Ed Dawson
Duncan S. Wong (Eds.)

Information Security Practice and Experience

LNCS 4464

Third International Conference, ISPEC 2007
Hong Kong, China, May 2007
Proceedings

Springer

Lecture Notes in Computer Science 4464

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

Ed Dawson Duncan S. Wong (Eds.)

Information Security Practice and Experience

Third International Conference, ISPEC 2007
Hong Kong, China, May 7-9, 2007
Proceedings

Volume Editors

Ed Dawson
Queensland University of Technology
Information Security Institute
GPO Box 2434, Brisbane Qld 4001, Australia
E-mail: e.dawson@qut.edu.au

Duncan S. Wong
City University of Hong Kong
Department of Computer Science
83 Tat Chee Ave, Hong Kong, China
E-mail: duncan@cityu.edu.hk

Library of Congress Control Number: 2007924822

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-72159-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-72159-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12055104 06/3180 5 4 3 2 1 0

Preface

The third international conference on Information Security Practice and Experience (ISPEC 2007) was held in Hong Kong, China, May 7 – 9, 2007. The conference was organized and sponsored by City University of Hong Kong.

As applications of information security technologies become pervasive, issues pertaining to their deployment and operation are becoming increasingly important. ISPEC is an annual conference that brings together researchers and practitioners to provide a confluence of new information security technologies, their applications and their integration with IT systems in various vertical sectors. In 2005 and 2006, the first and second conferences were held successfully in Singapore and Hangzhou, China, respectively. The conference proceedings were published by Springer in the *Lecture Notes in Computer Science* series.

The Program Committee received 135 submissions, and accepted 24 papers for presentation. The final versions of the accepted papers, which the authors finalized on the basis of comments from the reviewers, are included in the proceedings. The entire reviewing process took nine weeks, each paper was carefully evaluated by at least three members from the Program Committee. The individual reviewing phase was followed by a Web-based discussion. Papers over which the reviewers significantly disagreed were further reviewed by external experts. Based on the comments and scores given by reviewers, the final decisions on acceptance were made. We appreciate the hard work of the members of the Program Committee and external referees, who gave many hours of their valuable time.

In addition to the contributed papers, there were four invited talks: Bill Caelli spoke on "Application Security—Myth or Reality?", Robert H. Deng on "Towards Efficient and Novel Security Solutions—A Marriage of Crypto and Trusted Computing Platform," Lucas Hui on "Computer Forensics Tools and Technology: Research and Development in Hong Kong" and Victor K. Wei on "E-voting by Zero-Knowledge."

We would like to thank all the people involved in organizing this conference. In particular, we would like to thank colleagues from the Department of Computer Science, City University of Hong Kong, for their time and efforts, as well as Dennis Liu, Chung Ki Li and Qiong Huang for their excellent work on maintaining the submission/reviewing software and taking care of all the technical aspects of the review process. Finally, we would like to thank all the authors who submitted papers to the conference.

May 2007

Ed Dawson
Duncan Wong

Organization

ISPEC 2007 was organized by the Department of Computer Science, City University of Hong Kong, China.

General Chair

Xiaotie Deng	City University of Hong Kong, China
C. H. Lee	City University of Hong Kong, China

Program Committee Co-chairs

Ed Dawson	QUT, Australia
Duncan Wong	City University of Hong Kong, China

Steering Committee

Feng Bao	I2R, Singapore
Robert H. Deng	Singapore Management U, Singapore

Organizing Committee

Xiaotie Deng	City University of Hong Kong, China
L. F. Kwok	City University of Hong Kong, China
C. H. Lee	City University of Hong Kong, China
Duncan Wong	City University of Hong Kong, China
Giovanna Yau	

Program Committee

Joonsang Baek	I2R, Singapore
Feng Bao	I2R, Singapore
Kefei Chen	SJTU, China
Liqun Chen	HP Bristol Labs, UK
Mathieu Ciet	Gemplus, France
Ed Dawson	QUT, Australia (Co-chair)
Cunsheng Ding	HKUST, China
Dengguo Feng	Chinese Academy of Sciences, China
Dieter Gollmann	TU Hamburg, Germany

Javier Herranz	CWI, The Netherlands
Lucas Hui	Hong Kong University, China
Min-Shiang Hwang	National Chung Hsing University, Taiwan
Tetsu Iwata	Nagoya University, Japan
Kazukuni Kobara	AIST, Japan
Kaoru Kurosawa	Ibaraki University, Japan
Kwangjo Kim	ICU, Korea
L. F. Kwok	City University of Hong Kong, China
Xuejia Lai	SJTU, China
Dong Hoon Lee	Korea University, Korea
Benoit Libert	UCL, Belgium
Joseph K. Liu	University of Bristol, UK
Javier Lopez	University of Malaga, Spain
Stefan Lucks	University of Mannheim, Germany
Richard Lui	City University of Hong Kong, China
Wenbo Mao	HP Lab, China
SangJae Moon	Kyungpook National University, Korea
Yi Mu	University of Wollongong, Australia
David Naccache	ENS and Paris II, France
Raphael C.-W. Phan	Swinburne University of Technology, Malaysia
Josef Pieprzyk	Macquarie University, Australia
Jean-Jacques Quisquater	UCL, Belgium
C. Pandu Rangan	Indian Institute of Technology, India
Reihaneh Safavi-Naini	University of Wollongong, Australia
Hovav Shacham	Weizmann Institute of Science, Israel
Ron Steinfeld	Macquarie University, Australia
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Future University - Hakodate, Japan
Xiaojian Tian	City University of Hong Kong, China
Guilin Wang	I2R, Singapore
Huaxiong Wang	Macquarie University, Australia
Stephen Wolthusen	University of London, UK
Duncan Wong	City University of Hong Kong, China (Co-chair)
Joe Yau	Hong Kong Baptist University, China
Sung-Ming Yen	National Central University, Taiwan
Yiqun Lisa Yin	Independent Consultant, USA
Siu-Ming Yiu	Hong Kong University, China
Fangguo Zhang	Sun Yat-sen University, China
Zhenfeng Zhang	Chinese Academy of Sciences, China
Yunlei Zhao	Fudan University, China
Yuliang Zheng	UNC Charlotte, USA
Jianying Zhou	I2R, Singapore
Huafei Zhu	I2R, Singapore

External Reviewers

Manfred Aigner
Man Ho Au
Philippe Bulens
Xuefei Cao
Julien Cathalo
Zhenchuan Chai
Chris Charnes
Chien-Ning Chen
Haibo Chen
Jing Chen
Lily Chen
Xiaofeng Chen
Yongxi Cheng
Benoit Chevallier-Mames
Yvonne Cliff
Scott Contini
Kim-Kwang Raymond Choo
Andrew Clark
Hanane Fathi
Benoit Feix
Evan Fleischmann
David Galindo
Zheng Gong
Qianhong Huang
Qiong Huang
Dennis Hofheinz
Xuan Hong
Jeffrey Horton
Chao-Chih Hsu
Zoe Jiang
Haimin Jin
Marc Joye
Tanmoy Kanti Das
Stefan Katzenbeisser
Eike Kiltz
Jongsung Kim
Hirotsugu Kinoshita
Divyan M. Konidala
Ulrich Kühn
Byoungcheon Lee
HoonJae Lee
Sang Gon Lee
Lan Li
Vo Duc Liem
Hsi-Chung Lin
Jenny Liu
Yu Long
Miao Ma
Adrian McCullagh
Pablo Najera
Dang Ngyuen Duc
Lan Nguyen
Juan Gonzalez Nieto
Peng Ning
Miyako Ohkubo
Yasuhiro Ohtaki
Pascal Paillier
Kun Peng
Ying Qiu
Rodrigo Roman
Chun Ruan
Eun-Kyung Ryu
Hendra Saptura
Werner Schindler
Francesc Sebé
Weijun Shen
Nicholas Sheppard
Mi Na Shim
SeongHan Shin
Masaaki Shirase
Nigel Smart
Dirk Stegemann
Purui Su
Willy Susilo
Tsuyoshi Takagi
Keisuke Tanaka
Hitoshi Tanuma
Emin Islam Tatli
Feng Tu
Jheng-Hong Tu
Damien Vergnaud
Lionel Victor
Jose L. Vivas
Eric Wang
Shuhong Wang
Zhenghong Wang
Brent Waters
Baodian Wei
Mi Wen
Jian Weng
Chi-Dian Wu
Qianhong Wu
Guomin Yang
Kee-Young Yoo
Jin Yuan
Erik Zenner
Rui Zhang
Chang'an Zhao

Sponsoring Institutions

City University of Hong Kong, China

Table of Contents

Invited Talks

Cryptanalysis

Signatures

Network Security and Security Management

Privacy and Applications

Cryptographic Algorithms and Implementations

Authentication and Key Management

Cryptosystems

Application Security – Myth Or Reality?

William J. Caelli

Information Security Institute
Queensland University of Technology,
GPO Box 2434, Brisbane. Qld. 4001. Australia
w.caelli@qut.edu.au
Senior Consultant–Information Assurance and Director
International Information Security Consultants (IISEC) Pty Ltd
21 Castle Hill Drive South, Gaven. Qld. 4211. Australia
w.caelli@iisec.com.au

Abstract. The Security services within applications have received recent attention. It has been suggested that this may be the only way to increase overall information system assurance in an era where ICT governance and compliance have taken on new force and the use of commodity level ICT products for critical information systems continues. While it has been argued that an application can be no more secure than its underlying computer sub-systems, security at the application layer was always envisaged as playing a major role, e.g. in the "Open Systems Interconnection (OSI)" security model. At a time when "end-user" programming is being advocated, the needs and parameters of security education and training are rapidly changing, and increased threats from global Internet connection are rapidly rising, there is a need to reconsider security schemes at the application level. This paper examines current trends in application design, development, deployment and management and evaluates these against known system vulnerabilities and threats.

Keywords: OSI security, access control, mandatory access control, security education, operating system security, application security, web services security.

1 Introduction – Security "Ignorant" Versus Security "Aware" Applications

Even by 1992 the Organisation for Economic Cooperation and Development (OECD) had set up a set of recommendations that set out guidelines for the security of information systems [1]. These guidelines were accompanied by a call for their implementation in the following statement:

"…. Governments are urged to establish legal, administrative and other measures, practices and institutions for the security of information systems."

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 1 – 10, 2007.

This theme was taken up in 1995 by the then Australian Governor-General who set the scene for information security and its future in the following statement reported by "The Australian" newspaper [2]:

> *"… Hayden also said it was 'incumbent on us as individual Australians' to seriously consider issues such as privacy, information security and copyright, equity and access and not just leave such concerns up to governments."*

By this time the British Standards Association had published its BS7799 standard, labelled as a *"Code of Practice for Information Security Management"* which was heralded as a document to *"provide a common basis for companies to develop, implement and measure effective security management practice"* and to *"provide confidence in intercompany trading"*. Its origin had been with the United Kingdom's Department of Trade and Industry (DTI) and a group of companies and other organisations. It set out ten categories of security controls, all of which are vital in the consideration of computer application security. These categories were, and still are, based upon the parameters shown in Table 1.

Table 1. OECD Parameters

Parameter	OECD-Category of Security Control
1	Security Policy
2	Security Organisation
3	Assets Classification and Control
4	Personnel Security
5	Physical and Environmental Security
6	Computer and Network Management
7	Systems Access Control
8	System Development and Maintenance
9	Business Contingency Planning
10	Compliance

Considering these admonitions in the light of global information networking it is vital to assess the simple fact that users "see" applications and seldom any underlying computer or data network structure. These can be quite specific, e.g. an inventory control package for an electrical products distributor, or generic by nature, e.g. web browser, office productivity suite, etc.

It is an accepted principle of computer science and engineering that a computer application can be **no more** secure than the libraries and middleware it incorporates that can themselves be **no more** secure than the operating system and sub-systems that support them which in turn can be **no more** secure than the underlying hardware and firmware of the computer or network system. While this is an obvious truth, applications themselves can be further subdivided into two broad classes, i.e. security "aware" versus security "ignorant" applications. In simple terms, a security "aware" program incorporates appropriate security mechanisms and services relative to the needs of the application and appropriate to the security environment of the information system in which it operates. By contrast, a security "ignorant" application simply depends upon other system wide security services and mechanisms to provide

necessary protection, e.g. operating system relevant access control parameters, network boundary/perimeter controls, and the like.

This broad dichotomy then further leads to a set of two differing "views" of such applications and their operation in any information system. These can be broadly categorised, as per Figure 1, as being the;

a. *"system-up"* view, versus the
b. *"application down"* view.

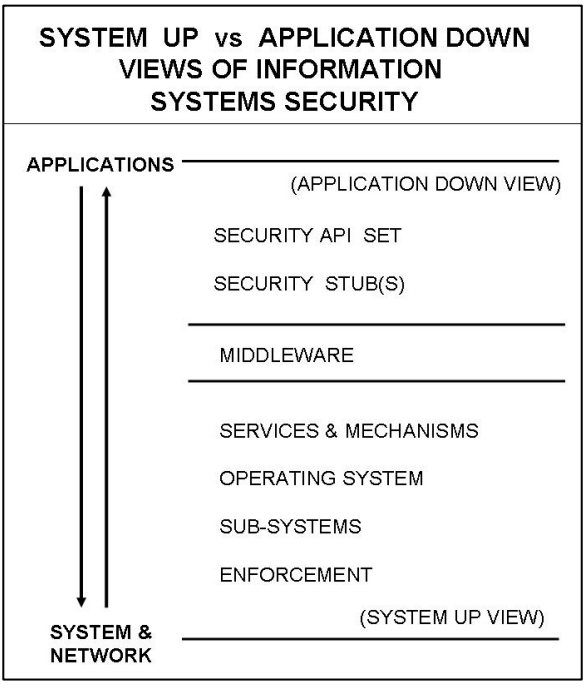


Fig. 1. Differing Views

The system up paradigm assumes that security responsibility for an information system, in general, lies outside the scope of the applications developer and belongs to the overall information system manager who controls those applica-tions according to a documented enterprise risk assessment and management policy. Indeed, it can be argued that this view if the one prevalent in such ICT processes as "business process management (BPM)" and allied schemes used for the creation of any overall information system. The alternative, but often co-existent, scheme of "application down" views of security can be clearly identified in particular application sectors, e.g. the banking and finance, healthcare, government services and allied areas. The main question for the ICT professional is one of how to balance these differing views and to determine just "where they meet".

For example, national and international standards exist for application security parameters in the banking/finance and healthcare sectors and these vary markedly in the degree of detail involved. From definition of actual security processes to data formats and storage parameters these specific requirements must form part of any enterprise analysis activity undertaken and must be an integral part of an overall application system. These security parameters, being application specific, have the property that they do not readily lend themselves to incorporation into "lower level" services in a system, e.g. access control schemes provided by an operating system. For the immediate future, application security specifics seem likely to remain for certain industry sectors. However, there is growing interest in the concept of a "regulatory layer", similar to the Open Systems Interconnection's (OSI) "presentation

layer" (Layer 6 of the OSI model) as shown in Table 2 at the end of this paper. In this model, security enforcing aspects of an application, particularly where security requirements are defined by legal and/or regulatory bodies, are isolated from the main application "logic" and placed in this "regulatory layer". Essentially what is demanded is reliable enforcement of international, national, state/province, local and enterprise security laws, regulations, guidelines and policies. Indeed, information security or information "assurance" is now an integral part of any enterprise information systems model in the public or private sectors alike. The important point is one of matching user expectations for simplified and understood access with these security/assurance parameters at the application level as illustrated in a newspaper cartoon from the early 1980s [3], given in Figure 2.

Fig. 2. ATM Security - 1983

2 The Open Systems Interconnection (OSI) Model as a Framework

The OSI model, with its 7-layer structure, also defined an architecture for the protection of interconnected systems; in principle, those applications and related systems that needed to communicate. At the management level, the "OSI Management" architecture, clearly identified five major sets of principles that governed:

- naming and configuration,
- security,
- error and fault handling,
- performance, and
- accounting.

OSI then clearly stated its security philosophy as follows:
"At various times, security controls must be established in order to protect information exchanged between application processes. Such controls should make the cost of obtaining or modifying data greater than the potential value of so doing, or make the time required to obtain the data so great that the value of the data is lost."
This led to the definition of three security management relevant parameters as follows:

- Mandatory security policies imposed by owners and administrators of communicating entities,

- Discretionary security policies that govern the groups of entities with which a particular entity wishes to communicate along with the protection services to be employed in the associated communication channels, and
- Control and distribution of required security information to communicating entities related to provision of required security services, reporting on security services provided and reporting on any security relevant events that may occur.

