{OKH.Enc}(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 1]|,$$

where $\$(N, M)$ returns a random string with the same length of $\text{OKH.Enc}(N, M)$.

4 Cryptanalysis of OKH

4.1 Some Properties

We first notice some properties of a special binary integer number 10^{n-1} in $\mathbb{Z}_{2^n}$.

Property 1. For any odd integers K_i and K_j,

$$10^{n-1} \cdot K_i \equiv 10^{n-1} \mod 2^n, \tag{1}$$

$$10^{n-1} \cdot K_i + 10^{n-1} \cdot K_j \equiv 0^n \mod 2^n. \tag{2}$$

Using these properties, we construct two pairs of messages which have the same authentication tag under OK-MAC with the same or different nonces.

Property 2. For arbitrary blocks $M_i \in \{0,1\}^n$ $(i = 1, \cdots, l)$, we have

$$\text{OK-MAC}(N, M_1 \cdots M_l 10^{n-1} 10^{n-1}) = \text{OK-MAC}(N, M_1 \cdots M_l), \tag{3}$$

$$\text{OK-MAC}(N, M_1 \cdots M_l 10^{n-1}) = \text{OK-MAC}(N', M_1 \cdots M_l), \tag{4}$$

where $N \oplus N' = 10^{nl-1}$, nl is the length of the nonce.

Proof. It is easy to verify the following two equations about OK-HASH,

$$\text{OK-HASH}(M_1 \cdots M_l 10^{n-1} 10^{n-1} 10^{n-1}) = \text{OK-HASH}(M_1 \cdots M_l 10^{n-1}),$$

$$\text{OK-HASH}(M_1 \cdots M_l 10^{n-1} 10^{n-1}) = \text{OK-HASH}(M_1 \cdots M_l 10^{n-1}) \oplus 10^{n-1}.$$

By the definition of OK-MAC, the equations of (3) and (4) follow. □

So if we look at the authentication code in OKH solely, OK-MAC is not a secure MAC, due to the fact that OK-HASH is not almost universal. We can query the MAC using one message, and get the tag, then the other message and the tag constitute a successful forgery immediately. But breaking authenticity of AE scheme is slightly different, what the adversary tries to find is a valid triple of nonce, ciphertext and tag which does not appear before. But we notice that in equation (3) $M_1 \cdots M_l$ is the prefix of $M_1 \cdots M_l 10^{n-1} 10^{n-1}$, which will help us to break the authenticity of OKH.

4.2 Breaking Authenticity of OKH

We give the following authenticity attacking algorithm. This attack only makes one special query to the encryption oracle $\text{OKH.Enc}_{K_e,K_h}(\cdot,\cdot)$, then returns a valid triple of nonce, ciphertext and tag which does not appear before.

Authenticity Attacking Algorithm $\mathcal{A}$:

1) Query $(N, M_1 \cdots M_l 10^{n-1} 10^{n-1})$ to the encryption oracle, where M_i $(i = 1, \cdots, l)$ are arbitrary blocks, and get $(C_1 C_2 \cdots C_{l+3}, T)$, where C_i $(i = 1, \cdots, l+3)$ are ciphertext blocks, T is the tag.
2) Return $(N, C_1 C_2 \cdots C_{l+1}, T)$.

Analysis of Algorithm $\mathcal{A}$. The ciphertext blocks to the query are $C_i = E_{K_e \oplus (N||i)}(M_i)$, $i = 1, \cdots, l$, $C_{l+1} = E_{K_e \oplus (N||(l+1))}(10^{n-1})$, $j = 1, 2, 3$. So the corresponding plaintext blocks of C_i $(i = 1, \cdots, l)$ and C_{l+1} under same nonce N are M_i $(i = 1, \cdots, l)$ and 10^{n-1}. 10^{n-1} is interpreted as the padding, therefore the final plaintext is $M_1 \cdots M_l$. Equation (3) shows that the tags of $(N, M_1 \cdots M_l 10^{n-1} 10^{n-1})$ and $(N, M_1 \cdots M_l)$ are the same. So $(N, C_1 C_2 \cdots C_{l+1}, T)$ is valid, which does not appear before. Therefore $\mathbf{Adv}^{auth}_{\text{OKH}}(\mathcal{A}) = 1$.

Remark 3. In the proof of authenticity in [1], the author did not consider the situation that one plaintext may be the prefix of the other. The security proof lies on the fact that the corresponding plaintexts of two different ciphertext differ in single block or several blocks. This is obvious not true.

4.3 Breaking Privacy of OKH

In equation (4), two messages have the same authentication tag under different nonces. Therefore we can make two nonce-respecting queries, resulting in two equal tags, which can be used to distinguish ciphertexts from random strings.

Privacy attacking algorithm $\mathcal{B}$:
1) Query $(N, M_1 \cdots M_l 10^{n-1})$, and get $(C_1 C_2 \cdots C_{l+2}, T)$.
2) Query $(N', M_1 \cdots M_l)$ where $N \oplus N' = 10^{nl-1}$, and get $(C'_1 C'_2 \cdots C'_{l+1}, T')$.
3) If $T = T'$, then return 1, else return 0.

Analysis of algorithm $\mathcal{B}$. If the oracle is OKH.Enc$(\cdot, \cdot)$. By equation (4), we know that $T = T'$ always holds. If the oracle is $\$(\cdot, \cdot)$, T and T' are two random strings. The probability of $T = T'$ is $1/2^n$, therefore $\mathbf{Adv}^{priv}_{\mathrm{OKH}}(\mathcal{B}) = 1 - 1/2^n$.

Remark 4. In the current real-or-random privacy model, OKH is totally insecure. We note that even in a more general left-or-right privacy model, OKH is not secure. In this model, the adversary can query $(N, M), (N, M')$ with restriction that $|M| = |M'|$, the oracle only returns left or right ciphertext, and after several queries the adversary must guess this one-bit information about left-or-right. In this model, the adversary can attack as following: 1) Query $(N, M_1 \cdots M_l 10^{n-1}), (N, M_1 \cdots M_l 0^n)$; 2) Query $(N', M_1 \cdots M_l), (N', M_1 \cdots M_l)$ where $N \oplus N' = 10^{nl-1}$; 3) If the two returned tags are equal, the adversary guesses it is left, else guesses right. It is easy to verify that the success probability is 1.

Remark 5. The proof of privacy in [1], the security lies only on the pseudorandomness of the underlying blockcipher, which assumes that once the key of the blockcipher is randomly selected, the blockcipher is indistinguishable from a uniformly random permutation, i.e. the blockcipher is a pseudorandom permutation (PRP). The encryption component of OKH is similar to the ECB mode, with exception that the keys to the underlying blockcipher are $K_e \oplus (N||i)$ $(i = 1, \cdots, l)$, which are all related by the key K_e. The only assumption of PRP can not guarantee the independence among the blockcipher invocations. With the assumption of PRP, it is easy to construct a new block cipher like [6], which is also a PRP, but the same under two different related keys such as $K_e \oplus (N||1)$ and $K_e \oplus (N||2)$. Then first two block encryptions of OKH are the same, which also can be used to break the privacy of OKH. Algorithm $\mathcal{B}$ only makes use of the weakness of OKH-MAC.

5 Conclusion

Although the security proofs were given in [1], the OKH authenticated encryption scheme is not secure at all. Both authenticity and privacy of OKH do not hold in the common security models. We only need one or two queries to break the security of OKH with success probability of 1 or almost 1.

Acknowledgement. This research is supported by the National Natural Science Foundation Of China (No. 60903219, 61272477, 61272476, 61202422), the National Grand Fundamental Research 973 Program of China (No. 2013CB338002), the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702 and the IIE's Research Project on Cryptography.

References

1. Alomair, B.: Authenticated encryption: How reordering can impact performance. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 84–99. Springer, Heidelberg (2012)
2. Bellare, M., Rogaway, P., Wagner, D.: The EAX mode of operation. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 389–407. Springer, Heidelberg (2004)
3. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC: Fast and secure message authentication. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 216–233. Springer, Heidelberg (1999)
4. Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 384–397. Springer, Heidelberg (2002)
5. FIPS-133: Federal information processing standards publication (FIPS 133). computer data authentication (1985)
6. Iwata, T., Kurosawa, K.: On the correctness of security proofs for the 3GPP confidentiality and integrity algorithms. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 306–318. Springer, Heidelberg (2003)
7. Jutla, C.S.: Encryption modes with almost free message integrity. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 529. Springer, Heidelberg (2001)
8. Kohno, T., Viega, J., Whiting, D.: CWC: A high-performance conventional authenticated encryption mode. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 408–426. Springer, Heidelberg (2004)
9. McGrew, D.A., Viega, J.: The galois/counter mode of operation (GCM) (2004), `http://csrc.nist.gov/groups/ST/toolkit/BCM/`
10. NIST: Recommendation for block cipher modes of operation: The CMAC mode for authentication. NIST Special Publication 800-38B (2005), `http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf`
11. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) ACM Conference on Computer and Communications Security, pp. 196–205. ACM (2001)
12. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM) (2002), `http://csrc.nist.gov/groups/ST/toolkit/BCM/`

Security Evaluation of Rakaposhi Stream Cipher

Mohammad Ali Orumiehchiha[1], Josef Pieprzyk[1], Elham Shakour[2], and Ron Steinfeld[3]

[1] Center for Advanced Computing, Algorithms and Cryptography, Department of Computing, Faculty of Science, Macquarie University, Sydney, NSW 2109, Australia
{mohammad.orumiehchiha,josef.pieprzyk}@mq.edu.au
[2] Faculty of Mathematics, Amirkabir University,Tehran, Iran
elham.shakoor@gmail.com
[3] Clayton School of Information Technology Monash University, Clayton VIC 3800, Australia
ron.steinfeld@monash.edu

Abstract. Rakaposhi is a synchronous stream cipher, which uses three main components: a non-linear feedback shift register (NLFSR), a dynamic linear feedback shift register (DLFSR) and a non-linear filtering function (NLF). NLFSR consists of 128 bits and is initialised by the secret key K. DLFSR holds 192 bits and is initialised by an initial vector (IV). NLF takes 8-bit inputs and returns a single output bit. The work identifies weaknesses and properties of the cipher. The main observation is that the initialisation procedure has the so-called sliding property. The property can be used to launch distinguishing and key recovery attacks. The distinguisher needs four observations of the related (K, IV) pairs. The key recovery algorithm allows to discover the secret key K after observing 2^9 pairs of (K, IV). Based on the proposed related-key attack, the number of related (K, IV) pairs is $2^{(128+192)/4}$ pairs.