These parameters then contributed to the development of the concept of the "Security Management Information Base (SMIB)" as part of an overall OSI compliant "Management Information Base (MIB)" that itself may be centralised or distributed.

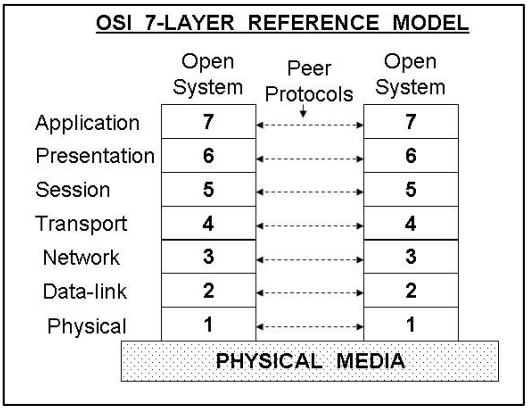


Fig. 3. OSI Model

From an application security viewpoint, the OSI model provides the best overview of the security parameters involved in creating trusted information systems and services. The seven layer model [4] is widely accepted as base reference model for discussion as shown in Figure 3. Indeed, the IS 7498-2 document, a subsidiary document to the IS 7498 standard, set out the overall security architecture in two distinct ways;

a. the definition of fourteen required security "services" and eight associated security "mechanisms", and

b. placement of these services in the layered structure of the OSI model.

The lists of services and related mechanisms are as follows while their placement is given in Table 2. The OSI security services are seen as:

- Peer entity authentication
- Data origin authentication
- Access control services
- Connection confidentiality
- Connectionless confidentiality
- Selective field confidentiality
- Traffic flow confidentiality
- Connection integrity with recovery
- Connection integrity without recovery
- Selective field connection integrity
- Connectionless integrity
- Selective field connectionless integrity
- Non-repudiation – origin, and
- Non-repudiation – Delivery

The associated security mechanisms are seen as being:

- Encipherment,
- Digital signatures,
- Access control,
- Data integrity,
- Authentication exchange,
- Traffic padding,
- Routing control, and
- Notarisation.

Table 2. Security by OSI Layers

OSI SECURITY

Security Services / OSI Layers

SERVICE	LAYER 1	2	3	4	5	6	7
Peer Entity Authentication	-	-	Y	Y	-	-	Y
Data Origin Authentication	-	-	Y	Y	-	-	Y
Access Control Service	-	-	Y	Y	-	-	Y
Connection Confidentiality	Y	Y	Y	Y	-	Y	Y
Connectionless Confident.	-	Y	Y	Y	-	Y	Y
Selective Field Confident.	-	-	-	-	-	Y	Y
Traffic Flow Confident.	Y	-	Y	-	-	-	Y
Connection Integrity + Recovery	-	-	-	Y	-	-	Y
Connection Integrity without Recovery	-	-	Y	Y	-	-	Y
Selective Field Connection Integrity	-	-	-	-	-	-	Y
Connectionless Integrity	-	-	Y	Y	-	-	Y
Selective Field Connectionless Integrity	-	-	-	-	-	-	-
Non-repudiation, Origin	-	-	-	-	-	-	Y
Non-repudiation, Delivery	-	-	-	-	-	-	Y

As can be clearly observed all but one service is seen in the model as being possible at the application layer, layer 7. This indeed places emphasis on the need to carefully consider just how application level security interacts with any other security service and mechanism that occurs below that layer. With the emerging concept of some form of "regulatory layer", an adjunct to the OSI "presentation layer" or even an added functionality for that layer, the role of these services needs to be clearly considered in any overall information system development.

The time is right to reconsider the extensive work done in this area during the 1980s and early 1990s and to incorporate the results of that extensive research and standards setting base into current Internet based connected application systems.

3 Seven Challenges

With the above background in mind, seven distinct challenges to the future of application development can be set out.

1. Education and Training for the ICT Professional

A major problem may be that the ICT professional, in the form of the "application architect", "enterprise systems analyst", "business process management (BPM)

analyst, etc. may have received little to no education or training in the areas of information security relevant to the profession and to the design of safe and secure information systems. The problem emerges of not only provision of that training but also of the necessity of "training the trainers", i.e. the base of academics and professionals involved in the associated educational institutions and enterprises capable of providing that education service. There is a question, therefore, of the potential for splits in the profession, e.g. between normal business process management analysis and analysis against associated governance and compliance requirements. This could mean that a stratification of professional service could be a result of lack of education professionals in the information security area.

2. Development for Least Privilege and Cooperating Systems
In an age of "outsourcing", "co-opetition", "integrated supply chains across enterprises", and so on, application systems need to be developed with the principle of "least privilege" in mind. It is simply too easy for an application to be developed around the concept that, in operation, that application has full access to all resources of the computer platform on which it operates. This means that a new level of application system development education needs to be embraced by universities, colleges, and allied enterprises.

3. Moving to Hardened System Environments
In essence there is a major question facing the development of application systems particularly for critical infrastructure systems, e.g. banking and finance, healthcare, government services and the like. There appears to be a realisation, particularly at the server level, that old and obsolete "discretionary access control (DAC)" structures are no longer sufficient for the protection of these systems. As such a move towards computer operating systems and allied middleware/support sub-systems that provide "Role Based Access Control (RBAC)" and even "Mandatory Access Control (MAC)" schemes appears to be gaining momentum. Application systems need to take advantage of this movement. However, as in the first challenge above, this simply means that ICT professionals need to have the requisite education and training to take advantage of and support these "hardened"systems. This, in particular, includes understanding the concepts and processes behind the internationally accepted "Common Criteria" for the evaluation of the security of systems, i.e. International Standard IS 15408.

4. Trusted DBMS, Middleware and Sub-systems
In support of increasing trust parameters in the operating system, there is growth in the development and support of such "hardened" services at the levels above the operating system. This is particular seen in such sub-systems as data base management systems (DBMS) and network access "stacks". Once again the application developer needs to be aware of and have the necessary tools to create applications that take advantage of these systems.

5. Web Services Security
There is undoubtedly justified scepticism about the overall, end-to-end, computer application to computer application security of information systems based around the deployment of web services structures. Mimosa recently stated:

"The sophisticated business-to-business interactions occurring at a service level with service-oriented architectures pose a major challenge to security. You don't go SOA to be more secure; you go SOA for the sake of efficiency and integration, standardization and code reuse. The returns are tantalizing, but like any other development scenario where a rush-to-market supersedes security, the results could be disastrous, experts say." [5]

The same theme was also put forward by Hulme as follows:

"After more than a decade of organizations focusing on locking down network perimeters, endpoint devices and email, Web applications have surfaced as the new attack flashpoint. Last year was a bad year for Web application security--whether it was overseas hackers reportedly accessing credit card information from thousands of transactions on the state of Rhode Island's Web site, or universities inadvertently spilling sensitive student information, including Social Security numbers, onto the Internet. Statistics back this up. Symantec said in its most recent Internet Security Threat Report that ***Web vulnerabilities constituted 69 percent of 2,249 new vulnerabilities the company documented for the first half of 2006, with 78 percent of "easily exploitable" vulnerabilities residing within Web applications.*** *Mitre Corp.'s September tally of publicly disclosed vulnerabilities mirror those findings, with cross-site scripting vulnerabilities surpassing buffer overflows as the most reported vulnerability. Four of the top five vulnerabilities were within Web applications, development platforms, or databases often directly exposed to the Internet." [6]* (Emphasis is by this author.)

Besides the fact that in early 2007 the overall security architecture for web services is incomplete, confused and complex, the structures set out depend totally upon underlying trusted systems functionality and enforcement in connected computer systems, particularly for cryptographic services and related secure key management.

This fact was clearly stated by Peterson and Lipson [7] of the USA's Software Engineering Institute of Carnegie Mellon University and Cigital, Inc. in the following way:

"The WS-Security standard does not address other issues related to security infrastructure such as key management, and it does not address policy, which must be set up separately."

This is a critical statement given that 2006 saw the dominance of attacks on system move to compromise of underlying computer operating systems and middleware through such vehicles as "root kits", etc.

At the same time, the temptation to develop application specific "add-on" packages for browser systems exists. In the open source arena, this approach is actually encouraged, e.g. through support for such add-on development for the FireFox browser system [8]. This approach is clearly evidenced by the following statement from the "Mozilla" web site referenced:

"Extensions are small add-ons that add new functionality to Mozilla applications such as FireFox and Thunderbird. They can add anything from a toolbar button to a completely new feature. They allow the application to be customized to fit the personal needs of each user if they need additional features, while keeping the applications small to download."

6. Connectivity
An application today "talks" to another application over Internet protocol based data communications networks. While security related protocols and structures exist at the network level and are in wide use, the problem exists of determination of the status of the cooperating computer systems used to host the applications. In application development, there is a major need for mechanisms and methodologies for an application to become "self-defending" through provision of its own defence mechanisms and through structures that allow it to determine the security status of the environment in which it is executing. This is still a major research area and no clear direction is obvious at resent.

7. Towards "Self-Defending" Objects
With the rapid rise of "mobility" in global information systems, protection of that system by use of "perimeter" protection mechanisms is no longer feasible or even effective. Access to large enterprise information systems can now be effected from everything form a mobile/cell phone or PDA to a high powered personal computer/workstation in an unprotected home environment to a sophisticated workstation within the enterprise itself. This means that a new paradigm is needed in thinking about the way in which applications will exist in the future. They may even propagate themselves around enterprise networks to provide service where and when needed. In this sense the concept of a "self-defending object", developed using software "components" that themselves exhibit that property, may become a vital concept for the future.

4 Conclusion

Once again the basic truth needs to be re-iterated. No application can any more secure than the sub-systems it depends upon. The future in research needs, in some way, to elucidate this truth by enabling an application to actually determine the security status of all the systems upon which it depends. Surprisingly this is no more than the basic theme set out in the underlying concept of "mandatory access control" with all elements of a system reliably tagged with relevant security data. The question proposed by this paper has been one of a value judgement. Can an application, by itself, be secure? The answer is undoubtedly "no" particularly as today's applications make use of numerous pre-existing components that the application developer has little to no knowledge of or responsibility for. However, van applications become security aware? The answer here seems to be a "yes"; particularly as such applications become "self-defining" in the future.

However, for the ICT professional, limited by pressures of creating applications "on time and on budget", the problem of dedication to application security could best be summarised in the words of Herodotus, 484 to 424 B.C., as follows:

"Of all man's miseries, the bitterest is this: to know so much and have control over nothing." [9]

References

1. OECD – Organisation for Economic Cooperation and Development, Directorate for Science, technology and Industry, Committee for Information, Computer and Communications Policy, Ad Hoc Group of Experts on Guidelines for the Security of Information Systems, "Recommendation of the Council Concerning Guidelines for the Security of Information Systems and Explanatory Memorandum to Accompany the Guidelines", DSTI/ICCP/AH(90)21/REV6, 18 September 1992.
2. The Australian, 31 January 1995.
3. The Weekend Australian, 13-14 February 1983.
4. "Information Processing Systems – Open Systems Interconnection – Basic Reference Model", IS 7498 / AS 2777-1985, Standards Australia 1985.
5. Mimoso, M. S. "Pitfalls aplenty going SOA", Information Security Magazine, 5 Feb 2007. http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci1241899,00.html cited at 11 Feb 2007.
6. Hulme, G., "Web apps remain a trouble spot", in SearchSecurity.com, 5 Feb 2007.
7. http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci1241953,00.html at 11 Feb 2007.
8. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/assembly/639.html?branch=1&language=1 and https://buildsecurityin.us-cert.gov/daisy/bsi/home.html
9. http://developer.mozilla.org/en/docs/Extensions
10. http://en.wikiquote.org/wiki/Herodotus

Tools and Technology for Computer Forensics: Research and Development in Hong Kong (Invited Paper)

Lucas C.K. Hui, K.P. Chow, and S.M. Yiu

Department of Computer Science
The University of Hong Kong
Hong Kong
{hui,chow,smyiu}@cs.hku.hk

Abstract. With the increased use of Internet and information technology all over the world, there is an increased amount of criminal activities that involve computing and digital data. These digital crimes (e-crimes) impose new challenges on prevention, detection, investigation, and prosecution of the corresponding offences. Computer forensics (also known as cyberforensics) is an emerging research area that applies computer investigation and analysis techniques to help detection of these crimes and gathering of digital evidence suitable for presentation in courts. This new area combines the knowledge of information technology, forensics science, and law and gives rise to a number of interesting and challenging problems related to computer security and cryptography that are yet to be solved. In this paper, we present and discuss some of these problems together with two successful cases of computer forensics technology developed in Hong Kong that enable the law enforcement departments to detect and investigate digital crimes more efficiently and effectively. We believe that computer forensics research is an important area in applying security and computer knowledge to build a better society.

Keywords: Computer forensics, digital crimes, forensics technology.

1 Introduction

The use of Internet and information technology has been increasing tremendously all over the world. In Hong Kong, according to the surveys conducted by Census and Statistics Department of the Government, the percentage of households with personal computers at home that are connected to Internet has increased by more than 75% from 2000 to 2005 (see Table 1) while for the business sector, the percentage of business receipts through electronic means has increased by almost four folds (see Table 2). As one may expect, the amount of criminal activities that involve computing and digital data (digital crimes or e-crimes) has also increased. From the statistics provided by the Hong Kong Police [5], the number of digital crimes in Hong Kong has increased more than double from 2001 to 2004.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 11–19, 2007.

Table 1. Penetration of Information Technology in the Household Sector in HK [2]

	Year 2000	Year 2005
Households with personal computers at home	49.7%	70.1%
Households with personal computers at home connected to Internet	36.4%	64.6%

Table 2. Penetration of Information Technology in the Business Sector in HK [2]

	Year 2000	Year 2005
Establishments with personal computers	51.5%	60.5%
Establishments with Internet connection	37.3%	54.7%
Establishments with Webpage or Website	7.3%	15.5%
Business receipts through electronic means	0.17%	0.64%

These digital crimes (e-crimes) impose new challenges on prevention, detection, investigation, and prosecution of the corresponding offences. Computer forensics (also known as cyberforensics) is an emerging research area that applies computer investigation and analysis techniques to help detection of these crimes and gathering of digital evidence suitable for presentation in courts. While forensic techniques for analyzing paper documents are very well established, very few of these techniques can be applied to digital data and they were not designed for collecting evidence from computers and networks. This new area combines the knowledge of information technology, forensics science, and law and gives rise to many interesting and challenging problems related to computer security and cryptography that are yet to be solved.

Among other issues in collecting evidence from computers, one fundamental difference between paper documents and digital data is that electronic data can be easily copied and modified. A suspect may easily argue that the evidence found in his/her computer was implanted or modified by the law enforcement agency after the computer has been seized by the agency. It is very important to verify the *file system integrity* of the suspect's computer after it has been seized by the law enforement agency.

Another problem is that there are many different file formats, operating systems and file system structures. Electronic documents can be generated by various kinds of application programs such as word processors, spreadsheet software, database software, graphic editors, electronic mail systems. The documents can be stored as user files in user directories, or as fake system files in the system

directories, or hidden files. Sometimes, evidence can also be found in the *deleted* files. When a file is deleted, the operation system usually only removes the references to the file in the file allocation table (FAT). The actual content of the file is still physically stored on the disk until that area has been overwritten by another file. It is a time consuming task to inspect every possible storage area of the whole computer for potentially useful evidence. And it is also not possible to check every file using all available application programs manually. In this paper, we will briefly describe a cyber crime evidence collection tool [4], called *Digital Evidence Search Kit (DESK)* ich tries to handle the above problems. DESK is the product developed by our research team and the Hong Kong Police Force and several other law enforcement agencies of the Hong Kong Special Adiminstrative Region.

Besides the problem of evidence collection, e-crime detection is also very important. Intrusion detection (e.g. detection of distributed denial of service attack [9,13]) is one of the well-known examples. In this paper, we focus on another example - detection of copyright infringement through peer-to-peer (P2P) file sharing. According to a survey conducted by the Hong Kong Government in 2005 [7], the public awareness of IP (Intellectual Property) rights has significantly improved. Out of about 1200 respondents, only 15% admitted that they would often (0.7%) or sometimes (14.3%) buy pirated or counterfeit goods. This is already a remarkable improvement from the 24.7% in 1999. However, the percentage of respondents who admitted that they would illegally download and upload files to Internet for the purpose of sharing with others has increased from 3.5% in 2004 to 6.8% in 2005. This may indicate that the copyright infringement problem becomes more serious (at least in Hong Kong) as the peer-to-peer file-sharing protocols become more popular and mature.

In fact, this is not only a problem in Hong Kong. According to a third-party research, potential losses to the recording industry from P2P file-sharing was estimated at US$2.1 billion in 2004 [6]. Among the few successful P2P protocols in existence, BitTorrent (BT) has evolved into one of the most popular networks [8] and has managed to attract millions of users since inception. By the end of 2004, BitTorrent was accounting for as much as 50% of all P2P-related traffic [11]. Without doubt, P2P technology offers a wonderful platform for individuals and organizations to share their digital materials worldwide extremely easily. Unfortunately, its illegitimate use on unauthorised sharing of copyrighted files is increasingly rampant and is reaching an alarming height.

With the existence of the overwhelming private BitTorrent networks, it is difficult to gauge the actual numbers of BT users. What we are certain, however, is the tremendous loss to the media industries. Over the years, law enforcement agencies have set out operations to fight against these illegal activities. With much of their effort, the world's first conviction of piracy of BitTorrent user was sentenced in the fall of 2005. However, the outcome seems not to be an effective deterrent to average BT users. Although many individuals realize that what they are doing is a kind of online piracy and is illegal under recently enacted legislation, they still pursue the file sharing as before. One critical issue behind

this is the limited manpower and resources available to law enforcement agencies. BT users may feel that it is almost impossible to crack down every single member of the enormous BT user base. To tackle this problem, it is desirable to have an automated system for monitoring these increasingly rampant BT activities. In this paper, we will briefly describe a rule-based BT monitoring system (BTM [3]) which takes the first step towards solving this problem.

The rest of the paper is organized as follows. The DESK system will be described in Section 2. Section 3 will briefly talk about the BTM system. Section 4 concludes the paper by presenting a few other related problems in computer forensics.

2 The Digital Evidence Search Kit

DESK (The Digital Evidence Search Kit) is a general purpose computer forensics system focusing on integrity control of the digital data. There are two design objectives of this tool. One of the objectives is to ensure the validity and reliability of the digital evidence. Once it has been proved that the tool has been used properly and in compliance with the Evidence Ordinance [10], the digital evidence found in the suspect's computer can be presented and used in courts for prosecution. Another objective is to provide an efficient and automatic search function to search for digital contents that can be used as evidence for the e-crime. DESK is also specially designed to be used in the bilingual environment of Hong Kong, so is capable of searching word patterns in both English and Chinese (traditional and simplified chinese characters).

2.1 The Framework of DESK

The DESK system consists of a DESK machine which is typically a notebook computer with a serial port and a floppy diskette used to start up the suspect's machine (subject machine). The DESK machine will be connected to the subject machine using a serial (RS-232) cable. There are two software components of DESK: the DESK client that is installed on the DESK machine; and the DESK server that is contained on the floppy diskette to be executed by the subject machine. The DESK client is mainly used to provide a user interface for issuing commands to inspect the subject machine.

The DESK server component, installed on the floppy diskette, has additional functionalities which include the followings.

1. To start up the subject machine: Usually the file (e.g. system files) in the subject machine will be modified if it is booted up by its own operating system.
2. To *lock* the subject machine: This is to protect the subject machine from any accidental corruption by the interrupts of the machine. This step is very important as it can ensure that the contents found on the subject machine

cannot be modified, thus ensures the integrity integrity of the subject machine while various forensic operations are being performed.
3. To provide a simple user interface for simple search operations: The user interface is much less sophisticated than that of the DESK client running on the notebook due to the storage limitations of floppy diskettes.

There are two main operations of DESK: keyword search and file system integrity checker.

Keyword Search: A pre-defined text pattern file which contains important keywords that can be specific to a particular case, in Chinese and/or English, to be searched for on the subject machine, is used for three different types of search, namely physical search, logical search and deleted file search. Physical search performs a search of the patterns in each physical sector of the subject machine's storage system. E-crime evidence stored purposely in unused sectors can be discovered. Logical seach, on the other hand, makes use of the information about the file system, so patterns stored across different sectors can be located. Deleted file search will try to locate the deleted file provided it is not yet overwritten by another file and perform the pattern search on these files.

File System Integrity Checker: There are two functions in this checker. Firstly, it is to ensure the integrity of the file system of the subject machine. We compute a hash value of the whole file system (e.g. a hard disk) of the subject machine. By recording this hard disk hash value properly, the law enforcement agency can easily prove that the content of the hard disk has not been modified after the machine has been captured by the agency. Also, in order to reduce the possibility of causing accidental damage to the hard disk, usually exact copies of disks (also called clone images) are made for the subsequent analysis. The hash values of the clone images and the original hard disk can be compared to show that the clone images are exactly the same as the original hard disk.

Secondly, the suspect may store some crime evidence in standard files of common software applications (e.g. freecell.exe). A hash value database that contains fingerprints (hash values) of all files in a standard software distribution are used to compare with the hash values of the corresponding files in the subject machine. More details of the DESK system can be found in [4].

2.2 Other Issues

There are other issues related to this research. For examples, it is very likely that there may be missing/bad sectors in the hard disk which may corrupte the data files in the system. How this can be handled to make sure that the recovered portion of the files can still be presented in courts needs more investigation. Also, the integrity checker relies very much on the hash functions. With the recent cracking of some well-known hash functions such as SHA-1 and MD5, may be a more detailed study needs to be done to make sure that the validity of the digital evidence is not questionable.

3 A Rule-Based BT Monitoring System

In this section, we briefly discuss the design of a rule-based BitTorrent monitoring system (BTM). For details, please refer to [3].

3.1 Basics of BitTorrent (BT)

A BitTorrent network is basically made up of four types of entities.

- Tracker: A server that coordinates the distribution of files. It acts as an information exchange center from which peers obtain necessary information about other peers to which they can connect.
- Torrent file: A small file which contains metadata about the files, including the address of the tracker, to be shared.
- Peer: Anyone that participates a download.
- Seeder: A peer who has a complete copy of the file and offers it for download. All peers (including the seeders) sharing the same torrent, are considered as a unit, called a *swarm*.

Note that the idea of BT is to redistribute the cost of upload to downloaders. When the peers are downloading the same file at the same time, they upload part of the files to one another. To start a download, a torrent file is generated, registered with a tracker and made available somewhere in a website. The owner of the initial copy of the file is referred as the initial seeder. Initially, peers will contact the initial seeder to request the file, as more and more peers join in, some peers will share their pieces with newly joined peers to offload the initial seeder.

3.2 The Framework of BTM

To track down the activities of a swarm, the torrent file is the key. BTM consists of two main components, the torrent searcher and the torrent analyzer. To locate torrent files, the torrent searcher searches target websites (or public forums) specified by user-inputted URLs. The torrent files will then be passed to the torrent analyzer for detailed analysis. There are several issues to be resolved by the torrent searcher. For examples, the searcher needs to explore the websites level by level following the hyperlinks to reach the torrent files. Automatic keyword searching needs to be performed by the searcher in order to explore potential illegal downloading activities in public forums. To conclude, this torrent searcher can be configurated to work on updated topics (e.g. newly released movies) and on scheduled websites/forums. It makes the monitoring possible for 24 hours.

After obtaining the torrent files from the torrent searcher, the torrent analyzer can locate and connect to the tracker(s) and retrieve the lists of peers currently participating in the torrent. It can further connect to the peers and gather useful information about the download activity and analyze the information to, say identify potential seeders and to determine if any necessary action needs to be triggered. The core engine inside the torrent analyzer is a rule-based system. Some preliminary tests have been conducted in some real scenarios. The

results are promising, however, more detailed analysis and experiments need to be performed to confirm its effectiveness.

3.3 Other Issues

This system represents the first step towards an automated monitoring system for the detection of copyright infringement activities through peer-to-peer file sharing. There are many other concerns that need an in-depth study. For examples, the anonymity level of BT is continuously being improved, how these anonymity features of the upgraded version affect the effectiveness of BTM is certainly one of the main concerns. On the other hand, the scalability of the tool is also a major issue needs to be resolved since the BT protocol seems to be very scalable and the number of peers can be huge.

4 Conclusion and Other Problems

In the previous two sections, we had described two examples in computer forensics research and development. To conclude this paper, we describe a few other related problems in computer forensics. Actually, we are working on some of these problems and preliminary research results may appear soon.

We believe that computer forensics research is an important area in applying security and computer knowledge to build a better society.

4.1 Live Systems Forensics

Most of existing computer forensics techniques concentrate on efficient search of digital evidence inside an inactive computer. The emphasis is on whether a particular piece of evidence exists in the machine or not. Recently research in computer forensics attempts to collect digital evidence from a live running system (e.g.[1]). This type of evidence may contain information that is transient, e.g. network connection. On the other hand, the ultimate goal of computer forensics is to reconstruct the crime scene in the digital world. Therefore one research direction is to concentrates on how to make use of the digital evidence collected from a live running system, filter out irrelevant information, and reconstruct the crime scene. This will involve not only carry out digital evidence search based on the syntactic elements, but also interpreting the evidence in a semantically correct way.

4.2 Cryptographic Scheme Design to Enhance Computer Evidence Validity

Digital data in a computer system needs confidentiality, integrity, and authentication control. With the viewpoint that those digital data may be extracted as

evidence by a computer forensics exercise, it will be better to design advanced cryptographic schemes which, during the time the digital data is generated, will provide cryptographic objects (such as hash values and digital signatures) at the same time. One example requiring this functionality is multi-media data. When a video file is used as a piece of computer evidence, we need to prove that a video file is really produced by a certain camera, it is really being created on a particular date, and is not tampered afterward. In addition, if part of the video file is corrupted, we still want the uncorrupted part to be valid evidence. This is an important research direction since our society is generating more and more different types of digital data, including text, documents, video, file systems, and others.

4.3 Authentication Schemes Providing Better Evidence

While authentication and the related topic of access control are being studied for a long time, there are still a lot of rooms for improvement regarding the provision of evidence. For example, to provide evidence about a login process using password, we need to assume the validity of the log file [12]. As a lot of criminal activities involve the step of impersonation, the computer evidence about authentication is particularly important. This situation is also being complicated by the diversified techniques of authentication, including password, digital signature, hardware cryptographic tokens, biometrics, one-time password, time-synchronous tokens, and third-party credentials. Therefore, the study of authentication with emphasis on the evidence provided is greatly desired.

4.4 Digital Objects with Dual Meanings

With the combination of cryptography, steganography, and the complicated data formats for digital documents, it is possible to create a digital object which can be interpreted in two or more different meanings. For example, a computer file can show different contents when it is being opened by two different software viewers. With one viewer the content can be a normal text story, while with another viewer it can be a child pornographic picture. Following the same idea, a more elaborate example is that a file can contain two encrypted portions. The file can be decrypted with two different decryption keys to show two different contents. What is the motivation of a person if he is storing such a file with dual meaning? Although finding the motivation of a person is not a computer security technical problem, there are technical problems involved: If a file with dual meanings is stored in a suspect's computer, will the computer forensics process be able to identify the two different meanings? What are the different technologies available for providing files with two or multiple meanings? Besides computer files, are there other kind of digital objects that can also have dual meanings? All these are interesting research topics with great impact.

References

1. Frank Adelstein. Live forensics: Diagnosing your system without killing it first. *Communications of the ACM*, 49(2):63–66, 2006.
2. Census and The Government of Hong Kong Special Administrative Region Statistics Department. Hong kong in figures, 2006 edition, 2006.
3. K.P. Chow, K.Y. Cheng, L.Y. Man, K.Y. Lai, L.C.K. Hui, C.F. Chong, K.H. Pun, W.W. Tsang, and H.W. Chan. A rule-based bt monitoring scheme for detecting copyright infringement activities, 2007. manuscript in preparation.
4. K.P. Chow, C.F. Chong, K.Y. Lai, L.C.K. Hui, K.H. Pun, W.W. Tsang, and H.W. Chan. Digital evidence search kit. In *Proceedings of the First International Workshop on Systematic Approaches to Digital Forensic Engineering*, pages 187–194, 2005.
5. The Government of Hong Kong Special Administrative Region Hong Kong Police. Technology crime statistics in hong kong, 2005.
6. IFPI. Ifpi external press pack, 2005.
7. The Government of Hong Kong Special Administrative Region Intellectual Property Department. Awareness of protection of intellectual property rights increases, 2006.
8. T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is P2P dying or just hiding? In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'04)*, volume 3, pages 1532–1538, 2004.
9. D. Moore, G.M. Voelker, and S. Savage. Inferring internet denial-of-service activity. In *Proceedings of the 10th USENIX Security Conference*, pages 9–22, 2001.
10. Hong Kong Ordinances. Evidence ordinance. Chapter 8.
11. A. Parker. Peer-to-peer in 2005, 2005. CacheLogic Research.
12. Bruce Schneier and John Kelsey. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999.
13. P. Vixie, G. Sneeringer, and M. Schleifer. Event report, events of 21-oct-2002, 2002.

A Linear Analysis of Blowfish and Khufu

Jorge Nakahara Jr.

UNISANTOS, Brazil
jorge_nakahara@yahoo.com.br

Abstract. This paper[1] describes a linear analysis of Blowfish (a block cipher designed by B. Schneier in 1993), and Khufu (a cipher designed by R.C. Merkle in 1989). The nonlinear cipher components of these ciphers are key dependent, and thus, unknown to unauthorized entities. Nonetheless, we estimate the fraction of user keys that generate weak nonlinear components (namely, with large enough bias). As far as we are aware of this paper reports the first known-plaintext (and ciphertext-only) attacks on these ciphers.

Keywords: Blowfish, Khufu, linear cryptanalysis, key-dependent S- boxes.

1 Introduction

This paper describes known-plaintext attacks on the Blowfish and Khufu block ciphers. Previous attacks reported in the literature on both of them operate in a chosen-plaintext setting.

For instance, Vaudenay in [25] examined a simplified variant of Blowfish [21] with the S-boxes known and not key-dependent. For this variant, a differential attack can recover the P-array with 2^{8r+1} chosen plaintexts (CP), where r is the number of rounds. For certain weak keys that generate weak S-boxes (the odds of getting them randomly are 1 in 2^{14}), the same attack requires only 2^{4r+1} chosen plaintexts to recover the P-array, where r is the number of rounds (assuming the S-boxes are known). With unknown S-boxes, this attack can detect whether a weak key is being used, but cannot determine what it is (neither the S-boxes, nor the P-array, nor the user key itself). This attack does not work against the full 16-round Blowfish, but the discovery of weak keys in Blowfish is significant. A weak key is this context is one for which two entries of a generated S-box are identical (making it non-injective).

Rijmen, in [19], describes a differential attack on 4-round Blowfish that does not depend on weak-key assumptions.

Biham *et al.* in [2] described attacks on up to 24-round Khufu, based on key-dependent impossible differential distinguishers, based on experiments on a 24-bit block mini-version of Khufu. But, the fact that the distinguishers are

[1] Research funded by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) under contract 2005/02102-9.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 20–32, 2007.

key-dependent imply that these attacks apply only to certain (user) keys, that is, the attacks work under weak-key assumptions. Table 1 summarizes the attack complexities on 16-round Khufu.

Table 1. Attack complexities on Khufu

# Rounds	Data	Time	Source	Attack
16	2^{18} CPACC	2^{18}	[26]	Boomerang
16	2^{41} CP	2^{41}	[2]	Imposs. Differential
16	2^{43} CP	2^{43}	[10]	Differential

CPACC: Chosen-Plaintext Adaptively-Chosen Ciphertext.
CP: Chosen-Plaintext.

This paper is organized as follows: Sect. 2 briefly describes the Blowfish block cipher. Sect. 3 briefly describes the Khufu block cipher. Sect. 4 presents preliminary concepts of linear cryptanalysis. Sect. 5 describes linear attacks on Blowfish. Sects. 6 and 6.1 describes linear attacks on Khufu. Sect. 7 concludes the paper.

2 Blowfish

Blowfish is block cipher designed by B. Schneier in 1993 as a replacement for the DES cipher [18]. Blowfish operates on 64-bit text blocks under a variable-length key, having between 32 and 448 bits, in steps of 8 bits. This cipher has a Feistel Network structure with 16 rounds [21,22]. An innovative feature of Blowfish was the use of key-dependent (nonlinear) tables called S-boxes and P-arrays. The P-array consists of eighteen 32-bit values: $P_1, \ldots, P_{18}$. There are also four 8×32-bit S-boxes. The nonlinear components of Blowfish are computed as follows:

(1) initialize the P-array and the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of π, less the initial 3: $P_1 = \texttt{243f6a88}_\texttt{x}$, $P_2 = \texttt{85a308d3}_\texttt{x}$, $P_3 = \texttt{13198a2e}_\texttt{x}$, $P_4 = \texttt{03707344}_\texttt{x}$,
(2) exclusive-or P_1 with the first 32 bits of the key, exclusive-or P_2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P_{14}). Repeatedly cycle through the key bits until the entire P-array has been exclusive-ored with key bits. Note that for every short key, there is at least one equivalent longer key. For example, if A is a 64-bit key, then AA, AAA, and so on, are equivalent keys.
(3) encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).
(4) replace P_1 and P_2 with the output of step (3).
(5) encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
(6) replace P_3 and P_4 with the output of step (5).