Further the cipher is studied when the registers enter short cycles. When NLFSR is set to all ones, then the cipher degenerates to a linear feedback shift register with a non-linear filter. Consequently, the initial state (and Secret Key and IV) can be recovered with complexity $2^{63.87}$.

If DLFSR is set to all zeros, then NLF reduces to a low non-linearity filter function. As the result, the cipher is insecure allowing the adversary to distinguish it from a random cipher after 2^{17} observations of keystream bits. There is also the key recovery algorithm that allows to find the secret key with complexity 2^{54}.

Keywords: Rakaposhi Stream Cipher, Related Key Attack, Weak State, Cryptanalysis, Distinguishing Attack, Key Recovery Attack.

1 Introduction

Stream ciphers are symmetric cipher systems, which provide confidentiality in many applications ranging from mobile phone communication to private virtual networks. They may be implemented efficiently in software and hardware and

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 361–371, 2013.

are a preferred choice when dealing with an environment that has restricted computing resources (such as smart cards, RF tags). The inner work of stream ciphers is controlled normally by two parameters: an initialisation vector (IV) and a key (K). The initialisation vector is public and the key is secret.

There are many tools for analysis of stream ciphers. The most two prominent are the linear and differential attacks. There are also many more "exotic" tools for analysis. One such tool is the related key attack. This attack is especially dangerous when an adversary has access to many pairs of (IV, K). This may happen if the adversary is able to experiment with a stream cipher device that has been left unattended (so-called midnight or lunchtime attacks). The adversary may modify the unknown secret key by forcing changes on few key bits, may try different IVs or may modify the clock procedure.

The Rakaposhi stream cipher was designed by Cid, Kiyomoto, and Kurihara in 2009 (see [1]). The cipher is based on a non-linear feedback shift register (NLFSR) and a dynamic linear shift register (DLFSR). The design was crafted to be suitable for lightweight implementations, where computing, power and time resources are in short supply. The cipher claims 128-bit security and has been designed to complement the eStream portfolio for hardware-oriented stream ciphers. The designers of the cipher claim that Rakaposhi is an efficient synchronous stream cipher that resists all known attacks and they conjecture that it is also secure against other yet unknown attacks.

This work analyses the Rakaposhi cipher and shows its weaknesses. In particular, we

- examine the resistance of the cipher against the related key attack, where the adversary can access related pairs (IV, K),
- study the security implications when NLFSR enters a short cycle,
- investigate the security level when DLFSR enters a short cycle.

1.1 Related Works

The related key attack is studied in the context the Rakaposhi cipher and its initialization procedure. Similar analysis can be found in [3,4,8] but in a different context. The related key attack can be seen as a member of the differential cryptanalysis toolbox. We use the slide attack published by Canniere et. al. in [2] to launch the related key attack. The second part of our work is influenced by the paper of Zhang and Wang [9], in which the authors study the security of the Grain stream cipher [5,6]. While working on the paper, we have become aware of a paper [7] that shows a similar analysis of the initialization procedure of Rakaposhi but in our paper, we have improved dramatically the efficiency of key recovery attack and we have also identified two classes of weak states.

The rest of the paper is structured as follows. Section 2 describes briefly the Rakaposhi stream cipher. Section 3 presents the weaknesses of the cipher and investigates the security of the initialisation procedure under the related key attack. Section 4 discusses the security implications when one of the registers (either NLFSR or DLFSR) enters a short cycle. Section 5 concludes the work.

2 Description of Rakaposhi Stream Cipher

The Rakaposhi stream cipher consists of the following three building blocks (see Figure 1) :

- a 128-bit NLFSR also called the register A,
- a 192-bit DLFSR also called the register B,
- a non-linear function NLF

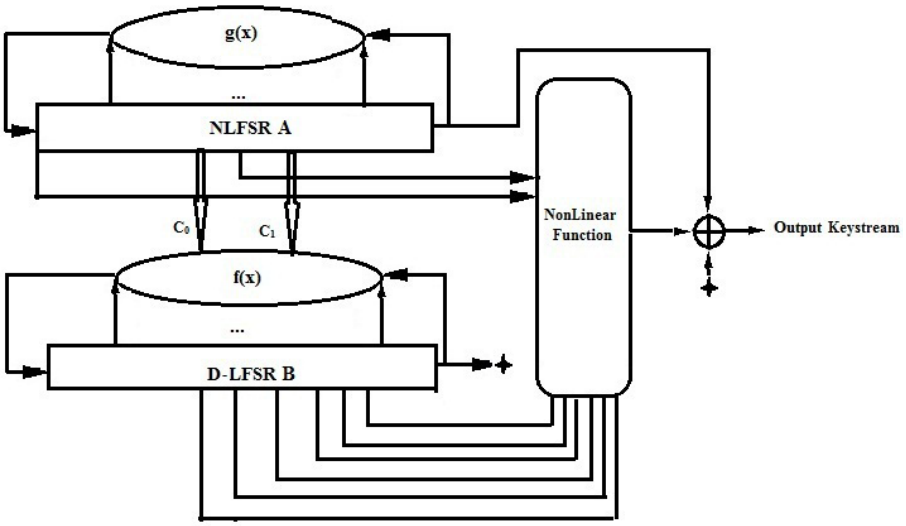


Fig. 1. Rakaposhi Stream Cipher

The NLSFR register A is defined by its feedback function:

$$\begin{aligned} g(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = {} & x_1x_3x_9 \oplus x_1x_7x_9 \oplus x_5x_8 \oplus x_2x_5 \oplus \\ & x_3x_8 \oplus x_2x_7 \oplus x_9 \oplus x_8 \oplus x_7 \oplus x_6 \oplus \\ & x_5 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0 \oplus 1, \end{aligned}$$

where $a_{t+128} = g(a_t, a_{t+6}, a_{t+7}, a_{t+11}, a_{t+16}, a_{t+28}, a_{t+36}, a_{t+45}, a_{t+55}, a_{t+62})$ and a_{t+i} is i^{th} bit of the register A at clock t.

The DLFSR register B is controlled by two bits (c_0, c_1) taken from the state of NLFSR. The bits choose one of four possible characteristic polynomials of DLFSR. The form of the polynomials is as follows:

$$\begin{aligned} f(x) = {} & x_{192} \oplus x_{176} \oplus c_0x_{158} \oplus (1 \oplus c_0)x_{155} \oplus c_0c_1x_{136} \oplus \\ & c_0(1 \oplus c_1)x_{134} \oplus c_1(1 \oplus c_0)x_{120} \oplus (1 \oplus c_0)(1 \oplus c_1)x_{107} \oplus \\ & x_{93} \oplus x_{51} \oplus x_{49} \oplus x_{41} \oplus x_{37} \oplus x_{14} \oplus 1, \end{aligned} \tag{1}$$

where the bits (c_0, c_1) are the 42^{th} and 90^{th} bits of the register A at clock t, respectively. The recursive relation for DLFSR is as follows:

$$\begin{aligned} b_{t+192} = b_t \oplus b_{t+14} \oplus b_{t+37} \oplus b_{t+41} \oplus b_{t+49} \oplus b_{t+51} \oplus b_{t+93} \oplus \\ \overline{c_0} \cdot \overline{c_1} \cdot b_{t+107} \oplus \overline{c_0} \cdot c_1 \cdot b_{t+120} \oplus c_0 \cdot \overline{c_1} \cdot b_{t+134} \oplus \\ c_0 \cdot c_1 \cdot b_{t+136} \oplus \overline{c_0} \cdot b_{t+155} \oplus c_0 \cdot b_{t+158} \oplus b_{t+176} \end{aligned} \tag{2}$$

where $\overline{c_i} = 1 \oplus c_i$ denotes the negation of c_i and b_{t+i} is the i^{th} bit of B at clock t.

Rakaposhi uses a non-linear filtering function $NLF : GF(2^8) \rightarrow GF(2)$, which is based on the AES S-Box. The NLF function is a balanced Boolean function and its algebraic degree equals to 7. NLF takes 8-bit inputs (2 bits from A and 6 bits from B) and outputs

$$s_t = NLF(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128}),$$

where the two bits a_{t+67}, a_{t+127} are taken from A and the other bits from B. Finally, the keystream output is generated by linear combination of the outputs of both registers A and B with the output of the NLF function. The reader interested in more detail is referred to the original paper [1].

2.1 Initialisation Procedure

The goal of the initialisation procedure is to mix IV and the secret key K. Assume that $IV = [iv_0, \cdots, iv_{191}]$ and $K = [k_0, \cdots, k_{127}]$. K and IV are loaded to NLFSR and DLFSR, respectively so

$$\begin{aligned} a_i &= k_i \text{ for } \ 0 \leq i \leq 127 \\ b_j &= iv_j \text{ for } \ 0 \leq j \leq 191, \end{aligned}$$

where the bits of registers A and B are a_i and b_j, respectively. The registers A and B are then clocked 448 times without producing any output keystream bits. This stage is divided into two phases:

Phase 1: the output of NLF is linearly combined with the feedback of the register B for the first 320 clocks.

Phase 2: the output of NLF is linearly combined with the feedback of the register A for the next 128 clocks.

After finishing Phase 2, the cipher starts producing keystream outputs.

3 Cryptanalysis of Rakaposhi Stream Cipher

Now, we show how we can launch the distinguishing and key recovery attacks on the Rakaposhi cipher. The attacks use a sliding property of the cipher. An interesting property of the proposed attacks is that their complexities are not affected by the number of clocks, which the cipher performs during the initialisation process. This means that the attacks works even if the number of clocks is increased.

3.1 Properties of Rakaposhi Cipher

We present some cryptographic properties of the Rakaposhi stream cipher that corroborate the proposed attacks.

1. The secret key and IV are loaded in two registers A and B, respectively. Consequently, at clock $t = 0$, A contains K and B contains IV.
2. The initialisation procedure applies the same primitives that are used during the keystream generation stage. This implies that the initialisation for the key and IV is similar to the initialisation for the key and IV when they are shifted by one position. We refer to this characteristics as the sliding property.
3. The register A (NLFSR) has a short cycle of the length 1. When the state of A becomes all ones, then A stays in this state forever.
4. The register B (DLFSR) has a short cycle of the length 1. When the state of B becomes all zeros, then B stays in this state forever.

The first two properties mean that the adversary may find related (K, IV) pairs, which produce keystream outputs that are shifted. The third and fourth properties can be exploited by the adversary so they can distinguish the cipher from a truly random binary source and recover internal state of the cipher and finally corresponding secret key.

3.2 Related Key Attack on Rakaposhi

In our sliding attack we assume that we have two related pairs (K, iv) and $(\widehat{K}, \widehat{iv})$. Consider the initialisation procedure for the two pairs. Let $K = (k_0, \cdots, k_{127})$ and $iv = (iv_0, ..., iv_{191})$ be loaded into the registers A and B, respectively. Denote the states of the registers A and B at the clock t by A^t and B^t, respectively. The evolution of states over time is described below.

$$
\begin{array}{rll}
 & A^0 = [k_0, \cdots, k_{127}] & B^0 = [iv_0, \cdots, iv_{191}] \\
 & A^1 = [k_1, \cdots, k_{127}, a_{128}] & B^1 = [iv_1, \cdots, iv_{191}, b_{192}] \\
 & \vdots & \vdots \\
\xRightarrow{Phase\ 2\ of\ Initialization} & A^{320} = [a_{320}, \cdots, a_{448}] & B^{320} = [b_{320}, \cdots, b_{512}] \\
 & \vdots & \vdots \\
\xRightarrow{Initialization\ finished} & A^{448} = [a_{448}, \cdots, a_{576}] & B^{448} = [b_{448}, \cdots, b_{640}] \\
\xRightarrow{Key\ Generation\ started} & A^{449} = [a_{449}, \cdots, a_{577}] & B^{449} = [b_{449}, \cdots, b_{641}] \\
 & A^{450} = [a_{450}, \cdots, a_{578}] & B^{450} = [b_{450}, \cdots, b_{642}]
\end{array}
$$

The keystream output bits z_i; $i \geq 0$, are computed as follows:

$$\begin{aligned} z_0 &= a_{449} \oplus b_{449} \oplus NLF(a_{448+67}, a_{448+127}, \\ &\quad b_{448+23}, b_{448+53}, b_{448+77}, b_{448+81}, b_{448+103}, b_{448+128}) \\ z_1 &= a_{450} \oplus b_{450} \oplus NLF(a_{449+67}, a_{449+127}, \\ &\quad b_{449+23}, b_{449+53}, b_{449+77}, b_{449+81}, b_{449+103}, b_{449+128}) \\ &\vdots \end{aligned}$$

The relation between keystreams generated by the cipher when initialised by the related pairs is described by the following theorem.

Theorem 1. *Given two pairs* (K, iv) *and* $(\widehat{K}, \widehat{iv})$*, where* $K = (k_0, \cdots, k_{127})$ *and* $iv = (iv_0, ..., iv_{191})$*. If the pair* $(\widehat{K}, \widehat{iv})$ *satisfies the following equations*

$$\begin{cases} \widehat{k_i} & = k_{i+1} \quad 0 \leq i \leq 126, \;\; \hat{k}_{127} = a_{128} \\ \widehat{iv_i} & = iv_{i+1} \quad 0 \leq j \leq 190, \;\; \widehat{iv}_{191} = b_{192} \end{cases} \tag{3}$$

then the keystream output bits $\widehat{z_i} = z_{i+1}$ *for* $i \geq 0$ *with probability* 2^{-2}*.*

Proof. By satisfying Equation (3), the internal states of $[A^{320}, B^{320}]$ are equal to $[\widehat{A}^{319}, \widehat{B}^{319}]$. But at the next clock, the states may not be identical because the state $[\widehat{A}^{320}, \widehat{B}^{320}]$ is still at the first step while $[A^{321}, B^{321}]$ is running at the second step. If $\hat{b}^{512} = b^{511}$, which occurs with probability 1/2, then

$$[\widehat{A}^{319}, \widehat{B}^{319}] = [A^{320}, B^{320}].$$

The same argument is valid for the states $[A^{448}, B^{448}]$ and $[\widehat{A}^{447}, \widehat{B}^{447}]$. The states are identical, when $\hat{a}^{446} = b^{447}$, which also happens with probability 1/2. Consequently, $\widehat{z_i} = z_{i+1}$ for $i \geq 0$ with probability 1/4. □

Table 1 presents some (K, IV) pairs, which produce shifted identical key stream outputs. According to Theorem 1, the adversary can use this weakness to generate the same but $l-$bit shifted keystream outputs by defining related (K, IV) pairs with probability $2^{-2 \cdot l}$. The discovered weakness allows the adversary to distinguish the cipher from a random bit generator. Assume that the adversary can apply related (K, IV) pairs but he does not know the exact values of secret key. Then, after applying m ($m \gg 4$) different (randomly generated) related (K, IV) pairs, on the average $m/4$ of generated key stream outputs have identical sequences with just one bit shift.

3.3 Recovery of Secret Keys

Now, we propose a key recovery attack that exploits the sliding property of pairs (K, IV). We show an algorithm that allows to recover the 128-bit key after about 2^9 initialisation operations with related (K,IV) pairs. The attack can find the secret key with probability close to one.

Assume that both (K, IV) and $(\widehat{K}, \widehat{IV})$ generate almost identical keystream bits, where the second keystream is a shifted by one bit copy of the first keystream. At clock $t = 1$, the first generated bit is b_{192}, which is equal to:

$$\begin{aligned} b_{192} = {} & b_0 \oplus b_{14} \oplus b_{37} \oplus b_{41} \oplus b_{49} \oplus b_{51} \oplus b_{93} \oplus (1 \oplus c_0)(1 \oplus c_1)b_{107} \\ & \oplus (1 \oplus c_0)c_1 b_{120} \oplus c_0(1 \oplus c_1)b_{134} \oplus c_0 c_1 b_{136} \oplus (1 \oplus c_0)b_{155} \oplus c_0 b_{158} \oplus b_{176} \\ & \oplus NLF(a_{67}, a_{127}, b_{23}, b_{53}, b_{77}, b_{81}, b_{103}, b_{128}), \end{aligned}$$

where $c_0 = a_{41}$ and $c_1 = a_{89}$. Since the contents of b_i $(0 \leq i \leq 191)$ are known and can be chosen by the adversary, then Equation (4) is a non-linear relation based on only 4 unknown variables $a_{41}, a_{89}, a_{67}, a_{127}$. We now take a closer look at Equation (4):

$$\begin{aligned} b_{192} = {} & b_0 \oplus b_{14} \oplus b_{37} \oplus b_{41} \oplus b_{49} \oplus b_{51} \oplus b_{93} \oplus (1 \oplus a_{41})(1 \oplus a_{89})b_{107} \\ & \oplus (1 \oplus a_{41})a_{89} b_{120} \oplus a_{41}(1 \oplus a_{89})b_{134} \oplus a_{41} a_{89} b_{136} \oplus (1 \oplus a_{41})b_{155} \\ & \oplus a_{41} b_{158} \oplus b_{176} \oplus a_{67} a_{127} b_{23} b_{53} b_{77} b_{81} b_{103} \oplus a_{67} a_{127} b_{23} b_{53} b_{77} b_{81} \\ & \oplus a_{67} a_{127} b_{23} b_{53} b_{77} b_{103} \oplus a_{67} a_{127} b_{23} b_{53} b_{81} b_{103} b_{128} \\ & \oplus a_{67} a_{127} b_{23} b_{53} b_{81} b_{103} \oplus a_{67} a_{127} b_{23} b_{53} b_{81} b_{128} \\ & \oplus a_{67} a_{127} b_{23} b_{53} b_{81} \oplus a_{67} a_{127} b_{23} b_{53} b_{103} b_{128} \oplus a_{67} a_{127} b_{23} b_{77} b_{81} b_{103} \\ & \oplus a_{67} a_{127} b_{23} b_{77} \oplus a_{67} a_{127} b_{23} b_{81} b_{103} \oplus a_{67} a_{127} b_{23} b_{81} b_{128} \oplus a_{67} a_{127} b_{23} b_{128} \\ & \oplus a_{67} a_{127} b_{23} \oplus a_{67} a_{127} b_{53} b_{77} b_{81} b_{103} b_{128} \oplus a_{67} a_{127} b_{53} b_{77} b_{81} b_{128} \\ & \oplus a_{67} a_{127} b_{53} b_{77} b_{81} \oplus a_{67} a_{127} b_{53} b_{77} b_{128} \oplus a_{67} a_{127} b_{53} b_{77} \oplus a_{67} a_{127} b_{53} b_{103} \\ & \oplus a_{67} a_{127} b_{77} b_{81} b_{103} b_{128} \oplus a_{67} a_{127} b_{77} b_{81} b_{103} \oplus a_{67} a_{127} b_{77} b_{81} b_{128} \\ & \oplus a_{67} a_{127} b_{77} b_{103} b_{128} \oplus a_{67} a_{127} b_{77} b_{128} \oplus a_{67} a_{127} b_{81} b_{103} b_{128} \\ & \oplus a_{67} a_{127} b_{81} b_{103} \oplus a_{67} a_{127} b_{81} \oplus a_{67} a_{127} b_{103} \oplus a_{67} a_{127} \qquad (4) \\ & \oplus a_{67} b_{23} b_{53} b_{77} b_{81} b_{103} \oplus a_{67} b_{23} b_{53} b_{77} b_{81} b_{128} \oplus a_{67} b_{23} b_{53} b_{77} \\ & \oplus a_{67} b_{23} b_{53} b_{81} b_{103} b_{128} \oplus a_{67} b_{23} b_{53} b_{81} b_{103} \oplus a_{67} b_{23} b_{53} b_{81} b_{128} \\ & \oplus a_{67} b_{23} b_{53} b_{103} \oplus a_{67} b_{23} b_{77} b_{81} b_{103} b_{128} \oplus a_{67} b_{23} b_{81} b_{103} \\ & \oplus a_{67} b_{23} b_{81} \oplus a_{67} b_{23} b_{103} b_{128} \oplus a_{67} b_{23} b_{128} \oplus a_{67} b_{53} b_{77} b_{81} b_{103} b_{128} \\ & \oplus a_{67} b_{53} b_{77} b_{81} b_{103} \oplus a_{67} b_{53} b_{77} b_{81} b_{128} \oplus a_{67} b_{53} b_{77} b_{81} \oplus a_{67} b_{53} b_{77} b_{128} \\ & \oplus a_{67} b_{53} b_{81} b_{103} b_{128} \oplus a_{67} b_{53} b_{81} \oplus a_{67} b_{53} b_{103} \oplus a_{67} b_{53} \oplus a_{67} b_{77} b_{81} b_{103} \\ & \oplus a_{67} b_{77} b_{103} b_{128} \oplus a_{67} b_{81} b_{103} \oplus a_{67} b_{103} \oplus a_{67} \oplus a_{127} b_{23} b_{53} b_{77} \\ & \oplus a_{127} b_{23} b_{53} b_{81} b_{103} \oplus a_{127} b_{23} b_{53} b_{81} b_{128} \oplus a_{127} b_{23} b_{53} b_{81} \\ & \oplus a_{127} b_{23} b_{53} \oplus a_{127} b_{23} b_{77} b_{81} b_{103} \oplus a_{127} b_{23} b_{77} b_{103} \oplus a_{127} b_{23} b_{77} \\ & \oplus a_{127} b_{23} b_{81} \oplus a_{127} b_{23} \oplus a_{127} b_{53} b_{77} b_{81} b_{103} b_{128} \oplus a_{127} b_{53} b_{77} b_{81} b_{128} \\ & \oplus a_{127} b_{53} b_{77} b_{103} b_{128} \oplus a_{127} b_{53} b_{77} b_{103} \oplus a_{127} b_{53} b_{77} \oplus a_{127} b_{53} b_{81} b_{103} \\ & \oplus a_{127} b_{53} b_{81} \oplus a_{127} b_{53} b_{103} \oplus a_{127} b_{53} b_{128} \oplus a_{127} b_{77} b_{81} b_{103} b_{128} \\ & \oplus a_{127} b_{77} b_{81} b_{128} \oplus a_{127} b_{81} b_{103} \oplus a_{127} b_{81} b_{128} \oplus a_{127} b_{81} \\ & \oplus a_{127} b_{103} b_{128} \oplus a_{127} b_{103} \oplus a_{127} \oplus NLF'(b_{23}, b_{53}, b_{77}, b_{81}, b_{103}, b_{128}) \end{aligned}$$