(7) continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations and 4168 bytes of memory are required to generate and store all required subkeys. Whenever possible, applications should store the subkeys rather than execute this derivation process multiple times since running the key schedule for each new key costs 521 Blowfish encryptions. This slow key schedule means that Blowfish has a very low key agility.

The Feistel structure of Blowfish can be described as follows. Start with a 64-bit plaintext block $M_0 = (L_0, R_0)$, where L_0 and R_0 are 32-bit strings. The output of the i-th round is denoted

$$M_i = (L_i, R_i) = (R_{i-1} \oplus F(L_{i-1} \oplus P_i), L_{i-1} \oplus P_i)\,,$$

where $1 \leq i \leq 16$, and F is the round function. The ciphertext is $M_{16} = (L_{15} \oplus P_{16} \oplus P_{18}, R_{15} \oplus F(L_{15} \oplus P_{16}) \oplus P_{17})$. The round function $F : \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{32}$ combines the four 8×32 S-boxes, S_i, as follows:

$$F(X) = F(x_0|x_1|x_2|x_3) = ((S_0[x_0] \oplus S_1[x_1]) \boxplus S_2[x_2]) \oplus S_3[x_3]\,.$$

3 Khufu

Khufu is a block cipher designed by R.C. Merkle in 1989 [17] for fast encryption in software. Khufu is a Feistel Network cipher operating on 64-bit blocks, parameterized by a user key of up to 512 bits, and iterating $8r$ rounds, where $1 \leq r \leq 8$ is called the number of octets. Originally, $r = 2$ was suggested. The cipher operates on 32-bit words, with the following operations: exclusive-or, byte rotations, and table lookups (one 8×32-bit key-dependent S-box per octet). Each S-box represents $(2^8) \cdot 32 = 2^{13}$ bits of secret data, and is used for one octet only. Let a plaintext block be denoted by $P = (L_0, R_0)$, the S-box by S, a left rotation of x by n bits by $x \lll n$, and the least significant n bits of x by $\mathrm{lsb}_n x$. Then the i-th round of Khufu outputs (R_i, L_i), where $R_i = L_{i-1} \lll s_i$, and $L_i = R_{i-1} \oplus S[\mathrm{lsb}_8(L_{i-1})]$, where s_i denotes a fixed rotation amount (a multiple of 8 bits). For each round in an octet, the values of s_i are, in order, 16, 16, 24, 24, 16, 16, 8, 8, repeated cyclically. There is an input transformation in which two 32-bit subkeys, K_1 and K_2, are exclusive-ored to the plaintext, and an output transformation in which subkeys K_3 and K_4 are exclusive-ored to the output of the last round. The key schedule pre-computes the S-boxes used in each octet. Each S-box is 8×32 bits. Since the key schedule is quite time consuming, Khufu is better suited for bulk data processing, and that do not require changing keys often.

4 Linear Cryptanalysis

The linear cryptanalysis (LC) technique was discovered by Matsui and applied successfully against the DES [14] and FEAL [16] block ciphers. This technique is

one of the most general known attacks on block ciphers, and has become a benchmark technique for evaluating the security of any new modern cipher (GOST [24], AES [9], LOKI97 [6,11], RC6 [20], IDEA [8], Serpent [3]). LC is a known-plaintext (KP) attack, but it has already been used in other contexts [12,13].

The fundamental tool of a linear attack is a linear distinguisher which consists of a linear relationship between bits of plaintext, ciphertext and key, holding with non-uniform probability (different from 1/2). This discrepancy between the associated probability of a cipher and that of random behavior is called the bias, and denoted p'. The number of known plaintexts needed for a high success attack is inversely proportional to the bias $N = 8 \cdot (p')^{-2}$, according to [14]. Thus, the larger the bias, the less plaintext is needed for a high success attack. More formally, let an S-box $S : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^m$, and two bit strings, $\Gamma X \in \mathbb{Z}_2^n$ and $\Gamma Y \in \mathbb{Z}_2^m$, known as bit masks. The linear relation involving the input bits of S designated by ΓX and the output bits designated by ΓY is denoted $X \cdot \Gamma X \oplus S[X] \cdot \Gamma Y = 0$. The probability that this relation holds is

$$P_{\Gamma X, \Gamma Y} = \frac{\#\{X \in \mathbb{Z}_2^n | X \cdot \Gamma X = S[X] \cdot \Gamma Y\}}{\#\{X \in \mathbb{Z}_2^n\}} .$$

The bias of this linear relation is $|P_{\Gamma X, \Gamma Y} - 1/2|$. An exhaustive list containing all input and output bit masks of S is called the Linear Approximation Table (LAT) of S (see [14]). The LAT allows one to identify the most promising linear relations, namely the ones with highest bias.

Linear relations for individual cipher components can be derived piecewise. Then, the separate linear relations can be combined, forming relations up to round level, and further on to multiple rounds. The bias of each combination of two approximations is derived using Matsui's Piling-Up Lemma [14]. We will employ this lemma in our analysis, even though it is not strictly correct [5,23].

For both Blowfish and Khufu, we looked for attacks that did not require more text than the full codebook (2^{64} text blocks for both ciphers). Moreover, we assume that user keys are at least 64 bits long, so that an exhaustive key search is not better than collecting the full codebook.

Distinguishers can be used either to distinguish a cipher from a random permutation or in a key-recovery attack. In the latter, there is usually a separation between the distinguisher itself (some rounds of the cipher) and the rest of the cipher, in which subkeys are recovered. Sometimes, this boundary is not clear as in [4], in which subkeys within the distinguisher itself are recovered.

5 Linear Analysis of Blowfish

Unlike the differential attacks in [25], our linear analysis does not depend on duplicate entries (collisions) in the S-boxes, namely, we do not require the S-boxes to be non-injective mappings.

Since the S-boxes and P-arrays in Blowfish are key-dependent, an adversary would not be able to compute the LAT, because these nonlinear components are unknown. But, this fact does not forbid one from computing the LAT for

a sample of S-boxes derived from a random set of keys. Moreover, since the dimensions of Blowfish's S-boxes are fixed (8×32), they are non-surjective. Therefore, we looked for linear relations for these S-boxes with the form $0 \overset{S-box}{\rightarrow} \Gamma$, where 0 stands for the null 8-bit mask, and Γ stands for a nonzero (non-trivial) 32-bit mask. This notation means that the exclusive-or of no input bits (to the S-boxes) can equal an exclusive-or of output bits selected by Γ.

Due to the modular addition operation in the F function of Blowfish, we looked for bit masks Γ with nonzero bits preferably in the least significant bit positions. This choice avoids decreasing the bias due to the carry bits. The best tradeoff for the bit mask value is $\Gamma = 1$ (Fig. 1). Thus, the linear relations we

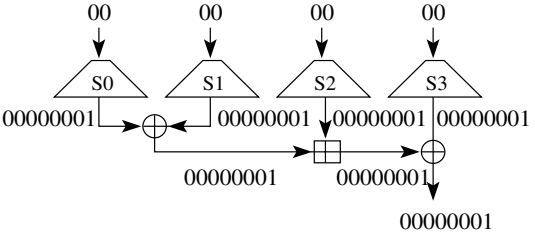


Fig. 1. Bit masks (in hex) showing the propagation of a linear relation across the F function of Blowfish

chose for the F functions have either the form[2] $\texttt{00000000}_\texttt{x} \overset{F}{\rightarrow} \texttt{00000000}_\texttt{x}$ or the form $\texttt{00000000}_\texttt{x} \overset{F}{\rightarrow} \texttt{00000001}_\texttt{x}$. The rationale is to minimize the number of active S-boxes in building linear relations across the round function (a strategy already used against the DES cipher [15]). Nonetheless, due to the construction of the F function, all four S-boxes are active under the second linear relation to the F function. As for the P-arrays, unlike the S-boxes, there is only one P-array per round and they act like unknown constants exclusive-ored to the left-half of each text block prior to the F function. The P-arrays do not affect the bias of linear approximations of the F functions. For our linear bit masks, only the least significant bit (LSB) is involved in the approximation, and its exact value does not matter in the P-arrays (this LSB only changes the sign of the bias).

We exploited iterative linear relations, namely, linear relations that can be concatenated with themselves. Consequently, it leads to a fixed decrease in the bias for every concatenation. We arrive at the 2-round iterative linear relations in Fig. 2(a) and 2(b), with one active F function and four active S-boxes for every two rounds. Looking closely, one can notice many dualities between our linear distinguisher in Fig. 2 and the differential distinguisher in [25]: while Fig. 2 uses nonzero bit masks only at the output of the S-boxes and round function, the nonzero differences in [25] appear only at the input to the same S-boxes and round functions. While Fig. 2 exploits the least significant bit positions, the

[2] The subscript $\texttt{x}$ indicates hexadecimal value.

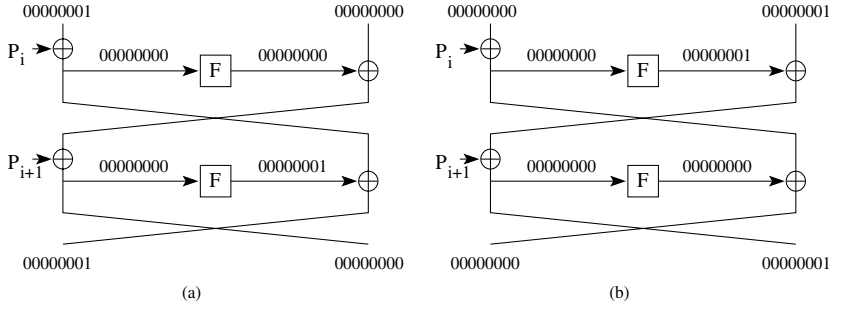


Fig. 2. Two-round iterative linear relations for Blowfish

differences in [25] exploit the most significant bit positions (taking advantage of the discarded carry bit from the leftmost bit position).

The linear distinguishers in Fig. 2 applied to $2t$-round Blowfish allow to recover two bits of information of the P-array, based on an exclusive-or combination of bits from the plaintext and ciphertext: $(L_0 \oplus L_{2t}) \cdot \mathtt{00000001_x} = (P_1 \oplus P_3 \oplus \ldots \oplus P_{2t+1}) \cdot \mathtt{00000001_x}$ (using Fig. 2(a)), and $(R_0 \oplus R_{2t}) \cdot \mathtt{00000001_x} = (P_2 \oplus P_4 \oplus \ldots \oplus P_{2t}) \cdot \mathtt{00000001_x}$ (using Fig. 2(b)).

We have estimated how many user keys lead to S-boxes whose LATs [14] contain the highly (and nonzero) biased entry for the input/output masks ($\mathtt{00_x}$, $\mathtt{00000001_x}$). In particular, if any single S-box has a zero entry in the LAT for these masks, then we call the particular user key that generated that S-box strong against our linear distinguisher (although, not necessarily strong against all possible linear distinguishers). Thus, this analysis raises the question on weak linear keys in Blowfish.

In [1], Biham has predicted that a $m \times n$ S-box in which "if $n \geq 2^m$, the S-box must have an affine combination of only output bits, which does not depend on the input bits at all. Such combinations cause ... the existence of a 2-round iterative characteristic with probability $1/2 \pm 1/2$ (of the form $0 \rightarrow X$), and thus enable attacks which require a few known plaintexts.". Our results for Blowfish are twofold. On one hand, we have $m = 8$, $n = 32$, but $32 < 2^8$. On the other hand, we have identified some keys for which the linear relation $\mathtt{00_x} \rightarrow \mathtt{00000001_x}$ holds with nonzero probability, but it is not $1/2 \pm 1/2$. The linear approximation probability in our case ranges between these two extremes.

Table 2 lists the results of bias estimations on Blowfish for about $2^{28.97}$ randomly selected keys (that is, randomly selected S-boxes), and their susceptibility to the linear distinguisher in Fig. 2. In our simulations we have used Paul Kocher's C-language implementation of Blowfish.

The #KP in Table 2 is estimated as $8 * (\text{bias})^{-2}$, according to Matsui [14]. This number must be smaller than the codebook size, 2^{64}. The number of rounds in Table 2 stands for how many rounds can be attacked using Fig. 2. We notice that a fraction of 18.51% of the keys tested, at least one of the resulting S-boxes have a zero bias (trivial linear relation), therefore, avoiding Fig. 2 altogether. These user keys are strong (linear) keys. A fraction of about 78.29% of the keys tested

have a bias higher than 2^{-30}, thus allowing a linear attack up to two rounds requiring less than 2^{63} KP (and equivalent encryption effort and memory). A fraction of about 3.19% of the keys tested have a bias higher than $2^{-15.75}$, thus allowing a linear attack up to four rounds requiring less than 2^{64} KP. A tiny fraction of only 0.0000013% of the keys tested have a bias higher than $2^{-10.83}$, thus allowing a linear attack up to six rounds requiring less than $2^{63.98}$ KP.

Table 2. Simulation results of bias of linear distinguisher in Fig. 2 on a random sample of user keys of Blowfish

#Rounds	Bias	Fraction of Keys
0	0	18.51%
2	$> 2^{-30}$	78.29%
4	$> 2^{-15.75}$	3.19%
6	$> 2^{-10.83}$	0.0000013%
8	$> 2^{-8.25}$	—

We have not found any particular pattern in the user keys that allowed us to identify precisely a weak linear key (namely, weak under a linear attack). An alternative approach to identify the susceptibility of a Blowfish instance to our linear attack is to simply compute the entry ($00_{\mathtt{x}}$, $00000001_{\mathtt{x}}$) in the LAT of each of the four generated S-boxes and verify their combined bias (using the Piling-Up Lemma). If the resulting bias is zero, then definitely the given user key is not vulnerable to our linear distinguisher. Otherwise, a linear attack (for variable number of rounds) using relation in Fig. 2 may be possible depending on the magnitude of the bias.

Additionally, we have analysed the alternative bit masks $00000002_{\mathtt{x}}$ and $00000003_{\mathtt{x}}$. In these cases we need to take into account a decrease in the bias due to a carry bit into the second least significant bit position in the linear approximation. Before accounting for this decrease in the bias, we have looked at a sample of randomly selected user keys, and we have not detected a high enough bias in the S-boxes. Thus, we find no advantage of the latter over the original mask $00000001_{\mathtt{x}}$. For this same reason, we did not use the bit mask $80000000_{\mathtt{x}}$ (or similarly $80800000_{\mathtt{x}}$, $00808000_{\mathtt{x}}$, and so on). On the one hand they could lead to a ciphertext-only attack (on the assumption that the plaintext was ASCII coded). But, on the other hand, the avalanche of carry bits accumulating up to the most significant bit positions (on the byte boundaries) would excessively decrease the bias, making the attack infeasible.

6 Linear Analysis of Khufu

Similar to Blowfish S-boxes, the ones for Khufu are also key-dependent, and have exactly the same dimensions, 8×32 bits. The S-boxes change every eight rounds, but only one S-box is used per round. Moreover, the round function of Khufu

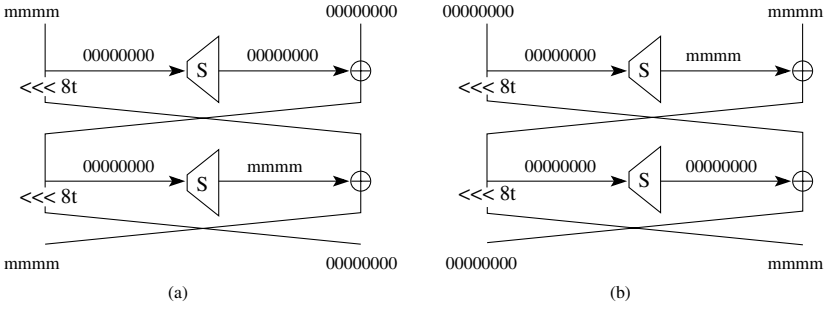


Fig. 3. Two-round iterative linear relations for Khufu

uses bit rotations instead of the exclusive-or with a P-array, as in Blowfish. An important observation is that the bit rotations are always done in multiples of eight bits, which motivates the format of our linear bit masks. Thus, our linear masks for the S-boxes have the form $\mathtt{00}_{\mathtt{x}} \xrightarrow{S} \mathtt{mmmm}_{\mathtt{x}}$, where $m \in \mathbb{Z}_2^8 - \{0\}$. Only the output bit mask is nonzero. The reason for the repetition of the byte value m is that it is invariant (or rotation-symmetric) with respect to the variable bit rotation in Khufu. Namely, $\mathtt{mmmm}_{\mathtt{x}} \lll 8t = \mathtt{mmmm}_{\mathtt{x}}$, where t is a multiple of 8. The rationale is to construct iterative linear relations, just as we did for Blowfish. Notice that $\mathtt{mmmm}_{\mathtt{x}} \ggg 8t = \mathtt{mmmm}_{\mathtt{x}}$ also holds. Thus, the direction of rotation is irrelevant.