Table 1. Shifted identical key stream outputs corresponding two related (K, IV) pairs

	Key	IV	Output bits
1	10011011110011101010000	01000111100000010100010011010110	00000110000100011100111
	10000000111010011000000	00000000001000000110111000011111 0	11000001010001010100102
	00011000101100010011111	01110101011000011111000011001100 1	10010100001101111001106
	01110111010000010010010	11000101110110110000010100110110 0	10101010110100000011012
	00000001001100110010010	01000111000000010010010010111100 1	11001110010001000111107
	0100110101111001111	10101101010110001011001010 1	011110000011110101
	00110111100111010100011	10001111000000101000100011010110 0	00001100001000111001117
	00000001110100110000000	00000000010000001101110000111110 0	10000010100010101001015
	00110001011000100111102	11101010110000111110000110011001 1	00101000011011110011016
	11101110100000100100102	10001011101101100000101001101100 0	01010101101000000110113
	00000010011001100100102	10001110000000100100100101111001 1	10011100100010001111004
	100110101111001111	01011010101100010110010101 1	111100000111101010
2	00000000010111111011000	00111000101110001110111000101100 0	01110100100110000001117
	11101100111001000101117	01000100011110000110010101001011 7	00110010110000101110102
	01110110110011110010102	01011001000101000000110100110001 1	11111001100001111011107
	11011101100001110010002	01001001101101110001110011010101 1	10011110000100100110102
	11011000001110010110102	11111001010000110011001110111000 1	11100100000111001010002
	100111100110110000	00000111100100011001010110 5	100010110111101111
	00000000101111110110012	01110001011100011101110001011000 0	11101001001100000001107
	11011001110011000101107	10001000111100001100101010010111 0	01100101100001011101015
	11101101100111001010102	10110010001010000001101001100011 0	11110011000011110111007
	10111011000011100100012	10010011011011100011100110101011 1	00111100001001001101015
	10110000011100101101012	11110010100001100111001110111000 0	11001000001110010100012
	001111001101100001	00000111001000110010110101 1	000101101111011111

where NLF' is a Boolean function including all monomials of NLF, in which variables a_{67}, a_{127} do not exist. Note that the adversary does not need to solve the equation. Instead, the adversary can recover four bits of the secret key by choosing appropriate bits for IVs. For example, if

$$\begin{cases} b_i = 0 & i \in \Phi \\ b_{158} = 1 \end{cases}$$

where $\Phi = \{0, 14, 37, 41, 49, 51, 93, 107, 120, 134, 136, 155, 176, 23, 53, 77, 81, 103, 128\}$, then $b_{192} = a_{41}$. Consequently, $\widehat{iv}_{191} = k_{41}$. In this way, the adversary is able to retrieve the four secret key bits. The number of the required related pairs (K, IV) is 4. On the average, to find the valid pairs, the adversary needs 16 pairs. In other words, to retrieve 4 secret key bits, the adversary should run the initialisation algorithm 16 times for the related (K, IV) pairs. Now, the adversary can keep going and continue the attack finding consecutive 4-bit parts of the secret key. Finally, to determine the whole 128-bit secret key, the adversary needs to apply $512 = 32 \times 16$ related (K, IV) pairs on the average.

4 Weak (K, IV) Pairs

In this section we study the security implications of short cycles of the two registers A and B. Note that the initialisation procedure takes K and IV, loads them to A and B, respectively and then the cipher is clocked 448 times. At the end of the initialisation, the cipher can be set in the following weak states:

- the register A contains all ones and the state loops forever. To identify the collection of pairs (K, IV) that leads to this state of A, it is enough to set $A = \mathbf{1}$ and to set B to an arbitrary 192-bit vector and clock backwards. This process will generate 2^{192} pairs (K, IV) that leads the initialisation to weak states.
- the register B contains all zeros and the state loops forever. Again, to identify the collection of pairs (K, IV) that leads to this state of B, it is enough to set $B = \mathbf{0}$ and to set A to an arbitrary 128-bit vector and clock backwards. This process will generate 2^{128} pairs (K, IV) that leads the initialization to weak states.
- both registers $A = \mathbf{1}$ and $B = \mathbf{0}$. There is a single pair of (K, IV) only. To identify it, set registers appropriately and clock backwards. This case is not very interesting as it can be easily identified.

4.1 Weak (K, IV) Pairs Leading to $A = 1$

It may happen that after the initialisation, the pair (K, IV) leads to $A = \mathbf{1}$. An immediate consequence of this is that the register A contains all ones and it stays in this state for all clocks. The adversary is able to identify this case and they are also able to recover the weak pair (K, IV) that has led to $A = \mathbf{1}$. Clearly, if the adversary knows IV, then the task of finding K is easier.

Note that the cipher with the register A in the state of all ones is equivalent to a 192-bit LFSR whose outputs are filtered by a non-linear Boolean function h with an 6-bit input. The function h is the non-linear function NLF with two bits set to ones (those that are coming from A). The function is a balanced function from $h : GF(2^6) \to GF(2)$ of degree 5 and non-linearity 20 and it is given below.

$$\begin{aligned} h(x_1, x_2, x_3, x_4, x_5, x_6) = {} & 1 \oplus x_1 \oplus x_1x_2 \oplus x_3 \oplus x_1x_3 \oplus x_1x_4 \oplus x_3x_4 \\ & \oplus x_2x_3x_4 \oplus x_5 \oplus x_1x_2x_5 \oplus x_2x_3x_5 \oplus x_1x_4x_5 \\ & \oplus x_3x_4x_5 \oplus x_1x_3x_4x_5 \oplus x_2x_3x_4x_5 \oplus x_1x_6 \oplus x_2x_6 \\ & \oplus x_1x_3x_6 \oplus x_1x_2x_3x_6 \oplus x_4x_6 \oplus x_1x_4x_6 \oplus x_1x_2x_4x_6 \\ & \oplus x_3x_4x_6 \oplus x_1x_3x_4x_6 \oplus x_2x_3x_4x_6 \oplus x_1x_2x_3x_4x_6 \\ & \oplus x_5x_6 \oplus x_2x_3x_5x_6 \oplus x_4x_5x_6 \oplus x_2x_4x_5x_6 \\ & \oplus x_1x_2x_4x_5x_6 \oplus x_2x_3x_4x_5x_6 \end{aligned}$$

The function can be approximated by a linear Boolean function $1 \oplus x_1 \oplus x_1 \oplus x_6$ with probability:

$$Pr(h = (1 + x_1 + x_2 + x_6)) = \frac{44}{64} = 0.6875 = 0.5 + 2^{-2.415}$$

The algebraic immunity of the function is 3 and the number of the annihilators is 10. To recover the contents of the register B, we may apply a basic algebraic attack that needs $2^{22.75}$ observations of the keystream bits and whose complexity is $2^{63.87}$. Once the adversary knows the contents of B at the end of the initialization, he can clock backwards to recover the weak pair (K, IV).

4.2 Weak (K, IV) Pairs Leading to $B = \mathbf{0}$

The second class of weak (K, IV) pairs leads to the state with $B = \mathbf{0}$. In this case the register B stays in the zero state for all clocks. Consequently, all the outputs of DLFSR are zeros, which is equivalent to removal of the register B from the cipher. The goal of the adversary is to recover the pair (K, IV). Now we show that the adversary is able to recover the initial state (and the secret key by clocking NLFSR backwards) faster than in 2^{54} steps.

Note that the NLF function is now used with its 6 bits coming from the register B set to zero. Consequently, the keystream output function that is a linear combination of the least significant bit of the register A with the output of the NLF function. The keystream output function is denoted by $\ell : \{0,1\}^3 \to \{0,1\}$ and is of the following form:

$$l(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_1 x_2 \oplus x_3.$$

The function ℓ is a non-linear balanced Boolean function of degree 2. One of the best approximations of ℓ is the linear function x_3. It is easy to check that

$$Pr(l = x_3) = \frac{6}{8} = 0.75 = 0.5 + 2^{-2} \tag{5}$$

Distinguishing Attack. If $B = \mathbf{0}$, then the adversary may distinguish the generated keystream bits from a random bit generator. Consider the keystream output bits at clocks $t+0$, $t+6$, $t+7$, $t+11$, $t+16$, $t+28$, $t+36$, $t+45$, $t+55$, $t+62$. If we use the approximation (see Equation (5)) then we can write

$$\begin{aligned} Pr(z_{t+128} &= g(z_{t+0}, z_{t+6}, z_{t+7}, z_{t+11}, z_{t+16}, z_{t+28}, z_{t+36}, z_{t+45}, z_{t+55}, z_{t+62})) \\ &\approx 0.502 \end{aligned} \tag{6}$$

This means that the adversary requires around 2^{17} observations of the keystream output bits to tell apart the cipher from a random bit generator with negligible error probability.

Recovery Attack. To recover the pair (K, IV), the adversary may use the linear approximation of ℓ and try to guess the contents of A. The probability of the correct guess for the state is $(0.75)^{128} = 2^{-53.12}$, which is much smaller than the probability 2^{-128}. In other words, the cipher has at most 54 bits of security.

5 Conclusions

In this paper, we analysed the initialisation algorithm of the Rakaposhi stream cipher. We started from observations about cryptographic weak points of the cipher. We discovered the so-called sliding property of the pairs (K, IV). This property can be exploited by launching distingushing and key recovery attacks. We showed that there is a distinguishing attack that needs four related (K, IV) pairs only. Our key recovery attack recovers all bits of the secret key K after observing 2^9 related (K, IV) pairs.

In the second part of the work, we studied the security of Rakaposhi when either the register A or B enters a short cycle at the end of the initialisation procedure. When the register A loops in the all-ones state, then the adversary is able to recover the pair (K, IV). Rakaposhi in this case degenerates to a LFSR cipher with a non-linear filter function. It is shown that the initial state of the register B can be discovered by an algorithm of time complexity $2^{63.87}$.

If the register B enters the zero state at the end the initialisation procedure, then we showed two efficient algorithms: one to distinguish Rakaposhi from a random bit generator and the other to recover the pair (K, IV). The distinguisher needs 2^{17} keystream bit observations. The key recovery algorithm requires around 2^{54} operations. Note that this cryptographic weakness can be explored by the adversary when they have access to the cipher device and are allowed to play with the device by running it for different IVs.

References

1. Cid, C., Kiyomoto, S., Kurihara, J.: The RAKAPOSHI stream cipher. In: Qing, S., Mitchell, C.J., Wang, G. (eds.) ICICS 2009. LNCS, vol. 5927, pp. 32–46. Springer, Heidelberg (2009)
2. De Cannière, C., Küçük, Ö., Preneel, B.: Analysis of grain's initialization algorithm. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 276–289. Springer, Heidelberg (2008)
3. Englund, H., Johansson, T., Sönmez Turan, M.: A framework for chosen IV statistical analysis of stream ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)
4. Filiol, É.: A new statistical testing for symmetric ciphers and hash functions. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 342–353. Springer, Heidelberg (2002)
5. Hell, M., Johansson, T., Meier, W.: Grain - a stream cipher for constrained environments. ECRYPT Stream Cipher Project (2005)
6. Hell, M., Johansson, T., Meier, W.: Grain - a stream cipher for constrained environments. Int. J. Wire. Mob. Comput. 2, 86–93 (2007)
7. Isobe, T., Ohigashi, T., Morii, M.: Slide cryptanalysis of lightweight stream cipher RAKAPOSHI. In: Hanaoka, G., Yamauchi, T. (eds.) IWSEC 2012. LNCS, vol. 7631, pp. 138–155. Springer, Heidelberg (2012)
8. Saarinen, M.-J.O.: Chosen-iv statistical attacks on estream stream ciphers. In: eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, pp. 5–19 (2006)
9. Zhang, H., Wang, X.: Cryptanalysis of stream cipher grain family. In: Cryptology ePrint Archive, Report 2009/109 (2009)

Improved Algebraic and Differential Fault Attacks on the KATAN Block Cipher

Ling Song and Lei Hu

State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences,
Beijing 100093, China
{lsong,hu}@is.ac.cn

Abstract. Improved algebraic attack and differential fault attack on the KATAN block cipher are presented. In the SAT-based algebraic analysis, we improve the ANF-to-CNF conversion to make good use of short equations in the algebraic representation of the cipher. An optimal number of plaintext/ciphertext pairs with a certain structure are used, and 84, 70, and 65 rounds of KATAN32, KATAN48, and KATAN64 are broken, respectively, which are 5 more rounds of the cipher than previous works under the same attack scenario. In the differential fault attack, a new method of recovering secret key bits from faulty and fault-free ciphertexts is developed under one-bit and two-bit fault models, and its iteration application can retrieve the whole 80-bit secret key of the full-round KATAN32, KATAN48, and KATAN64 with 132, 44, and 52 fault injections under the one-bit fault model and with 140, 60, and 60 fault injections under the two-bit fault model, respectively. The time complexity of the attack is negligible, which is a great improvement on previous differential fault attacks on KATAN of time complexity 2^{59}, 2^{55}, and 2^{55} and with 115, 211, and 278 fault injections, respectively, under the one-bit fault model.

Keywords: KATAN block cipher, algebraic attack, differential fault attack, SAT solver.

1 Introduction

KATAN is a lightweight block cipher proposed in 2009 at the cryptographic hardware and embedding system (CHES) conference [10]. It has 80-bit keys and three choices of small block size as 32, 48, or 64 bits, and gets a highly compact design structure by iterating simple nonlinear functions 254 rounds, aiming to achieve an amazingly efficient hardware implementation and provide cryptographic building blocks for constrained devices such as RFID tags.

Since the publication, several attacks on KATAN have been proposed in the literature. The first cryptanalytic result is a SAT-based algebraic attack on reduced-round KATAN [5], which exploits a special preprocessing procedure and breaks 79-round KATAN32, 64-round KATAN48 and 60-round KATAN64 with SAT solvers in a complexity lower than brute force search. The second

R.H. Deng and T. Feng (Eds.): ISPEC 2013, LNCS 7863, pp. 372–386, 2013.

cryptanalytic result is a differential fault attack on KATAN [1], in which one-bit fault model is considered and the cube technique [11] is used to determine fault positions and extract low degree polynomial equations to retrieve secret key bits. The attack achieves a complexity of $2^{59}, 2^{55}, 2^{55}$ for KATAN32, KATAN48 and KATAN64, respectively. Also there are two differential attacks on KATAN [14,2]. The former one breaks 78-, 70-, and 68-round KATAN32, KATAN48, and KATAN64, respectively. The latter one uses a framework which unifies several standard differential attacks. However, this framework requires the entire code-book, which makes the framework less practical.

Algebraic cryptanalysis combined with SAT solving is a developing tool for analyzing stream and block ciphers and even for evaluating hash functions [13]. In this paper we present a better SAT-based algebraic attack on KATAN, in which we improve the method of converting the algebraic expressions derived from the cryptographic system into CNF clauses, enabling us to speed up the SAT solving. Other strategies have been explored also, and we found that multiple pairs of plaintexts/ciphertexts can benefit greatly the SAT solving and there exists an optimal number for such pairs. As a consequence, our SAT-based algebraic attack can break reduced-round KATAN with more rounds than [5]. A phenomena we discovered in the experiments is that for the equation system derived from the cipher, when the optimal number of plaintext/ciphertext pairs are used, the ratio between the number of equations and the number of variables is always close to a constant $\lambda = 1.21$.

Besides the SAT-based algebraic attack, we propose an improved differential fault attack on KATAN. Differential fault analysis was developed by Biham and Shamir [6] as a type of implementation-based attack. The model for fault attack we used in this paper is the transient one-bit fault model as used in [1], which assumes that the adversary can choose the target round and induce one bit of fault into the internal state in the execution of a cipher but without damaging the bit position permanently. Under the one-bit model, we fully take advantage of additional equations derived from fault ciphertexts to retrieve secret key bits with a procedure like what we use in the SAT-based algebraic attack. We observed that if last continuous subkey bits are recovered, an iterative procedure can be applied to decrypt back to find more subkey bits. Concretely, four times of iterations and 132, 44, 52 faults are required to retrieve the secret key of full-round KATAN32, KATAN48 and KATAN64 respectively. Furthermore, utilizing ciphertext differentials in a slightly more complicated way, we develop a two-bit fault model. Under this new model 140, 60, 60 faults are enough to get the secret key for full-round KATAN32, KATAN48 and KATAN64 respectively. Both the two fault attacks have a negligible time complexity, which is a great improvement since the previous work [1] requires a time complexity more than 2^{55}.

This paper is organized as follows: Section 2 briefly describes the KATAN block cipher; Section 3 presents our SAT-based algebraic attack on KATAN; in Section 4 we elaborate on our new differential fault attack on KATAN; and finally, the last section is the conclusion.

2 Description of KATAN

KATAN is a lightweight, hardware-oriented block cipher, consisting of three suggested versions, KATAN32, KATAN48, and KATAN64, where $n = 32, 48$, or 64 and indicates the block size of the cipher [10]. All these versions accept 80-bit keys and iterate 254 rounds with the same nonlinear functions.

As an example KATAN32 is described, KATAN48 and KATAN64 will be then explained by their differences with KATAN32.

KATAN32 receives a 32-bit plaintext and outputs a 32-bit ciphertext. Blocks of bits are always numbered in right-to-left order starting from 0. A 32-bit plaintext $P = (p_{31} \cdots p_1 p_0)$ is loaded into two registers L_1 and L_2 of length $|L_1| = 13$ and $|L_2| = 19$, respectively, where the least significant bit p_0 is loaded to the least significant bit of L_2, while the most significant bit P_{31} is loaded to the most significant bit of L_1, namely $(p_{31} \cdots p_1 p_0) = (L_1[12] \cdots L_1[0] L_2[18] \cdots L_2[0])$. At each round, two nonlinear functions are evaluated as follows:

$$f_a(L_1) = L_1[x_1] + L_1[x_2] + (L_1[x_3] \cdot L_1[x_4]) + (L_1[x_5] \cdot IR) + k_a,$$
$$f_b(L_2) = L_2[y_1] + L_2[y_2] + (L_2[y_3] \cdot L_2[y_4]) + (L_2[y_5] \cdot L_2[y_6]) + k_b,$$

where IR is a round-dependent constant which makes use of $L_1[x_5]$ whenever $IR = 1$ (see [10] for its details); $x_1, \cdots, x_5$ and $y_1, \cdots, y_6$ are indices in the intervals $[0, 12]$ and $[0, 18]$, respectively (see Table 1); k_a and k_b are two subkey bits, $k_a = k_{2i}$ and $k_b = k_{2i+1}$ for the i-th round. Then the two registers L_1 and L_2 are left shifted and $f_a(L_1)$ and $f_b(L_2)$ are loaded into $L_2[0]$ and $L_1[0]$, respectively.