Since we aim at iterative linear relations with trivial input masks, we could not make one-round linear distinguishers. The first linear relations we got for Khufu are 2-round iterative (Fig. 3(a) and 3(b)).

Our simulations with a C-code implementation of Khufu (by R.C. Merkle) with linear approximations using masks $\mathtt{mmmm}_{\mathtt{x}}$, $m \in \mathbb{Z}_2^8 - \{0\}$, for a typical S-box are summarized in Table 3. The nonzero biases are relatively high, ranging between 2^{-3} and 2^{-6}.

Thus, a fraction of 14.9% of the masks with the form $\mathtt{mmmm}_{\mathtt{x}}$, $m \in \mathbb{Z}_2^8 - \{0\}$, has zero bias, thus avoiding our linear distinguishers in Fig. 3 altogether.

A fraction of about 3% of the masks with the form $\mathtt{mmmm}_{\mathtt{x}}$, $m \in \mathbb{Z}_2^8 - \{0\}$, has bias 2^{-3} for a 2-round relation in Fig. 3. It allows one to mount a linear attack on 24-round Khufu, to recover one-bit of information on the (key-dependent) S-box and of K_1 or K_2, using about $8 \cdot (2^{11-12*3})^{-2} = 2^{53}$ KP, according to [14].

Table 3. Simulation results of bias of S-boxes of Khufu for random keys

# Masks $\mathtt{mmmm}_{\mathtt{x}}$, $0 < m < 256$	Bias	Fraction of Keys
38	0	14.9%
13	2^{-3}	5%
66	2^{-4}	25.88%
98	2^{-5}	38.43%
40	2^{-6}	15.68%

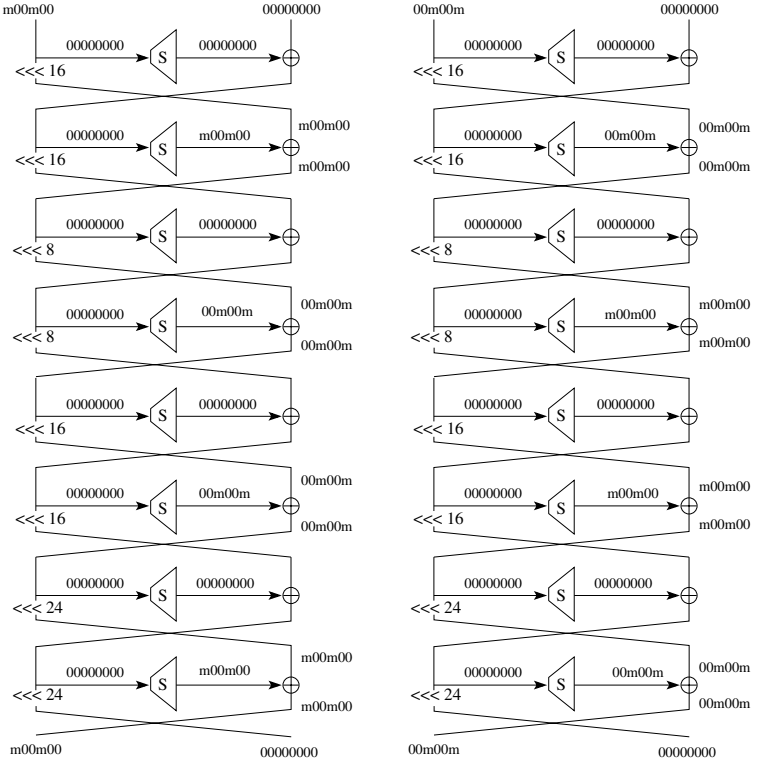


Fig. 4. Two 8-round iterative linear relations for Khufu

We recover only one bit of information because the bitmask is 1-bit wide and the S-boxes are unknown.

For a fraction of 25.88% of the masks with the form $\texttt{mmmm}_{\texttt{x}}$, $m \in \mathbb{Z}_2^8 - \{0\}$, has bias 2^{-4} for a 2-round relation in Fig. 3. It allows one to mount a linear attack on 16-round Khufu, to recover one-bit of information on the S-box, using about $8 \cdot (2^{7-8*4})^{-2} = 2^{53}$ KP.

We could have constructed longer iterative linear relations, by combining bit masks which were rotation-symmetric for different rotation amounts. This requires at least two non-trivial linear relations, such as $\texttt{00000000}_{\texttt{x}} \xrightarrow{F} \texttt{00m00m}_{\texttt{x}}$ and $\texttt{00000000}_{\texttt{x}} \xrightarrow{F} \texttt{m00m00}_{\texttt{x}}$. See Fig. 4. These iterative linear relations contain four active F functions, and, thus, four active S-boxes per octet.

Further, alternative 8-round iterative linear relations, are in Fig. 5, where $m \in \mathbb{Z}_2^8 - \{0\}$.

6.1 A Ciphertext-Only Attack on Khufu

A particularly interesting rotation-symmetric bit mask for Khufu uses the linear approximation $\texttt{00}_{\texttt{x}} \xrightarrow{F} \texttt{80808080}_{\texttt{x}}$ for its round function. Note that this mask

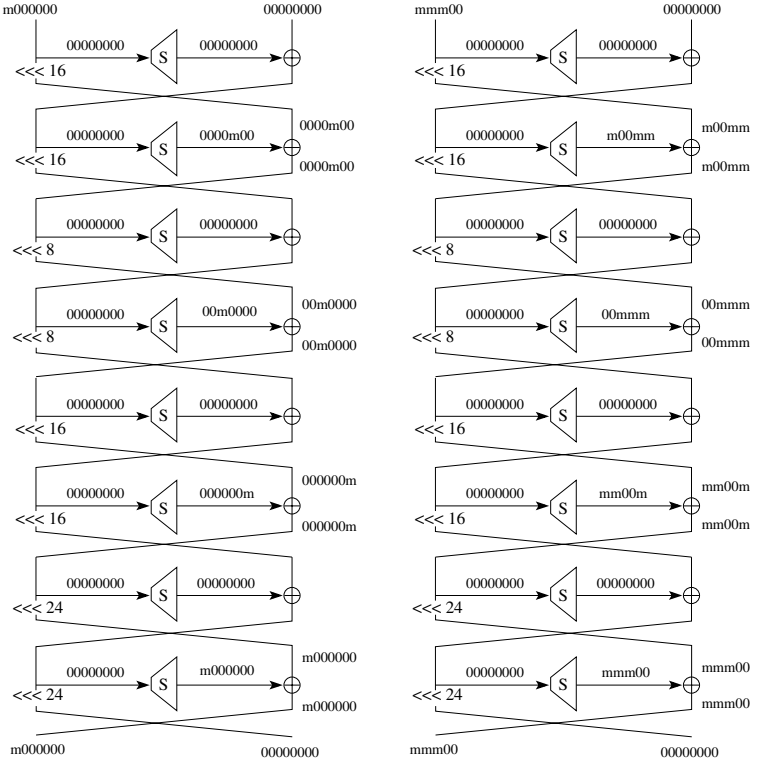


Fig. 5. More 8-round iterative linear relations for Khufu

depends only on the most significant bit of plaintext and ciphertext bytes. If the plaintext is in a natural language, made up of ASCII text only, then, an attack using this mask would work under a ciphertext-only (CO) setting.

Our simulations on a C-code implementation of Khufu (by R.C. Merkle) using bit mask $80808080_{\mathtt{x}}$ in Fig. 3 have found user keys for which the bias of the first two generated S-boxes ranged between 2^{-7} and 2^{-4}. Assuming a bias of 2^{-4}, one would be able to recover one bit of information on the key-dependent S-box, on up to 16-round Khufu, using $8 \cdot (2^{7-8*4})^{-2} = 2^{53}$ CO.

Alternatively, with a bias of 2^{-7} one could recover one bit of information on the S-box of 8-round Khufu, using $8 \cdot (2^{3-7*4})^{-2} = 2^{53}$ CO.

Our simulations with bit masks $80008000_{\mathtt{x}}$ and $00800080_{\mathtt{x}}$ in Fig. 4 have found user keys for which the bias of the first S-box (first octet) ranged between 2^{-5} and 2^{-6} for about 0.4% of the keys; between 2^{-6} and 2^{-7} for about 3% of the keys, and between 2^{-7} and 2^{-8} for about 14% of the keys. Assuming a bias of 2^{-5} would allow to recover one bit of information on the S-box, using $8 \cdot (2^{3-5*4})^{-2} = 2^{37}$ CO. Assuming a bias of 2^{-6}, the attack would cost $8 \cdot (2^{3-6*4})^{-2} = 2^{45}$ CO. Assuming a bias of 2^{-7}, the attack would cost $8 \cdot (2^{3-7*4})^{-2} = 2^{53}$ CO. Assuming a bias of 2^{-8}, the attack would cost $8 \cdot (2^{3-8*4})^{-2} = 2^{61}$ CO.

Our simulation with masks 80000000_x, 00800000_x, 00008000_x, 00000080_x, in Fig. 5, did not find any user key which generate S-boxes with nonzero bias for these bit masks. Our simulation using bits masks 80808000_x, 80800080_x, 80008080_x, 00808080_x, in Fig. 5, have found user keys for which the bias of the first octet were larger or equal to 2^{-7}. About 0.012% of the keys had bias between 2^{-6} and 2^{-7}; about 0.17% of the keys has bias between 2^{-7} and 2^{-8}; The remaining keys had zero bias, or too small bias to allow a linear attack. The same attack complexities for the distinguisher in Fig. 4 in the previous paragraph also apply in this case.

7 Conclusion

This paper described linear distinguishers and attacks on Blowfish and Khufu, depending on the linear profile of its key-dependent S-boxes. As far as we are aware of, this is the first known-plaintext attack on reduced-round Blowfish and Khufu.

Although the effectiveness of these attacks depend on the unknown user key and S-boxes, our simulations demonstrate that the security of Blowfish and Khufu depends heavily on the key. Our results, though, do not affect either the security of the full 16-round Blowfish or that of the 32-round Khufu.

We did not find any bit pattern in the user key that allowed us to identify whether any particular key value is weak or strong (concerning the linear distinguishers in Fig. 2 and Fig. 3).

Anyway, a simple test for a user worried about the susceptibility of a particular Blowfish instance to our linear attacks is to compute the entry (00_x, 00000001_x) in the LAT for each of the four generated S-boxes and verify if their combined bias (using the Piling-Up Lemma) is zero or too small to allow an attack on

Table 4. Summary of linear attacks on Blowfish and Khufu (fraction of keys are estimated)

Cipher	# Rounds	Data/Mem./Time	Comments
Khufu	8	2^{37} CO	$p' = 2^{-5}$ per S-box, 80008000_x, 00800080_x
	8	2^{45} CO	$p' = 2^{-6}$ per S-box, 80008000_x, 00800080_x
	8	2^{53} CO	$p' = 2^{-7}$ per S-box, 80008000_x, 00800080_x
	8	2^{53} CO	$p' = 2^{-4}$ per S-box, mask 80808080_x
	8	2^{61} CO	$p' = 2^{-8}$ per S-box, 80008000_x, 00800080_x
	16	2^{53} CO	$p' = 2^{-7}$ per S-box, mask 80808080_x
	16	2^{53} KP	$p' = 2^{-3}$ per S-box, mask $\texttt{mmmm}_x$
	24	2^{53} KP	$p' = 2^{-4}$ per S-box, mask $\texttt{mmmm}_x$
Blowfish	2	$< 2^{63}$ KP	$p' > 2^{-30}$, 78.1% of keys
	4	$< 2^{64}$ KP	$p' > 2^{-15.75}$, 3.21% of keys
	6	$< 2^{64}$ KP	$p' > 2^{-10.83}$, 0.000002% of keys

KP: Known Plaintext
CO: Ciphertext Only

his particular instance of Blowfish. This way, a user can filter potential weak keys prior to encryption. The same approach applies to Khufu under Fig. 4 and Fig. 5.

Table 4 summarizes the known-plaintext and ciphertext-only attacks on Blowfish and Khufu.

References

1. E. Biham, *On Matsui's Linear Cryptanalysis*, Technion, CS Dept. Technical Report CS0813, 1994.
2. E. Biham, A. Biryukov, A. Shamir, *Miss-in-the-Middle Attacks on IDEA, Khufu and Khafre*, 6th Fast Software Encryption Workshop, L.R. Knudsen, Ed., LNCS 1636, 1999, Springer-Verlag, 124–138.
3. E. Biham, O. Dunkelman, N. Keller, *Linear Cryptanalysis of Reduced Round Serpent*, 8th Fast Software Encryption Workshop, M. Matsui,Ed., Springer-Verlag, LNCS 2355, 2001, 16–27.
4. A. Biryukov, *The Boomerang Attack on 5 and 6-round Reduced AES*, Proceedings of AES4 Conference, H. Dobbertin, V. Rijmen, A. Sowa, Eds., Springer-Verlag, LNCS 3373, 2004, 11–15.
5. U. Blöcher, M. Dichtl, *Problems with the Linear Cryptanalysis of DES using More than One Active S-box per Round*, 1st Fast Software Encryption Workshop, R. Anderson,Ed., Springer-Verlag, LNCS 809, 1994, 256–274.
6. L. Brown, J. Pieprzyk, *Introducing the New LOKI97 Block Cipher*, 1st AES Conference, California, USA, Aug. 1998, http://csrc.nist.gov/encryption/aes/
7. J.H. Cheon, M. Kim, K. Kim, J.-Y. Lee, S. Kang, *Improved Impossible Differential Cryptanalysis of Rijndael and Crypton*, K. Kim,Ed., Springer Verlag, LNCS 2288, 2001, 39–49.
8. J. Daemen, R. Govaerts, J. Vandewalle, *Weak Keys for IDEA*, Adv. in Cryptology, Crypto'93, D.R. Stinson, Ed., Springer-Verlag, LNCS 773, 1994, 224–231.
9. J. Daemen, V. Rijmen, *The Design of Rijndael – AES – The Advanced Encryption Standard*, Springer-Verlag, 2002.
10. H. Gilbert, P. Chavaux, *A Chosen-plaintext attack on the 16-round Khufu Cryptosystem*, Advances in Cryptology, Crypto'94, Springer-Verlag, 1994, 359–368.
11. L.R. Knudsen, *Weaknesses in LOKI97*, 1999, http://csrc.nist.gov/encryption/aes/
12. L.R. Knudsen, J.E. Mathiassen, *A Chosen-Plaintext Linear Attack on DES*, 7th Fast Software Encryption Workshop, B. Schneier, Ed., Springer-Verlag, LNCS 1978, 2000, 262–272.
13. L.R. Knudsen, V. Rijmen, *Ciphertext-Only Attack on Akelarre*, Cryptologia, vol. XXIV, no. 2, Apr. 2000, 135–147.
14. M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Adv. in Cryptology, Eurocrypt'93, T. Helleseth,Ed., Springer-Verlag, LNCS 765, 1994, 386–397.
15. M. Matsui, *On Correlation Between the Order of S-boxes and the Strength of DES*, Adv. in Cryptology, Eurocrypt'94, A. De Santis,Ed., Springer-Verlag, LNCS 950, 1995, 366–375.
16. M. Matsui, A. Yamagishi, *A New Method for Known-Plaintext Attack of FEAL Cipher*, Adv. in Cryptology, Eurocrypt'92, R.A. Rueppel,Ed., Springer-Verlag, LNCS 658, 1993, 81–91.
17. R.C. Merkle, *Fast Software Encryption Functions*, Advances in Cryptology, Crypto'90, A.J. Menezes, S.A. Vanstone, Eds, LNCS 537, Springer-Verlag, 1991, 476–501.