For KATAN48, at each round $f_a(L_1)$ and $f_b(L_2)$ are computed twice with the same pair of subkey bits k_a and k_b and the registers L_1 and L_2 are shifted twice. For KATAN64, $f_a(L_1)$ and $f_b(L_2)$ are applied three times per round, again with the same subkey bits k_a and k_b, and the registers L_1 and L_2 are shifted three times. The indices x_i and y_i are listed in Table 1.

Table 1. Parameters of KATAN

Cipher	$\|L_1\|$	$\|L_2\|$	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5	y_6
KATAN32	13	19	12	7	8	5	3	18	7	12	10	8	3
KATAN48	19	29	18	12	15	7	6	28	17	21	13	15	6
KATAN64	25	39	24	15	20	11	9	38	25	33	21	14	9

The KATAN cipher uses the same key schedule. Let $K = (k_{79} \cdots k_1 k_0)$ be the secret key, the subkey bits are generated as follows:

$$k_i = \begin{cases} k_i, & \text{for } 0 \le i \le 79, \\ k_{i-80} + k_{i-61} + k_{i-50} + k_{i-13}, & \text{otherwise.} \end{cases}$$

More details about the KATAN cipher can be found in [10].

3 Improved SAT-Based Algebraic Attack on KATAN

Algebraic cryptanalysis is a type of attack that depends on solving a system of multivariate quadratic equations (by introducing additional variables and equations, any system of equations can be represented as a quadratic equation system) derived from a given cipher. It is known that the problem of solving a multivariate simultaneous system of quadratic equations (the MQ problem) is NP-hard. The SAT problem, finding a satisfying assignment for a logical expression, is also NP-hard. In [4], Bard, Courtois and Jefferson studied methods and techniques to solve the MQ problem derived from ciphers with SAT solvers. Their idea is based on the fact that the MQ and SAT problems are polynomially equivalent while the SAT problem is more well studied. Therefore they introduced a method (called the BCJ method below) to convert an MQ problem (over GF(2)) to a SAT problem. Later, this method was slightly extended in the GoS (Grain of Salt) tool [17], and has been applied to cryptanalyze Keelog [8], Bivium [15] and PRINT-48 [7], making SAT solvers enjoy a boom in the area of cryptanalysis.

In [5], Bard et al analyzed KATAN with SAT solvers using BCJ conversion. Before conversion, they simplified the algebraic equations with a preprocessing procedure, which speeded up the SAT solving indirectly. They also found that if the plaintexts have some special structures, the attack is much stronger. In this section, we do some further research along this line. We improve the preprocessing procedure and try a different conversion method. Experiments show a better result than [5].

3.1 Preliminaries on SAT Solving

The BCJ Conversion Technique. A common input format of SAT solvers is the Conjunctive Normal Form (CNF), and a common ANF-to-CNF transformation used in cryptanalysis is the BCJ conversion. From now on $\bar{x}$ denotes the complement of binary variable x. Given an algebraic equation representation of a cryptosystem, the BCJ conversion proceeds in two major steps. First, substitute every nonlinear terms in the algebraic equation with new variables to get a linear system. Second, transform the linear system and nonlinear terms into CNF clauses.

As an example, consider an equation

$$abc + ce + a + d + e = 0.$$

First, we introduce new variables x for abc and y for ce, we get

$$\begin{aligned} x + y + a + d + e &= 0, \\ abc &= x, \\ ce &= y. \end{aligned}$$

Second, convert the above equations to logical formulas with the following two techniques.

Technique 1: $abc = x$ is converted to

$$(a \vee \bar{x}) \wedge (b \vee \bar{x}) \wedge (c \vee \bar{x}) \wedge (x \vee \bar{a} \vee \bar{b} \vee \bar{c}).$$

Therefore, each monomial of degree $d > 1$ requires $d + 1$ clauses to describe.

Technique 2: $x + y + a + d + e = 0$ is converted to

$$(\bar{x} \vee y \vee a \vee d) \wedge (x \vee \bar{y} \vee a \vee d) \wedge (x \vee y \vee \bar{a} \vee d) \wedge (x \vee y \vee a \vee \bar{d})$$
$$\wedge (\bar{x} \vee \bar{y} \vee \bar{a} \vee d) \wedge (\bar{x} \vee \bar{y} \vee a \vee \bar{d}) \wedge (\bar{x} \vee y \vee \bar{a} \vee \bar{d}) \wedge (x \vee \bar{y} \vee \bar{a} \vee \bar{d}).$$

If the equation is an XOR clause of l variables, the conversion requires

$$\binom{l}{1} + \binom{l}{3} + \cdots + \binom{l}{2\lfloor l/2 \rfloor - 1} = 2^{l-1}$$

CNF clauses since all arrangements of l variables with odd number (less than l) of complements are needed to consider.

Since the number of clauses for linear equations grows exponentially with the length l, it is better to introduce more new variables and cut down the length. Note that the longest length of equations permitted after cutting is called the *cutting number* in [4], and we can also choose 3, 4, 5 or 6 as the cutting number.

Preprocessing Procedure. The time complexity of SAT solving closely relates to the sparsity of the target system of algebraic equations, the numbers of variables and clauses of the corresponding CNF problem. Therefore we hope to derive a sparse algebraic equation system and transform it to a CNF problem with as less variables and clauses as possible.

To this goal, Bard et al proposed a preprocessing procedure [5]. For a system of polynomial equations over GF(2) and a threshold W, this procedure first spots the shortest equation of length (the number of nonconstant terms in the equation) $\leq W$, then chooses a monomial from this equation and eliminates it with this equation. This procedure iterates until no equation of length $\leq W$ can be found. Consequently, the substitution in the processing procedure reduces the number of variables in the equation system. When $W = 1$, this preprocessing procedure definitely benefits the conversion to get a small CNF problem. However, when W is chosen as other values, we are not sure about the size of the resulting CNF problem, but the experiments in [5] showed $W = 2$ turns out to be the best in a statistical sense.

We use pre(W) to denote this preprocessing procedure.

Fixing Variables. The guess-and-determine trick for SAT based algebraic attack was initially proposed in [4]. It means that with f key bits fixed, a good attack requires to recover the rest key bits faster than brute force search, i.e.,

$$t_{\text{brute force}} > t_{\text{SAT}} \cdot 2^f.$$

During the experiments, f is set to be a value such that the time of SAT solving is reasonable, for example in an hour. Our following experiments generally choose $f = 45$ if no specific indication is stated.

SAT Solvers for Cryptanalysis. Two most commonly used SAT solvers in cryptanalysis are MiniSat [12] and CryptoMiniSat [18]. Utilizing some ideas of MiniSAT, CryptoMiniSat is recently developed and it adds a lot of new heuristics to handle XOR clauses directly, i.e., we can feed XOR clauses into CryptoMiniSat without any conversion. Thus we can develop another conversion which conforms with BCJ except the XOR clause transformation part. This conversion is called the XOR conversion below.

3.2 Attack Strategies

Algebraic Description of KATAN. We give an algebraic description of KATAN that resembles the *Floating Representation* in [16]. Specifically, we introduce intermediate variables for every subkey bit and state bit, and generate equations in the way described in Algorithm 1, where we take KATAN32 as an example.

Algorithm 1. GenerateEquations(n, p, c)

Input: round number n, plaintext p and the corresponding ciphertext c
Output: equations describing the cryptosystem

$E \leftarrow \emptyset$;
for $i = 80$ *to* n **do**
  $E \leftarrow E \cup \{k_i + k_{i-80} + k_{i-61} + k_{i-50} + k_{i-13}\}$;
end
$L_1||L_2 \leftarrow p$;
for $i = 0$ *to* n **do**
  $E \leftarrow E \cup \{s_{2i} + f_a(L_1),\ s_{2i+1} + f_b(L_2)\}$;
  $(L_1[12], L_1[11], \cdots, L_1[1], L_1[0]) \leftarrow (L_1[11], L_1[10], \cdots, L_1[0], f_b(L_2))$;
end
Substitute$(E,\ L_1||L_2 : c)$;
return E ;

Following Algorithm 1, an instance of KATAN32 is represented by an algebraic system with 936 equations and 984 variables. The size of this algebraic system is smaller than the system derived in [5], which needs 8620 equations and 8668 variables.

MiniSat or CryptoMiniSat. We have conducted experiments to evaluate which SAT solver outperforms the other for analyzing KATAN. Results show that CryptoMiniSat is better when multiple plaintext/ciphertext pairs are used, see Table 2. It can be concluded that CrytpoMiniSat makes good use of relationship inside the problem. However, this does not comply with the report of [5] that MiniSat runs faster than CryptoMiniSat.

Cutting Number. The choice of cutting number influences the resulted CNF problem. In the literature the number 6 is a common choice and it is also the case in our experiments. See Table 3 for the comparison of SAT solving timing for the cutting number 4, 5 and 6 (3 is too small).

Table 2. MiniSat vs. CryptoMiniSat

#rounds	#fixing bits	#pairs	MiniSat	CryptoMiniSat
66	30	1	42.68	64.10
66	30	5	0.26	0.15
78	45	20	123.31	5.19

Note: Using the BCJ conversion, average CPU time (in seconds) on 50 instances

Table 3. Comparison for cutting numbers

conversion	4	5	6
BCJ	80.08	62.89	52.21
XOR	200.73	159.10	57.44

Note: For 70-round KATAN32, fixing 30 key bits, 10 P/C pair, average CPU time (in seconds) on 50 instances.

Number of Plaintext/Ciphertext Pairs. The equation system of KATAN is underdefined if only one plaintext/ciphertext pair is used because of the key size larger than the block size. This problem can be remedied by the use of multiple plaintext/ciphertext pairs. However, if too many plaintext/ciphertext pairs are used, the size of the resulted CNF problem will grow large quickly, which makes it difficult to solve by SAT solvers. Then it comes to a question that how many pairs are optimally needed. Our experiments show that there exists an optimal number for plaintext/ciphertext pairs, and this optimal number is increasing with the number of rounds, as depicted in Figure 1 for KATAN32.

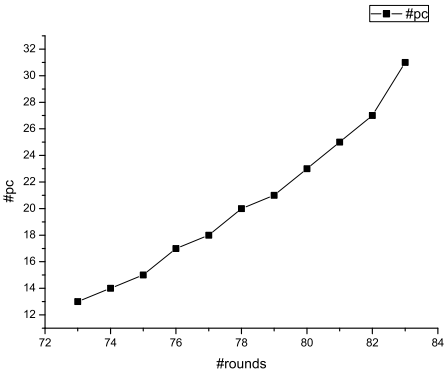


Fig. 1. Optimal number of plaintext/ciphertext pairs

Different Structures of Plaintexts. It is found that chosen-plaintext attack is much stronger than known-plaintext attack in [5], where the chosen plaintexts follow some specific structure. We explored several other plaintext structures.