18. NBS, *Data Encryption Standard (DES)*, FIPS PUB 46, Federal Information Processing Standards Publication 46, U.S. Department of Commerce, Jan. 1977.
19. V. Rijmen, *Cryptanalysis and Design of Iterated Block Ciphers*, Dept. Elektrotechniek, Katholieke Universiteit Leuven, Belgium, Oct. 1997.
20. R.L. Rivest, M.J.B. Robshaw, R. Sidney, Y.L. Yin, *The RC6 Block Cipher*, 1st AES Conference, California, USA, Aug. 1998, http://csrc.nist.gov/encryption/aes/
21. B. Schneier, *Blowfish–One Year Later*, Dr. Dobbs Journal, Sep. 1995.
22. B. Schneier, *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, Fast Software Encryption Workshop, R. Anderson,Ed., Cambridge Security Workshop Proceedings, Dec. 1993, Springer-Verlag, LNCS 809, 191-204.
23. A.A. Selçuk, *On Bias Estimation in Linear Cryptanalysis*, Progress in Cryptology - INDOCRYPT 2000, B. Roy, E. Okamoto, Eds., Springer-Verlag, LNCS 1977, 2000, 52–66.
24. V.V. Shorin, V.V. Jelezniakov, E.M. Gabidulin, *Linear and Differential Cryptanalysis of Russian GOST*, Proc. of Workshop on Coding and Cryptography 2001, Jan. 2001, D. Augot,Ed., 467–476.
25. S. Vaudenay, *On the Weak Keys of Blowfish*, Technical Report, Liens - 95- 27, Ecole Normale Superieure.
26. D. Wagner, *The Boomerang Attack*, 6th Fast Software Encryption Workshop, L.R. Knudsen, Ed., LNCS 1636, 1999, Springer-Verlag, 156–170.

Related-Key Rectangle Attack on 43-Round SHACAL-2*

Gaoli Wang

School of Mathematics and System Sciences, Shandong University,
Jinan, China
wanggaoli@mail.sdu.edu.cn

Abstract. SHACAL-2 is a 256-bit block cipher with up to 512 bits of key length based on the hash function SHA-2. It was recommended as one of the NESSIE projection selections. As far as the number of the attacked rounds is concerned, the best cryptanalytic result obtained on SHACAL-2 so far is the analysis of a related-key rectangle attack on the 42-round SHACAL-2 [13]. In this paper we present a related-key rectangle attack on 43-round out of the 64-round of SHACAL-2, which requires $2^{240.38}$ chosen plaintexts and has time complexity of $2^{480.4}$ 43-round SHACAL-2 encryptions. In this paper we also identify and fix some flaws in previous attack on SHACAL-2.

Keywords: Block cipher, SHACAL-2, Related-Key Rectangle attack, Differential characteristic.

1 Introduction

Differential cryptanalysis [3] is one of the most powerful known attacks on block ciphers, which was introduced by E. Biham and A. Shamir in 1990.

The related-key attack [4] was introduced by E. Biham in 1993, in which the attacker chooses the relationship between two unknown keys. The attack is based on a key scheduling algorithm and shows that a block cipher with a weak key scheduling algorithm may be vulnerable to this kind of attack. Many cryptanalytic results of the attack were presented in [14,15,16,17].

The related-key boomerang and rectangle attacks were proposed by Kim et al. [8,9] and independently by Biham et al. [6]. This attack is a combination of the related-key and the rectangle attacks, and shares the features of rectangle and related-key attacks. The attacker examines quartets of plaintexts encrypted under four related keys. This attack exploits two types of related-key rectangle distinguishers to retrieve the related keys. Our distinguishers can be used in analyzing block ciphers which have a good related-key differential followed by another good related-key differential or which have a good related-key differential followed by a good differential.

* Supported by National Natural Science Foundation of China Key Project No.90604036, National Outstanding Young Scientist No.60525201 and 973 Program No.2007CB807902.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 33–42, 2007.

SHACAL-2 [2] is a 256-bit block cipher with up to 512-bit key length based on the hash function SHA-2. It was submitted to the NESSIE project (New European Schemes for Signatures, Integrity, an Encryption) and was recommended as one of the NESSIE projection selections. It has 64 rounds. The best cryptanalytic result obtained on SHACAL-2 so far is the analysis of a related-key rectangle on 42-round SHACAL-2 [13]. See Table 1 for a summary of our results and the comparison with the previous attacks.

Table 1. Comparison of our results with the previous attacks on SHACAL-2

Type of Attack	Number of Rounds	Complexity Data/Time/Memory
Impossible Differential	30	$744CP/2^{495.1}/2^{14.5}$[10]
Differential-Nonlinear	32	$2^{43.4}CP/2^{504.2}/2^{48.4}$[11]
Square-Nonlinear	28	$463 \cdot 2^{32}CP/2^{494.1}/2^{45.9}$[11]
Related-Key Differential-Nonlinear	35	$2^{42.32}$RK-CP$/2^{452.10}/2^{47.32}$[12]
Related-Key Rectangle	37	$2^{233.16}$RK-CP$/2^{484.95}/2^{238.16}$[12]
	40	$2^{243.38}$RK-CP$/2^{448.43}/2^{247.38}$[13]
	42	$2^{243.38}$RK-CP$/2^{488.37}/2^{247.38}$[13]
	43	$2^{240.38}$RK-CP$/2^{480.4}/2^{245.38}(New)$

CP: Chosen Plaintexts, RK-CP: Relate-Key Chosen Plaintexts,
Time: Encryption units, Memory: Bytes of memory

The rest of the paper is organized as follows: In Section 2, we introduce some useful properties of the nonlinear functions in SHACAL-2 and some notations, and give a short description of the related-key rectangle attack. In Section 3, we describe the related-key rectangle attack on 43-round SHACAL-2. Finally, we summarize the paper in section 4.

2 Background

2.1 Description of SHACAL-2

SHACAL-2 [2] is a 256-bit block cipher based on the compression function of the hash function SHA-2. The algorithm is composed of 64 rounds with variable key length of up to 512-bit, and it is advised to use keys of at least 128-bit.

For a 256-bit plaintext $P = A_0\|B_0\|C_0\|D_0\|E_0\|F_0\|G_0\|H_0$ the corresponding 256-bit ciphertext C is denoted by $A_{64}\|B_{64}\|C_{64}\|D_{64}\|E_{64}\|F_{64}\|G_{64}\|H_{64}$. The $r-th$ round of encryption is as follows.

$$T^1_{i+1} = H_i + g_1(E_i) + G_1(E_i, F_i, G_i) + Con_i + K_i \quad (1)$$
$$T^2_{i+1} = g_0(A_i) + G_0(A_i, B_i, C_i) \quad (2)$$
$$H_{i+1} = G_i \quad (3)$$
$$G_{i+1} = F_i \quad (4)$$
$$F_{i+1} = E_i \quad (5)$$

$$E_{i+1} = D_i + T^1_{i+1} \tag{6}$$
$$D_{i+1} = C_i \tag{7}$$
$$C_{i+1} = B_i \tag{8}$$
$$B_{i+1} = A_i \tag{9}$$
$$A_{i+1} = T^1_{i+1} + T^2_{i+1} \tag{10}$$

for $i = 0, ..., 63$ where $+$ denotes the addition modulo 2^{32} of 32-bit words, K_i are the 32-bit round subkeys, and Con_i denotes the 32-bit round constants which are different in each of the 64 rounds. The function in the above encryption process are as follows.

$$G_1(X,Y,Z) = I(X,Y,Z) = (X \wedge Y) \oplus (\neg X \wedge Z)$$
$$G_0(X,Y,Z) = J(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$$
$$g_0(X) = ROTR_2(X) \oplus ROTR_{13}(X) \oplus ROTR_{22}(X)$$
$$g_1(X) = ROTR_6(X) \oplus ROTR_{11}(X) \oplus ROTR_{25}(X)$$

where $\neg X$ denotes the complement of 32-bit word X and $ROTR_i(X)$ means the right rotation of X by i bit positions.

The key scheduling algorithm of SHACAL-2 supports a maximum 512-bit key and shoter keys are padded by zeros to a 512-bit string. For a 512-bit key string $K = K_0K_1, ..., K_{15}$ the key expansion is as follows.

$$K_i = h_1(K_{i-2}) + K_{i-7} + h_0(K_{i-15}) + K_{i-16}, \ (16 \le i \le 63)$$
$$h_1(X) = ROTR_7(X) \oplus ROTR_{18}(X) \oplus SR_3(X)$$
$$h_0(X) = ROTR_{17}(X) \oplus ROTR_{19}(X) \oplus SR_{10}(X)$$

where SR_i denotes the right shift of 32-bit word X by i bit positions.

2.2 Some Basic Conclusions and Notations

In this section we will present some properties of the two nonlinear functions in our attack.

Proposition 1. *For the nonlinear function* $I(X,Y,Z) = (X \wedge Y) \oplus (\neg X \wedge Z)$ *, there are the following properties:*

1. $I(x,y,z) = I(\neg x,y,z)$ *if and only if* $y = z$.
   $I(0,y,z) = 0$ *and* $I(1,y,z) = 1$ *if and only if* $y = 1$ *and* $z = 0$.
   $I(0,y,z) = 1$ *and* $I(1,y,z) = 0$ *if and only if* $y = 0$ *and* $z = 1$.
2. $I(x,y,z) = I(x,\neg y,z)$ *if and only if* $x = 0$.
   $I(x,0,z) = 0$ *and* $I(x,1,z) = 1$ *if and only if* $x = 1$.
3. $I(x,y,z) = I(x,y,\neg z)$ *if and only if* $x = 1$.
   $I(x,y,0) = 0$ *and* $I(x,y,1) = 1$ *if and only if* $x = 0$.

Proposition 2. *For the nonlinear function* $J(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$ *, there are the following properties:*

1. $J(x,y,z) = J(\neg x,y,z)$ *if and only if* $y = z$.
   $J(0,y,z) = 0$ *and* $J(1,y,z) = 1$ *if and only if* $y = \neg z$.
2. $J(x,y,z) = J(x,\neg y,z)$ *if and only if* $x = z$.
   $J(x,0,z) = 0$ *and* $J(x,1,z) = 1$ *if and only if* $x = \neg z$.
3. $J(x,y,z) = J(x,y,\neg z$ *if and only if* $x = y$.
   $J(x,y,0) = 0$ *and* $J(x,y,1) = 1$ *if and only if* $x = \neg y$.

Notations. In order to describe our attack conveniently, we quote some notations.

1. The bit positions in a 32-bit word are labeled as $31, 30, 29, \ldots, 2, 1, 0$, where bit 31 is the most significant bit and bit 0 is the least significant bit.
2. $A_{i,j}$, $B_{i,j}$, $C_{i,j}$, $D_{i,j}$, and $E_{i,j}$ represent respectively the $j-th$ bit of A_i, B_i, C_i, D_i, and E_i where the least significant bit is the 1-st bit, and the most significant bit is the 32-th bit.
3. e_j represent the 32-bit word composed of 31 $0's$ and 1 in the $j-th$ place, $e_{j,k} = e_j \oplus e_k$ and $e_{j,k,l} = e_j \oplus e_k \oplus e_l$, etc.
4. $\Delta(A,B)$ denotes the difference between A and B.

2.3 Short Description of the Related-Key Rectangle Attack

The related-key rectangle attack was introduced in [8,9] and independently in [6]. Here we give a short description of this attack. Assume that a block cipher $E : \{0,1\}^n \times \{0,1\}^k \longrightarrow \{0,1\}^n$ can be described as $E = E_1 \cdot E_0$, such that there is a related-key differential $\alpha \longrightarrow \beta$ with probability p_β for E_0, and there is a related-key differential $\gamma \longrightarrow \delta$ with probability q_γ for E_1, i.e.,

$$Pr[\Delta(E_0(X,K), E_0(X^*,K^*)) = \beta | \Delta(X,X^*) = \alpha, \Delta(K,K^*) = \Delta K^*] = p_\beta$$

$$Pr[\Delta(E_1(Y^*,K^*), E_1(Y'^*,K'^*)) = \delta | \Delta(Y^*,Y'^*) = \gamma, \Delta(K^*,K'^*) = \Delta K'] = q_\gamma$$

We use the master key K and the related keys K^*, K' and K'^* with difference $\Delta(K,K^*) = \Delta(K',K'^*) = \Delta K^*$ and $\Delta(K,K') = \Delta(K^*,K'^*) = \Delta K'$. The related-key rectangle distinguisher is as follows:

1. Choose m_1 plaintext pairs (P_i, P_i^*) at random such that $\Delta(P_i,P_i^*) = \alpha$. Encrypt P_i and P_i^* under E_0 with key K and K^* respectively to get the intermediate values X_i and X_i^*. Encrypt X_i and X_i^* under E_1 with key K and K^* respectively to get the ciphertexts C_i and C_i^*.
2. Choose m_2 plaintext pairs $(P_j', P_j'^*)$ at random such that $\Delta(P_j',P_j'^*) = \alpha$. Encrypt P_j' and $P_j'^*$ under E_0 with key K' and K'^* respectively to get the intermediate values X_j' and $X_j'^*$. Encrypt X_j' and $X_j'^*$ under E_1 with key K' and K'^* respectively to get the ciphertexts C_j' and $C_j'^*$.
3. Search two pairs of plaintexts $P_i, P_i^\star$ and $P_j', P_j'^*$, and their corresponding ciphertexts C_i, C_i^* and $C_j', C_j'^*$ respectively, satisfying: $\Delta(P_i,P_i^*) = \Delta(P_j',P_j'^*) = \alpha$, $\Delta(X_i,X_i^*) = \Delta(X_j',X_j'^*) = \beta$, $\Delta(X_i,X_j') = \Delta(X_i^*,X_j'^*) = \gamma$, and $\Delta(C_i,C_j') = \Delta(C_i^*,C_j'^*) = \delta$.

A plaintext quartet $(P_i, P_i^*, P_j', P_j'^*)$ satisfying all these conditions is called a right quartet. More generally, a right quartet represents one which satisfies any β and γ difference conditions for given α and δ differences. As described in [7,8,9], the expected number of right quartets is $\sum_{\beta\gamma} m_1 m_2 2^{-n} p_\beta^2 q_\gamma^2 = m_1 m_2 2^{-n} p^2 q^2$, where $p = (\sum_\beta p_\beta^2)^{\frac{1}{2}}$, $q = (\sum_\gamma q_\gamma^2)^{\frac{1}{2}}$. For a random permutation the expected number of right quartets is $m_1 m_2 2^{-2n}$. Therefore as long as $pq > 2^{-\frac{n}{2}}$ we can distinguish between a random permutation and E, and use this distinguisher later to recover the key.

3 Related-Key Rectangle Attack on 43-Round SHACAL-2

As stated earlier, as far as the number of the attacked rounds is concerned, the best cryptanalytic result obtained on SHACAL-2 so far is the analysis of a related-key rectangle attack on 42-round SHACAL-2 [13]. They chose two pools of plaintexts of $2^{178.38} \times 2^{64} = 2^{242.38}$ each, and presented 12 bits conditions of the intermediate values, which will remove the differential probability incurred by the G_0 and G_1 functions in Rounds 1 and 2. They concluded that after Step 1, there remains $2^{242.38} \times 2^{-12} = 2^{230.38}$ intermediate values of each pool, then the expected number of the right quartets is $(2^{230.38})^2/2 \times 2^{-456.76} = 2^3$, where the distinguisher holds with probability $2^{-456.76}$. From the differential characteristic for E_0 in [13], we know that the differential in Step 0 is $(0, e_M, e_{31}, ?, e_{9,13,19}, e_{18,29}, e_{31}, ?) \longrightarrow (0, 0, e_M, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31})$. Obviously it needs some conditions of plaintext to ensure that the differential holds with probability 1. But [13] didn't present any condition of plaintexts. There is another flaw in [13] as follows. Considering $Q^l_{i_0,j_0} \oplus Q^l_{i_1,j_1}$ and $Q^{*l}_{i_0,j_0} \oplus Q^{*l}_{i_1,j_1}$, where $Q^l_{i_0,j_0}$, $Q^l_{i_1,j_1}$ are the intermediate values of S_i, and $Q^{*l}_{i_0,j_0}$, $Q^{*l}_{i_1,j_1}$ are the intermediate values of S_i^*, so it is sufficient to guess the subkeys k^l and k^{*l}, and it is not necessary to guess the additive difference between the subkeys k^l and k^{*l}. Therefore, there are some flaws in the attack procedure of the 42-round analysis in [13].

Our attack is based on the following observation.

Observation 1. Suppose the plaintext P_0 and P_1 are encrypted using the same key, and we know the actual values of $(A_0^i, B_0^i, C_0^i, D_0^i, E_0^i, F_0^i, G_0^i, H_0^i)$ and $(A_1^i, B_1^i, C_1^i, D_1^i, E_1^i, F_1^i, G_1^i, H_1^i)$, then we know the actual values of $(A_0^{i-1}, B_0^{i-1}, C_0^{i-1}, D_0^{i-1}, E_0^{i-1}, F_0^{i-1}, G_0^{i-1})$, $(A_1^{i-1}, B_1^{i-1}, C_1^{i-1}, D_1^{i-1}, E_1^{i-1}, F_1^{i-1}, G_1^{i-1})$ and the additive difference between H_0^{i-1} and H_1^{i-1}, hence we know the actual values of $(A_0^{i-5}, B_0^{i-5}, C_0^{i-5})$ and $(A_1^{i-5}, B_1^{i-5}, C_1^{i-5})$, and the additive difference between D_0^{i-5} and D_1^{i-5}.

3.1 Related-Key Differential Characteristics for SHACAL-2

In our attack, we use the differential characteristics based on [13], and our differential in Step 0 is

$$(0, e_M, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}, \Delta_{i,j}) \longrightarrow (0, 0, e_M, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31})$$

where $g_1(E^0 \oplus e_{9,13,19}) - g_1(E^0) + \Delta_{i,j} = 0$. From Prop.1 and Prop.2, the probability of Step 0 will be 1 if we fix some bits conditions presented in Table 2. Since $D^2 = B^0$, $H^2 = F^0$, according to the encryption algorithm, the probability of Step 2 will be increased up to 2^{-10} by the conditions $B_{0,i} = \neg F_{0,i} (i = 18, 29)$. From [13] we know that the probability from Step 2 to Step 24 is 2^{-37}, so the probability of our first differential characteristic is 2^{-46}. As stated in [13], the second differential characteristic is $2^{-63.38}$. So the 35-round related-key rectangle distinguisher holds with probability $2^{-474.76}$.

Table 3 present the details of the first 25-round related-key differential characteristic. The difference of the master keys is $(e_{31}, 0, 0, 0, 0, 0, 0, 0, e_{31}, 0, 0, 0, 0, 0, 0, 0)$.

Table 4 presents the details of the second 10-round related-key differential characteristic. This differential characteristic use the same master key.

Table 2. The fixed plaintext bits for SHACAL-2

A_0	B_0	E_0	F_0
$A_{0,31} = B_{0,31}, A_{0,i} = C_{0,i}$	$B_{0,i} = \neg F_{0,i} (i = 19, 30)$	$E_{0,31} = 0$	$F_{0,i} = G_{0,i}$
$(i = 6, 9, 18, 20, 25, 29)$	$B_{0,9} = \neg E_{0,9}$	$E_{0,i} = 0 (i = 18, 29)$	$(i = 9, 13, 19)$

Table 3. The First Related-Key Differential Characteristic for SHACAL-2

i	ΔA_i	ΔB_i	ΔC_i	ΔD_i	ΔE_i	ΔF_i	ΔG_i	ΔH_i	ΔK_i	$Prob.$
0	0	e_M	e_{31}	0	$e_{9,13,19}$	$e_{18,29}$	e_{31}	$\Delta_{i,j}$	e_{31}	1
1	0	0	e_M	e_{31}	0	$e_{9,13,19}$	$e_{18,29}$	0	0	2^{-11}
2	e_{31}	0	0	e_M	0	0	$e_{9,13,19}$	$e_{18,29}$	0	2^{-10}
3	0	e_{31}	0	0	$e_{6,20,25}$	0	0	$e_{9,13,19}$	0	2^{-7}
4	0	0	e_{31}	0	0	$e_{6,20,25}$	$G_4[7,$	0	0	2^{-4}
5	0	0	0	e_{31}	0	0	$e_{6,20,25}$	0	0	2^{-3}
6	0	0	0	0	e_{31}	0	0	$e_{6,20,25}$	0	2^{-4}
7	0	0	0	0	0	e_{31}	0	0	0	2^{-1}
8	0	0	0	0	0	0	e_{31}	0	0	2^{-1}
9	0	0	0	0	0	0	0	e_{31}	e_{31}	1
10	0	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...
23	0	0	0	0	0	0	0	0	0	1
24	0	0	0	0	0	0	0	0	·	2^{-6}
25	$e_{13,24,28}$	0	0	0	$e_{13,24,28}$	0	0	0		

$g_1(E^0 \oplus e_{9,13,19}) - g_1(E^0) + \Delta_{i,j} = 0$, M={6,9,18,20,25,29}

3.2 The Key Recovery Attack Procedure for 43-Round SHACAL-2 with 512-Bit Keys

Assume that the master key is K and the related keys are K^* with differences $\Delta K = (e_{31}, 0, 0, 0, 0, 0, 0, 0, 0, e_{31}, 0, 0, 0, 0, 0, 0, 0)$. We will present a method to exploit the 35-round related-key rectangle distinguisher to find a master key

Table 4. The Second Related-Key Differential Characteristic for SHACAL-2

$Round(i)$	ΔA_i	ΔB_i	ΔC_i	ΔD_i	ΔE_i	ΔF_i	ΔG_i	ΔH_i	$Prob.$
25	e_{31}	e_{31}	$e_{M'}$	0	0	$e_{9,13,19}$	$e_{18,29,31}$	0	2^{-15}
26	e_{31}	e_{31}	e_{31}	$e_{M'}$	0	0	$e_{9,13,19}$	$e_{18,29,31}$	2^{-12}
27	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	0	$e_{9,13,19}$	2^{-7}
28	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	0	2^{-8}
29	0	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	2^{-7}
30	0	0	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	2^{-4}
31	0	0	0	0	0	e_{31}	e_{31}	e_{31}	1
32	0	0	0	0	0	0	e_{31}	e_{31}	2^{-1}
33	0	0	0	0	0	0	0	e_{31}	1
34	e_{31}	0	0	0	e_{31}	0	0	0	2^{-11}
35	$e_{6,9,18,20,25,29}$	e_{31}	0	0	$e_{6,20,25}$	e_{31}	0	0	

M'={6,9,18,20,25,29,31}

of 43-round SHACAL-2. The 256-bit value P is denoted by eight 32-bit words (A, B, C, D, E, F, G, H), and P^* is denoted by $(A^*, B^*, C^*, D^*, E^*, F^*, G^*, H^*)$. We denote the intermediate value just before round k by $Q^k_{i,j}$, and denote $Q^k_{i,j}$ by eight 32-bit words $A^k_{i,j}$, $B^k_{i,j}$, $C^k_{i,j}$, $D^k_{i,j}$, $E^k_{i,j}$, $F^k_{i,j}$, $G^k_{i,j}$ and $H^k_{i,j}$. Also, we denote $(\Delta A^{35}, \Delta B^{35}, \Delta C^{35}, \Delta D^{35}, \Delta E^{35}, \Delta F^{35}, \Delta G^{35}, \Delta H^{35})$ by Δ. The attack procedure for 43-round SHACAL-2 is performed as follows.

1. Choose $2^{175.38}$ structures S_i of plaintext $P_{i,j}$, $i = 1, 2, \ldots, 2^{175.38}$, $j = 1, 2, \ldots, 2^{64}$. XOR every 224 bits words (A, B, C, D, E, F, G) in S_i with the 224 bits value $(0, e_M, E_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31})$ (M={6,9,18,20,25,29}) and add 32-bit word H with 32-bit word $\Delta_{i,j}$ to get $2^{175.38}$ structures S^*_i, where in every structure the 192 bits words A, B, C, E, F, G are fixed, the 16 bits conditions presented in Table 2 are satisfied in every plaintext, and $g_1(E \oplus e_{9,13,19}) - g_1(E) + \Delta_{i,j} = 0$. Encrypt every plaintext in S_i and S^*_i using the key K and $K^* = K \oplus \Delta K$ to get the corresponding ciphertexts $C_{i,j}$ and $C^*_{i,j}$ respectively.
2. Guess two 96-bit subkeys (k^{42}, k^{41}, k^{40}) and $(k^{*42}, k^{*41}, k^{*40})$. For the guessed subkey pair, do the following:
   (a) Decrypt all the ciphertext $C_{i,j}$ and $C^*_{i,j}$ through rounds 42-40 using the subkey (k^{42}, k^{41}, k^{40}) and $(k^{*42}, k^{*41}, k^{*40})$ respectively to obtain the intermediate values $Q^{40}_{i,j}$ and $Q^{*40}_{i,j}$. We put all the intermediate values $Q^{40}_{i,j}$ in a table, and put $Q^{*40}_{i,j}$ in another table. We can get (A^{35}, B^{35}, C^{35}), $(A^*35, B^{*35}, C^{*35})$, $\Delta(D^{35}_{i_0,j_0}, D^{35}_{i_1,j_1})$ and $\Delta(D^{*35}_{i_0,j_0}, D^{*35}_{i_1,j_1})$ by observation 1.
   (b) Check whether $C^{40}_{i_0,j_0} \oplus C^{40}_{i_1,j_1}$ and $C^{*40}_{i_0,j_0} \oplus C^{*40}_{i_1,j_1}$ satisfy the first half of Δ. Record (k^{42}, k^{41}, k^{40}) and all the qualified quartets and then go to Step 3.
3. Guess two 32-bit subkeys k^{39}, k^{*39}, and decrypt all the remaining quartets $(Q^{40}_{i_0,j_0}, Q^{40}_{i_1,j_1}, Q^{*40}_{i_0,j_0}, Q^{*40}_{i_1,j_1})$ to obtain the actual values of $(A^{38}, B^{38}, C^{38}, D^{38}, E^{38}, F^{38}, G^{38})$, $(A^{*38}, B^{*38}, C^{*38}, D^{*38}, E^{*38}, F^{*38}, G^{*38})$, the additive

difference between $H^{38}_{i_0,j_0}$ and $H^{38}_{i_1,j_1}$, and the additive difference between $H^{*38}_{i_0,j_0}$ and $H^{*38}_{i_1,j_1}$, hence to get the actual values of $(A^{35}_{i_0,j_0}, B^{35}_{i_0,j_0}, C^{35}_{i_0,j_0})$, $(A^{35}_{i_1,j_1}, B^{35}_{i_1,j_1}, C^{35}_{i_1,j_1})$, $(A^{*35}_{i_0,j_0}, B^{*35}_{i_0,j_0}, C^{*35}_{i_0,j_0})$, $(A^{*35}_{i_1,j_1}, B^{*35}_{i_1,j_1}, C^{*35}_{i_1,j_1})$, the additive difference between $D^{35}_{i_0,j_0}$ and $D^{35}_{i_1,j_1}$, and the additive difference between $D^{*35}_{i_0,j_0}$ and $D^{*35}_{i_1,j_1}$ by observation 1. Since $H^{38} = E^{35}$ and $\Delta E^{35} = e_{6,20,25}$, we can discard all the quartets which do not satisfy $H^{38}_{i_1,j_1} - H^{38}_{i_0,j_0} \in \bigwedge_1$ and $H^{*38}_{i_1,j_1} - H^{*38}_{i_0,j_0} \in \bigwedge_1$, where $\bigwedge_1 = \{a + b + c | a = \pm 2^6, b = \pm 2^{20}, c = \pm 2^{25}\}$. Record $(k^{39}, k^{40}, k^{41}, k^{42})$ and all the qualified quartets and then go to Step 4.

4. Guess two 32-bit subkeys k^{38}, k^{*38}, and decrypt all the remaining quartets $(Q^{39}_{i_0,j_0}, Q^{39}_{i_1,j_1}, Q^{*39}_{i_0,j_0}, Q^{*39}_{i_1,j_1})$ to obtain the actual values of $(A^{37}, B^{37}, C^{37}, D^{37}, E^{37}, F^{37}, G^{37})$, $(A^{*37}, B^{*37}, C^{*37}, D^{*37}, E^{*37}, F^{*37}, G^{*37})$, the additive difference between $H^{37}_{i_0,j_0}$ and $H^{37}_{i_1,j_1}$, and the additive difference between $H^{*37}_{i_0,j_0}$ and $H^{*37}_{i_1,j_1}$. Since $H^{37} = F^{35}$ and $\Delta F^{35} = e_{31}$, we can discard all the quartets which do not satisfy $H^{37}_{i_1,j_1} - H^{37}_{i_0,j_0} \in \bigwedge_2$ and $H^{*37}_{i_1,j_1} - H^{*37}_{i_0,j_0} \in \bigwedge_2$, where $\bigwedge_2 = \{2^{31}, -2^{31}\}$. Record $(k^{38}, k^{39}, k^{40}, k^{41}, k^{42})$ and all the qualified quartets and then go to Step 5.
5. Guess two 32-bit subkeys k^{37}, k^{*37}, and decrypt all the remaining quartets $(Q^{38}_{i_0,j_0}, Q^{38}_{i_1,j_1}, Q^{*38}_{i_0,j_0}, Q^{*38}_{i_1,j_1})$ to obtain the actual values of $(A^{36}, B^{36}, C^{36}, D^{36}, E^{36}, F^{36}, G^{36})$, $(A^{*36}, B^{*36}, C^{*36}, D^{*36}, E^{*36}, F^{*36}, G^{*36})$, the additive difference between $H^{36}_{i_0,j_0}$ and $H^{36}_{i_1,j_1}$, and the additive difference between $H^{*36}_{i_0,j_0}$ and $H^{*36}_{i_1,j_1}$. Since $H^{36} = G^{35}$ and $\Delta G^{35} = 0$, we can discard all the quartets which do not satisfy $H^{36}_{i_1,j_1} = H^{36}_{i_0,j_0}$ and $H^{*36}_{i_1,j_1} = H^{*36}_{i_0,j_0}$. Record $(k^{37}, k^{38}, k^{39}, k^{40}, k^{41}, k^{42})$ and all the qualified quartets and then go to Step 6.
6. Guess two 32-bit subkeys k^{36}, k^{*36}, and decrypt all the remaining quartets $(Q^{37}_{i_0,j_0}, Q^{37}_{i_1,j_1}, Q^{*37}_{i_0,j_0}, Q^{*37}_{i_1,j_1})$ to obtain the actual values of $(A^{35}, B^{35}, C^{35}, D^{35}, E^{35}, F^{35}, G^{35})$, $(A^{*35}, B^{*35}, C^{*35}, D^{*35}, E^{*35}, F^{*35}, G^{*35})$, the additive difference between $H^{35}_{i_0,j_0}$ and $H^{35}_{i_1,j_1}$, and the additive difference between $H^{*35}_{i_0,j_0}$ and $H^{*35}_{i_1,j_1}$. Since $\Delta H^{35} = 0$, we can discard all the quartets which do not satisfy $H^{35}_{i_1,j_1} = H^{35}_{i_0,j_0}$ and $H^{*35}_{i_1,j_1} = H^{*35}_{i_0,j_0}$. If there exist more than 5 quartets passing this test, Record $(k^{36}, k^{37}, k^{38}, k^{39}, k^{40}, k^{41}, k^{42})$ and then go to Step 7. Otherwise, repeat Step 6 with another guessed subkeys. If all the possible key pairs in Step 6 are tested, then repeat Step 5 with another guessed subkeys. If all the possible key pairs in Step 5 are tested, then repeat Step 4 with another guessed subkeys. If all the possible key pairs in Step 4 are tested, then repeat Step 3 with another guessed subkeys. If all possible key pairs pairs in Step 3 are tested, then repeat Step 2 with another guessed subkeys.
7. For a suggested $(k^{36}, k^{37}, k^{38}, k^{39}, k^{40}, k^{41}, k^{42})$, exhaustively search for the remaining 288 key bits by trial encryption. If a 512-bit key is suggested, output it as the master key of 43-round SHACAL-2. Otherwise go to Step 2.

The data complexity of this attack is $2^{240.38}$ related-key chosen plaintexts. The memory requirements are about $2^{245.38} (= 2^{240.38} \times 32)$ memory bytes.

In Step 1, the time complexity is $2^{240.38}$ 43-round SHACAL-2 encryptions. The time complexity of Step 2 is about $2^{240.38} \times 2^{32\times 6} \times \frac{8}{43} \approx 2^{430}$ 43-round SHACAL-2 encryptions, and $2^{240.38} \times 2^{192} = 2^{432.38}$ memory access. For each guessed subkeys, we have $2^{239.38\times 2}/2 = 2^{477.76}$ quartets tested in Step 2. Since Sep 2 has a 256-bit filtering for the decrypted quartets, $2^{477.76} \times 2^{-256} = 2^{221.76}$ quartets are suggested in Step 2. The time complexity of Step 3 is about $2^{221.76} \times 2^{32\times 8} \times \frac{4}{43} \approx 2^{474.4}$ 43-round SHACAL-2 encryptions. Since there are 2^3 possible differences in $\bigwedge_1$, about $2^{221.76} \times (2^{-29})^2 = 2^{163.76}$ quartets are suggested in Step 3. The time complexity of Step 4 is about $2^{163.76} \times 2^{32\times 10} \times \frac{4}{43} \approx 2^{480.4}$ 43-round SHACAL-2 encryptions. Since there are 2 possible values in $\bigwedge_2$ (hence $\bigwedge_2$ has a 31-bit filterings), and $\Delta(H^{38})$ has a 3-bit filterings, about $2^{163.76} \times (2^{-31})^2 \times (2^{-3})^2 = 2^{95.76}$ quartets are suggested in Step 4. The time complexity of Step 5 is about $2^{95.76} \times 2^{32\times 12} \times \frac{4}{43} \approx 2^{476.4}$ 43-round SHACAL-2 encryptions. About $2^{95.76} \times (2^{-32})^2 \times = 2^{31.76}$ quartets are suggested in Step 5. The time complexity of Step 6 is about $2^{31.76} \times 2^{32\times 14} \times \frac{4}{43} \approx 2^{476.4}$ 43-round SHACAL-2 encryptions. About $2^{31.76} \times (2^{-32})^2 \times = 2^{-32.24}$ quartets are suggested in Step 6.