The plaintext structures we have evaluated are: (a) random plaintexts; (b) specific plaintext structure used in [5]; (c) plaintexts with one bit modular difference at the 19-th bit position; (d) conditional differential from [14]. Conversion methods we considered are pre(W)+BCJ and pre(W)+XOR where $W \leq 4$. The results are illustrated in Table 4.

Table 4. Different structures of plaintexts

structure	median	mean	best method
(c)	3.29	4.81	pre(1)+ XOR
(b)	3.85	35.91	pre(2)+ BCJ
(a)	4.01	45.23	pre(2)+ BCJ
(d)	5.37	23.69	pre(1)+ XOR

Note: For 78-round KATAN32, 20 P/C pairs, average CPU time (in seconds) on 50 instances.

Since the preprocessing procedure may speed up or slow down the SAT solving, we need to run the experiments many times and analyze the running time statistically. Note that these combinations of preprocessing procedure and conversion are evaluated first on the median of running time, then on the mean of running time. As we can see from Table 4, structure (c) is overwhelmingly better than random structure, and (d) is worse than random structure on a large part of the instances since its median is larger than that of random structure. As a result, it can be concluded that certain structure of plaintexts can speed up the SAT solving, but the best structure is difficult to find.

Enhanced Conversions. Due to the use of fixing key bits and the use of multiple plaintext/ciphertext pairs, many short equations, from which the Bard et at's preprocessing procedure benefits, appear in the algebraic description of the cipher. However, when $W \geq 2$, the resulted CNF problem does not always have less clauses. We exemplify this as follow. Let $x + y + 1 = 0$ be a target equation under processing. Set $x = y + 1$ and substitute x wherever it appears in the rest equations, say $xz + w + y = 0$, then we get $yz + w + y + z = 0$ with length 4, which is greater than the length of the original equation. As a consequence, the number of clauses is not always reduced with the preprocessing procedure.

We observe that $y + 1 = \bar{y}$ as long as the equations are over GF(2). Thus, during the conversion the equation $yz + w + y + z = 0$ can be regarded as $\bar{y}z + w + y = 0$ or $y\bar{z} + w + z = 0$, whose length is only 3. With this in mind, we modified the basic BCJ conversion to get an enhanced one named BCJ-E. The superiority of BCJ-E over BCJ can be clearly demonstrated for $W = 2$, as in Table 5. For $W > 2$, both BCJ and BCJ-E are much worse than the case when $W \leq 2$.

Similarly, the XOR conversion can be improved to a new conversion named XOR-E in the same way. For $W \leq 1$, XOR-E and XOR are almost the same, while for $W \geq 2$, experiments showed that XOR-E outperforms XOR as can be seen from Table 6. However, for both of XOR-E and XOR, they are less efficient in the case of $W > 1$ than in the case of $W \leq 1$.

Table 5. BCJ Vs. BCJ-E

conversion	# vars	# clauses	SAT solving time
BCJ	3146	41905	61.21
BCJ-E	3120	39764	5.87

Note: For $W = 2$, 78-round KATAN32, 20 P/C pairs with structure (c), average CPU time (in seconds) on 50 instances.

Table 6. XOR Vs. XOR-E

conversion	# vars	# clauses	SAT solving time
XOR	3268	7500	97.80
XOR-E	3091	7392	29.11

Note: For 78-round KATAN32, 20 P/C pairs with structure (c), average CPU time (in seconds) on 50 instances.

3.3 Attack Results

All experiments are carried out with Sage 4.8 on a Personal Computer with Intel(R) Core(TM) Quad CPU (2.83GHz, 3.25GB RAM, Linux). The BCJ conversion is adapted from the ANF-to-CNF converter of Martin Albrecht and Mate Soos [3]. The two solvers we used are CryptoMiniSat 2.9.2 and MiniSat 2.2.0.

Our experiments showed that CryptoMiniSat is faster than MiniSat for analyzing KATAN. Thus, it can be concluded that CryptoMiniSat is more dedicated for cryptography and has good performance in utilizing strong internal relationship of a problem. Next, given certain rounds of the cipher, there exists an optimal number of plaintext/ciphertext pairs which lead to SAT solving faster than others. Also, an efficient structure of plaintext is found. The results showed that the structure of plaintexts has a great influence on SAT solving, but it is not clear yet how it influences. Finally, we improve the conversion method so that it can speed up the SAT solving when used together with Bard et al's preprocessing procedure.

To make a clear comparison with exhaustive key search, we use the assumption in [5] that each round of KATAN takes 3 CPU cycles. The results of our algebraic attack with CryptoMiniSat are presented in Table 7 (only the results in this table are computed with 4 threads, and all others with 1 thread). The comparison between our results and the previous results in [5] is listed in Table 8, where 79 is the largest number of rounds the attack of [5] can do faster than brute force search.

An Experimental Phenomena. At the end of this section, we present a phenomena we have observed in our experiments. Let m and n denote the numbers of equations and of variables in the equation system to be solved with a SAT solver, respectively. We discovered that the ratio $\lambda = \frac{m}{n}$ is almost a constant for different instances when the experimental optimal number of plaintext/ciphertext

Table 7. The results of attacking KATAN with CryptoMiniSat solver

version	# rounds	brute force	mean	median	brute force/mean
KATAN32	84	764.90	289.92	268.83	2.64
KATAN48	70	1274.83	345.27	235.35	3.69
KATAN64	65	1775.66	282.23	193.43	6.29

Note: Average CPU time (in seconds) on 50 instances.

Table 8. The attack timing of 79-round KATAN32

source	method	structure	brute force	mean	median	brute force/mean
[5]	pre(2)+BCJ	(b)	2714.42	1146.77	873.24	2.37
this	XOR	(c)	2877.47	13.02	9.57	221.00

Note: For 20 P/C pairs, average CPU time (in seconds) on 50 instances. The last column denotes the ratio of the mean timings to brute force with respect to the attack.

pairs are used. For three versions of KATAN, the three corresponding λ are slightly different but all are very close to 1.21.

4 Differential Fault Attack on KATAN

A differential fault analysis (DFA) is a method to analyze a cipher by affecting its implementation. The idea is to induce a physical corruption to the internal state of the execution of the cipher, which leads to producing some information about the internal data and helping to recover the secret key of the cipher. In this section, we present differential fault attack on the KATAN block cipher with improved methods developed from [1], [16].

4.1 Related Works

The first related work on KATAN is a differential fault attack under transient one-bit fault model [1]. In this model, it is assumed that the adversary can induce one bit error into the internal state of a cipher during its execution (e.g., using a laser beam) without damaging the bit position permanently, and that the adversary can choose the target round in which faults should be induced. Using this model, what the adversary should do first is to determine the position of the faulty bit within the internal state. To find the exact position, they construct a differential characteristic for each internal state bit with one-dimension cubes (equivalent to a standard differential), which originates from [11]. Further, they utilize the cube summation method again to extract low degree polynomial equations, from which part of the key bits are retrieved.

Another related work is a differential fault attack on the stream cipher Trivium [16], where the cipher is described with the *Floating Representation*. Besides the equations derived from correct key streams, additional low degree polynomial

equations from faulty key streams are generated in a way slightly different from float representation. After algebraic equations are generated, a SAT solver is used to recover internal state bits.

Based on the two works mentioned above, we propose a new differential fault attack on full-round KATAN, which exploits a straightforward method to locate fault positions, recovers the secret key in only a negligible time and requiers less fault injections than the previous attack in [1].

4.2 Fault Position Determination

We still take KATAN32 as an example below. During the execution of the cipher, the adversary injects a fault at the t-th round. We denote the state bits at the t-th round by $s^{(t)} = (s_{31}^{(t)}, \cdots, s_0^{(t)})$, and the subkey bits used since round t by $k^{(t)} = (k_{2t}, k_{2t+1}, \cdots, k_{506}, k_{507})$. Then the 32-bit ciphertext can be represented by $s^{(t)}$ and the subkey bits k^t as $c = (c_{31}, \cdots, c_1, c_0) = \big(f_{31}(s^{(t)}, k^{(t)}),\ \cdots,\ f_1(s^{(t)}, k^{(t)}),\ f_0(s^{(t)}, k^{(t)})\big)$. We define the differential of c_i and the differential characteristic respect to a certain fault position p at round t in a way which is more straightforward than that used in [1].

Definition 1. *Assume that a fault is induced at the p-th ($0 \le p \le 31$) bit of the inner state at round t. The differential of c_i is determined by $f_i(s_p^{(t)}, k^{(t)}) + f_i(s_p^{(t)} + 1, k^{(t)})$. If $f_i(s_p^{(t)}, k^{(t)}) + f_i(s_p^{(t)} + 1, k^{(t)}) = 0$, $\Delta c_i = 0$; if $f_i(s_p^{(t)}, k^{(t)}) + f_i(s_p^{(t)} + 1, k^{(t)}) = 1$, $\Delta c_i = 1$; otherwise, let $\Delta c_i = 2$.*

Definition 2. *Assume that a fault is induced at the p-th ($0 \le p \le 31$) bit of the inner state at round t. The differential characteristic of position p at round t is $\Delta C_p = (\Delta c_{31}, \cdots, \Delta c_0)$.*

Naturally, we can determine the position p at round t only when the ΔC_p are different from each other. If the ΔC_p are different, round t is called *valid* for fault attack. For a valid round t we can build a lookup table, of which each item consists of a differential characteristic and the corresponding fault position. Given a ΔC, we can get the fault position p through this table. Note that we hope a small valid t can be obtained, since for smaller t more subkey bits are involved after fault injection. Furthermore, the attack is more practical if t can be chosen in a less restricted way. According to our experiments, a valid value for t is $t \ge 231$ for KATAN32, 232 for KATAN48, and 234 for KATAN64.

In our model one faulty bit can be induced. We also extend the one bit fault differential characteristic to two-bit fault differential characteristic based on the result of $f_i(s_p^t, s_{p+1}^t, k) + f_i(s_p^t + 1, s_{p+1}^t + 1, k)$.

Definition 3. *Assume that a two-bit fault is induced at the p-th, $(p+1)$-th ($0 \le p \le 30$) bit of the inner state at round t. The differential characteristic of position $p, p+1$ at round t, $\Delta C_{p,p+1}$, is defined as the sum of two integer vectors ΔC_p and ΔC_{p+1}, which is a vector with entries being 0, 1, and 2, and the entries are calculated according to the rule: $1 + 1 = 0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, and $2 + a = 2 + a = 2$ for any $a = 0, 1, 2$.*

Under the two-bit fault model, we need to distinguish all the ΔC_p and the $\Delta C_{p,p+1}$ with respect to round t. In this case, the smallest valid value for t is 237 for KATAN32, 234 for KATAN48, and 237 for KATAN64.

The fault model can also be extended to multiple-bit fault models. However, the smallest valid round is close to the last round, leading to a small number of key bits involved and recovered.

4.3 Getting Differential Equations

Each fault injection leads to additional equations. As in [16] we use $(\Delta L_1, \Delta L_2)$ to generate differential polynomial equations. By using the fact that

$$\Delta(x \cdot y) = \Delta x \cdot y + x \cdot \Delta y + \Delta x \cdot \Delta y,$$

we can construct the differentials of the functions f_a and f_b as follows:

$$\begin{aligned}
\Delta f_a(L_1) =& \Delta L_1[x_1] + \Delta L_1[x_2] + \Delta L_1[x_3] \cdot L_1[x_4] + L_1[x_3] \cdot \Delta L_1[x_4] \\
&+ \Delta L_1[x_3] \cdot \Delta L_1[x_4] + \Delta L_1[x_5] \cdot IR, \\
\Delta f_b(L_2) =& \Delta L_2[y_1] + \Delta L_2[y_2] + \Delta L_2[y_3] \cdot L_2[y_4] + L_2[y_3] \cdot \Delta L_2[y_4] \\
&+ \Delta L_2[y_3] \cdot \Delta L_2[y_4] + \Delta L_2[y_5] \cdot L_2[y_6] + L_2[y_5] \cdot \Delta L_2[y_6] \\
&+ \Delta L_2[y_5] \cdot \Delta L_2[y_6].
\end{aligned}$$

Assume we inject a one-bit fault. In this case the differential of the inner state at round t,

$$(\Delta L_1[12], \cdots, \Delta L_1[0], \Delta L_2[18], \cdots, \Delta L_2[0])$$

are zeros everywhere except at the fault injection position. For example, suppose that the injected bit is $L_2[5]$ then the differential vector will be

$$(0, \cdots, 0, \Delta L_2[5] = 1, \cdots, 0).$$

Below we explain how to use this model to generate the differential polynomial equations.

As we can see from Algorithm 2, we do not introduce intermediate variables for differential state bits. As a consequence, the 32 differential equations produced by this procedure probably have a degree 7, 8 or larger. Instead of recovering key bits with a SAT solver as in [16], we just pose the preprocessing procedure on equations returned by Algorithm 1 and Algorithm 2 with $W = 2$. However, this preprocessing procedure is slightly modified such that it can extract known subkey bits during substitution.

4.4 Iterative Procedure for Retrieving the Secret Key

We induce faults at valid rounds and use the differential of the corresponding faulty ciphertexts and fault-free ciphertexts to determine fault positions. Then equations of fault-free ciphertexts and differential equations of faulty ciphertexts

Algorithm 2. GenerateDifferentialEquations$(n, t, m, \Delta\mathbf{c}, \mathbf{p})$

Input: the whole round number n, the fault round number t, m ciphertext differential $\Delta\mathbf{c} = (\Delta c_0, \Delta c_1, \cdots, \Delta c_{m-1})$ and the corresponding fault position $\mathbf{p} = (p_0, p_1, \cdots, p_{m-1})$
Output: differential equations of m fault ciphertexts

$E \leftarrow \emptyset$;
for $i = 0$ *to* m **do**
  $\Delta L_1 \leftarrow (0, 0, \cdots, 0)$;
  $\Delta L_2 \leftarrow (0, 0, \cdots, 0)$;
  InjectFault$(\Delta L_1, \Delta L_2, p_i)$;
  for $j = t$ *to* n **do**
    $(\Delta L_1[12], \Delta L_1[11], \cdots, \Delta L_1[1], \Delta L_1[0]) \leftarrow$
    $(\Delta L_1[11], \Delta L_1[10], \cdots, \Delta L_1[0], \Delta f_b(L_2))$;
    $(\Delta L_2[18], \Delta L_2[17], \cdots, \Delta L_2[1], \Delta L_2[0]) \leftarrow$
    $(\Delta L_2[17], \Delta L_2[16], \cdots, \Delta L_2[0], \Delta f_a(L_1))$;
  end
  $E \leftarrow E \cup \{L_1[j] + \Delta c_{j+19}\}_{0 \le j \le 12} \cup \{L_2[j] + \Delta c_j\}_{0 \le j \le 18}$;
end
return E;

can be produced by Algorithms 1 and 2. Applying the preprocessing procedure in Subsection 3.1, part of the subkey bits subsequent to the fault round can be recovered.

In our experiments, 35, 37, 36 subkey bits can be recovered under the one-bit fault model if we induce 60, 23, 34 faults at the smallest valid round of KATAN32, KATAN48, KATAN64, respectively. We observe that among the obtained subkey bits, a great part of them are continuous, e.g., they are $k_{507}, k_{506}, k_{505}, \cdots$. If the last 20 continuous subkey bits are known, we can decrypt 10 rounds back and 10 earlier rounds become valid, i.e., the smallest valid round number changes to $t-10$. Inducing faults at $t-10, t-20, \cdots$, more subkey bits can be recovered. If 80 continuous subkey bits are obtained, the 80-bit secret key can also be recovered by shifting the LFSR back according to the key schedule. For KATAN32, inducing 33 faults at round 231 almost guarantee that the last continuous 20 subkey bits can be obtained, thus inducing 33 faults at round 231, 221, 211, 201 and iteratively running Algorithm 2 4 times, 80 continuous subkey bits can be known with a probability near to 1. Analogously, similar results can be obtained for KATAN48, KATAN64 under even two-bit fault model. The results are described in the next subsection.

4.5 Attack Results

We analyzed the KATAN32, KATAN48 and KATAN64 with differential fault attack under the one-bit and two-bit fault models respectively. Under both models, we get better attack results than [1] which is the best side channel attack against KATAN to this date. The reason is that the attack in [1] only utilized quadratic

and linear equations of the fault ciphertexts, while we made use of all differential equations. The additional equations may have large degrees, but their degrees may drop after substitution and become useful for us.

We introduce an iterative procedure aiming to retrieve the whole 80-bit secret key. The main steps of this attack are generating equations and making substitution, which take less than one minute. The result of our differential fault attack on KATAN is summarized in Table 9.

Table 9. Differential fault attack on KATAN

size	one-bit [1]	one-bit		two-bit	
	time/#injections	fault rounds	#injections	fault rounds	#injections
32	2^{59}/115	231,221,211,201	132	237,227,217,207,197	140
48	2^{55}/211	232,222,212,202	44	234,224,214,204	60
64	2^{55}/278	234,224,214,203	52	237,227,217,207	60

5 Conclusion

In this paper we have presented improved SAT-based algebraic attack and differential fault attack on the KATAN block cipher. Both the two attacks are currently best under same attack scenarios. In the SAT-based algebraic attack, we explored extensive strategies to speed up SAT solving and for each version of KATAN, 5 more rounds than previous results can be analyzed. Our results also suggested that CryptoMiniSat deserves more attention in cryptanalysis. For differential fault attack, our idea is to recover last continuous subkey bits. As long as last continuous subkey bit are retrieved, an iterative procedure can be constructed to get the whole secret key. This attack takes a negligible time complexity.

Acknowledgement. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. The work of this paper was supported by the National Key Basic Research Program of China (2013CB834203), the National Natural Science Foundation of China (Grants 61070172 and 10990011), the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702, and the State Key Laboratory of Information Security, Chinese Academy of Sciences.

References

1. Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J.: Fault analysis of the KATAN family of block ciphers. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 319–336. Springer, Heidelberg (2012)
2. Albrecht, M., Leander, G.: An All-in-One Approach to Differential Cryptanalysis for Small Block Ciphers. In: IACR Cryptology ePrint Archive, number 401 (2012)

3. Albrecht, M., Soos, M.: ANF2CNF - Converting ANF to CNF for Algebraic Attack Using SAT Solver (2010), http://gforge.inria.fr/frs/?group_id=2330&release_id=5449
4. Bard, G., Courtois, N., Jefferson, C.: Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomial over GF(2) via SAT-Solvers. In: IACR Cryptology ePrint Archive, number 024 (2007)
5. Bard, G.V., Courtois, N.T., Nakahara Jr, J., Sepehrdad, P., Zhang, B.: Algebraic, AIDA/Cube and Side Channel Analysis of KATAN Family of Block Ciphers. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 176–196. Springer, Heidelberg (2010)
6. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key cryptosystem. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
7. Bulygin, S., Buchmann, J.: Algebraic Cryptanalysis of the Round-Reduced and Side Channel Analysis of the Full PRINTCipher-48. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 54–75. Springer, Heidelberg (2011)
8. Courtois, N.T., Bard, G.V., Wagner, D.: Algebraic and slide attacks on keeLoq. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 97–115. Springer, Heidelberg (2008)
9. De Cannière, C.: Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006)
10. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
11. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
12. Een, N., Sorensson, N.: MiniSat v1.13 - A SAT Solver with Conflict-Clause Minimization (2005), http://www.minisatse.com/Papers.html
13. Homsirikamol, E., Morawiecki, P., Rogawski, M., Srebrny, M.: Security Margin Evaluation of SHA-3 Contest Finalists through SAT-Based Attacks. In: IACR Cryptology ePrint Archive, number 421 (2012)
14. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of NLFSE-Based Cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010)
15. McDonald, C., Charnes, C., Pieprzyk, J.: Attacking bivium with Minisat. In: IACR Cryptology ePrint Archive, number 040 (2007)
16. Mohamed, M.S.E., Bulygin, S., Buchmann, J.: Using SAT Solving to Improve Differential Fault Analysis of Trivium. In: Kim, T.-h., Adeli, H., Robles, R.J., Balitanas, M. (eds.) ISA 2011. CCIS, vol. 200, pp. 62–71. Springer, Heidelberg (2011)
17. Soos, M.: Grain of Salt - an automated way to test stream ciphers through SAT solver, http://www.msoos.org/grain-of-salt
18. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT Solvers to Crytpographic Problems. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 244–257. Springer, Heidelberg (2009)

Author Index