The expected number of right quartets are about $2^{477.76} \times 2^{-474.76} = 8$, for about $(2^{175.38}2^{64})^2/2 = 2^{477.76}$ quartets are tested in the attack and the 35-round related-key rectangle distinguisher holds with probability $2^{-474.76}$. Therefore the success rate of this attack (i.e. the probability that the number of remaining quartets for the right key pair is at least 6) is about 0.8 by the Poisson distribution $X \sim Poi(\lambda = 8)$, $Pr_X[X > 5] \approx 0.8$.

4 Conclusions

In this paper by using the related-key differential characteristics in [13], we fix some conditions (presented in Table 2) in each of the plaintexts, so that the differential of Step 0 will be hold with probability 1. Hence it will be not necessary to guess the subkey k^0 like in [13], which will reduce the time complexity. We can attack the 43-round SHACAL-2 using the related-key rectangle attack with data complexity of $2^{240.38}$ chosen plaintexts and time complexity of $2^{480.4}$ 43-round SHACAL-2 encryptions.

References

1. Helena Handschuh, David Naccache, SHACAL, preproceedings of NESSIE first workshop, Leuven, 2000.
2. H. Handschuh and D. Naccache, SHACAL: A Family of Block Ciphers, Submission to the NESSIE project, 2002.
3. E. Biham and A. Shamir, Differential Cryptanalysis of DES-like Cryptosystems, Proceedings of CRYPTO 1990, LNCS 537, pp. 2-21, Springer, 1990.
4. E. Biham, New Types of Cryptanalytic Attacks Using Related Keys, Proceedings of EUROCRYPT 1993, pp. 398-409, LNCS 765, 1993.
5. E. Biham, O. Dunkelman and N. Keller, Rectangle Attacks on 49-Round SHACAL-1, Proceedings of Fast Software Encryption 2003,LNCS2887, pp. 22-35, Springer, 2003.

6. E. Biham, Orr Dunkelman, Nathan Keller, Related-Key Boomerang and Rectangle Attacks, Advances in Cryptology, proceedings of EUROCRYPT'05, Lecture Notes in Computer Science 3494, pp. 507-525, Springer-Verlag, 2005.
7. J. Kim, D. Moon, W. Lee, S. Hong, S. Lee and S. Jung, Amplified Boomerang Attack against Reduced-Round SHACAL, Proceedings of ASIACRYPT 2002, LNCS 2501, pp. 243-253, Springer, 2002.
8. J. Kim, G. Kim, S. Hong, S. Lee and D. Hong, The Related-Key Rectangle Attack-Application to SHACAL-1, Proceedings of International Conference on Information Security and Privacy 2004, LNCS 3108, pp. 123-136, Springer, 2004.
9. Seokhie. Hong, Jongsung. Kim, Sangjin. Lee, Bart Preneel, Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192, Proceedings of Fast Software Encryption 12, Lecture Notes in Computer Science 3557, pp. 368-383, Springer-Verlag, 2005.
10. S. Hong, J. Kim, G. Kim, J.Sung, C. Lee and S. Lee, Impossible Differential Attack on 30-Round SHACAL-2, INDOCRYPT 2003, LNCS 2904, pp. 97-106, Springer-Verlag, 2003.
11. Y. Shin, J. Kim, G. Kim, S. Hong and S. Lee, Differential-Linear Type Attack on Reduced Rounds of SHACAL-2, ACISP 2004, Springer Berlin / Heidelberg ISSN: 0302-9743.
12. J. Kim, G. Kim, S. Lee, J. Lim and J. Song, Related-Key Attacks on Reduced Rounds of SHACAL-2, Proceedings of INDOCRYPT 2004.
13. J. Lu, J. Kim, N. Keller, and O. Dunkelman, Related-Key Rectangle Attack on 42-Round SHACAL-2, In Proceedings of the 9th Information Security Conference (ISC 2006), Lecture Notes in Computer Science, Springer-Verlag, 16 pages, 2006.
14. M. Blunden and A. Escott, Related Key Attacks on Reduced Round KASUMI, Proceedings of Fast Software Encryption 2001, LNCS 2355, pp. 277-285, Springer, 2002.
15. J. Kelsey, B. Schneier and D. Wagner, Key Schedule Cryptanalysis of IDEA, GDES, GOST, SAFER, and Triple-DES, Proceedings of CRYPTO 1996, LNCS 1109, pp. 237-251, Springer, 1996.
16. J. Kelsey, B. Schneir and D.Wagner, Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA, Proceedings of International Conference on Information and Communications Seucrity 1997, LNCS 1334, pp. 233- 246, Springer, 1997.
17. Y. Ko, S. Hong, W. Lee, S. Lee and J. Kang, Related Key Differential Attacks on 26 Rounds of XTEA and Full Rounds of GOST, Proceedings of Fast Software Encryption 2004, LNCS 3017, pp. 299-316, Springer, 2004.

On the Ability of AES S-Boxes to Secure Against Correlation Power Analysis

Zheng-lin Liu, Xu Guo, Yi-cheng Chen, Yu Han, and Xue-cheng Zou

Dept. of Electronic Sci. & Tech, Huazhong Univ. of Sci. & Tech., Wuhan, 430074, China
{lzl, xuguo, cyc, yuhan, xczou}@asian-micro.com

Abstract. Cryptographic substitution boxes (S-boxes) are an integral part of the Advanced Encryption Standard (AES). In this paper we conducted a simulation-based correlation power analysis (CPA) attack on AES implementations with different S-box structures. It shows that the abilities of AES and S-boxes to secure against CPA attack are correlated, and an evaluation of the ability of S-boxes to thwart CPA is presented in a quantitative way. By further exploiting the properties of S-boxes, an approximate linear relation between abilities of S-boxes to resist CPA and glitch power ratios of total power consumed by S-boxes is proved.

Keywords: correlation power analysis (CPA), Advanced Encryption Standard (AES), correlation coefficient, hamming distance.

1 Introduction

Advanced Encryption Standard is a new symmetric block cipher standard, which was issued by the National Institute of Standards and Technology (NIST) in 2001 [1]. There are two basic ways to protect the AES against the power analysis attack: The first one is to implement the AES based on special circuit architecture. For instance, Differential Cascade Voltage Switch Logic (DCVSL) [2], Wave Dynamic Digital Logic (WDDL) [3], and Random Switching Logic (RSL) [4] are announced as efficient countermeasures. Moreover, the asynchronous circuit implementation of AES was presented using self-timed dual-rail technology, which showed high security [5]. The alternative way is to randomize the intermediate results that occur during AES encryptions/decryptions. This masking of the intermediate results counteracts first-order DPA attacks and is usually used when the AES is implemented in software on a standard smart-card processor [6].

Although much research have been conducted to develop effective countermeasures against power analysis attacks in AES, few researchers emphasize the ability of each primitive AES component to resist the power analysis attack, especially to CPA, which has been proved to be more powerful than DPA in terms of efficiency, robustness and the number of experiments [7]. In this paper, the CPA attacks on a basic AES system with various S-boxes are conducted. And a comparison of correlation factors between different S-boxes is presented. In comparison with most of the S-box designs which

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 43 – 50, 2007.

merely consider cost metrics, our work focuses on exploiting the security properties of S-boxes. In the following section, the CPA method is introduced. In Section 3, a simulation-based CPA attack is conducted, and the results of attacking AES implementations with different S-boxes are described and compared. Section 4 addresses the correlation analysis on S-boxes. Finally, concluding remarks are made in Section 5.

2 Correlation Power Analysis Attacks

2.1 Theoretical Background

For AES the power analysis always attacks the first and the last round during the encryption. For an attacking time the CPA considers Hamming distance model [8]:

$$W = a \cdot H(D \oplus R) + b \tag{1}$$

where a is a scalar gain between the Hamming distance H and W the power consumed, D, R are the values in the previous state and current state respectively, and b is power dissipation induced by noise, offsets, and time dependent components.

When conducting CPA attack, we assume the reference state is a constant word, R. Here, R is set to be 0 (the initial state of register is 0). Then, we only consider partial key guess, and the corresponding partial plaintexts, PT_s. If PT_s contains n independent and uniformly distributed bits out of the total m bits, it has an average $\mu = n/2$ and a variance $\sigma^2 = m/4$. There still exists correlation between W and D:

$$\rho_{WD} = \frac{E(H(D)W) - E(H(D))E(W)}{\sigma(H(D))\sigma(W)} = \frac{a\sigma^2(H(D))}{\sigma(H(D))\sigma(W)} = \frac{a\sqrt{n}}{\sqrt{ma^2 + 4\sigma_b^2}} \tag{2}$$

According to the Equation (2), if the partial key guess is correct, the highest correlation coefficient can be achieved.

2.2 Power Analysis Method

A simulation-based analysis method was used to analyze the AES power. The netlist was acquired with the UMC 0.25 μm 1.8V CMOS technology. The circuit average power is computed using Prime Power. This approach allows designers to estimate the vulnerability to power analysis attacks in an early stage of the design flow.

We set up the CPA attack on the 8 most significant bits (MSBs) of the register in Fig. 1, and predicted the power consumption of bit-change during the process of storing these MSBs. We have chosen N random plaintexts and made one fixed but random key for the experiment. The simulator has calculated the total number of bit-changes between the previous and current values of these M MSBs of the register for the initial key addition. This result was stored in a file as an $N \times 1$ matrix (N=1000), *M0*, which contains values between 0 and M. Here, we have chosen M as 8. Then, we conducted CPA attacks

on the 8 MSBs of the register. We have measured the power consumption of AES during the first clock cycle of the encryption operation. The clock frequency applied to the system was 2.5 *MHz* and the sampling frequency was 1 *GHz*. And 400 samples were acquired. With these power values, we have produced an $N \times 400$ matrix, *M1*.

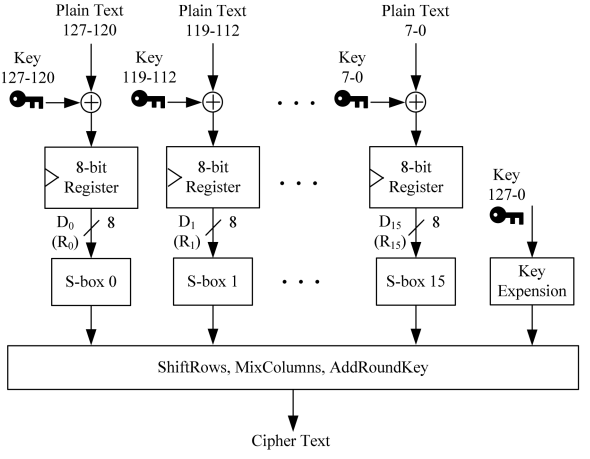


Fig. 1. Simplified block diagram of AES

3 CPA Attack on AES with Different S-Boxes

3.1 Simulation-Based Attacks

We chose two AES implementations with different S-box structures. One was implemented with combinatorial circuits using LUT (Look-Up Table), and the other was implemented with the multiplicative inverse in the composite field, GF(2^4) [9].

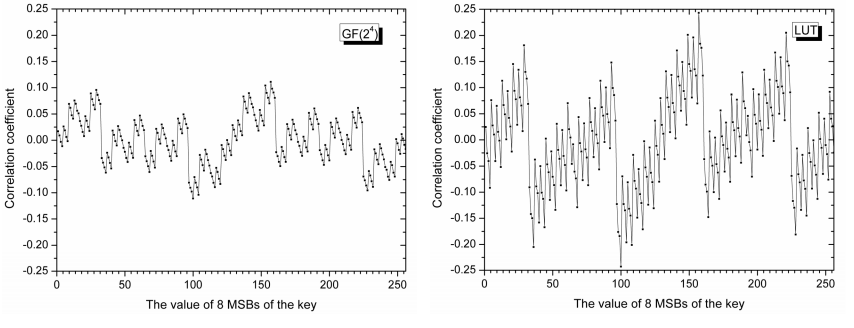


Fig. 2. The correlation coefficient between power and hamming distance of AES

The critical path delays of the two AES implementations are about 9 *ns* and 12 *ns*, respectively. Only the sample points of a measurement which directly reflect the bit-change in the attack operation are needed. We applied a pre-processing technique to reduce the amount of measured data and calculated the mean values of them.

As shown in Fig. 2, both of the results illustrate that the highest correlation occurs at 156. This value corresponds to *0x9C* which are the correct 8 MSBs of the key. The two figures show that the peak corresponding to the correct key becomes higher.

3.2 Analysis of Experimental Results

By using the CPA method we can successfully retrieve the partial secrete key of AES with two different S-boxes, but obvious differences between the two circumstances can be observed. We could find that the corresponding correlation coefficients of the peak points (denoted as C_p) in the two graphs are quite different. Moreover, the interval of correlation factors between the peak and the second highest point (denoted as $\triangle C$) also varies greatly. The C_p and $\triangle C$ have significant meanings while discussing the security properties of the AES.

Here we introduced the signal to noise ratio (SNR) of W to further interpret the results, and it is defined as:

$$SNR = \alpha\sigma_H / \sigma_b = \alpha\sqrt{m} / 2\sigma_b \tag{3}$$

Combining Equation (2), we can deduce the relation between correlation coefficient ρ_{WH} and SNR as follows:

$$\rho_{WH} = 1/\sqrt{1 + (1/SNR)^2} \tag{4}$$

From the above equations we can find that if the SNR of the AES system is decreased, the C_p would become smaller, and when the SNR is below a certain value, the C_p would drop down quickly. If the C_p is too small, it means the hamming distances and the power measurements are almost uncorrelated and the CPA attacks would show wrong key guesses. Generally, smaller C_p means it is more difficult for CPA attack to retrieve the key, and smaller $\triangle C$ means the correct key guess could be immerged by false key guesses affected by noise with a higher probability. Since the AES implementation with GF(2^4)-based S-box has both much smaller C_p and $\triangle C$, it is more secure than the one with LUT-based S-box in this CMOS technology. And we have conducted an experiment to approve the above discussion by adding the Gaussian white noise of mean 0 and variance 1.0e-6 to the measured power values.

4 Correlation Analysis on S-Boxes

The simulation results of the CPA attack show that the different abilities of AES to secure against CPA attack are determined by different S-boxes. Therefore we assume that there exists certain relation between AES and S-boxes in CPA resistant properties. According to the architecture of AES shown in Fig. 1 the total power consumption

contains register power, related combinatorial circuits power, KeyExpansion power and noise power, and it can be further divided into sixteen components related to sixteen S-boxes. Hence, P_{AES} can be defined as:

$$P_{AES} = a \cdot \sum_{i=0}^{15} H(D_i \oplus R_i) + \sum_{i=0}^{15} b_i + P_{key} + P_c \tag{5}$$

where D_i, R_i are the previous and current 8-bit register values directly related to S-boxes. The power consumed by MixColumns and the followed operations are relevant to four 8-bit register values, and the related power for each S-box is denoted as b_i. From Equation (5), it is clear that if we assume the total AES power is well adapted to the hamming distance model, we should firstly guarantee the model is also suitable for the power consumption of its components. Generally, the SubBytes operations consume much of the total power consumption in AES encryption operations [10]. Hence, we decided to exploit the CPA resistant properties of S-boxes.

There exists plenty of research devoted to the efficient design of cryptographic S-boxes, all of which can be attributed to three basic ways. The first one is to construct circuits directly from the truth-table of the S-box. Simply, a combinatorial LUT-based S-box (denoted as *LUT*) is used. The second method is to implement multiplicative inverse and affine transform with combinatorial circuits using direct relationship between input and output values of the S-box, such as SOP (Sum of Products) (denoted as *SOP*), and DSE (Decoder-Switch-Encoder) (denoted as *DSE*) [11]. The third approach is to implement the S-box with combinatorial logic using its arithmetic properties, such as an implementation of multiplicative inverse in the composite field GF(2^4) (denoted as *GF*) [9] and an power-efficient implementation in Galois field, PPRM (Positive Polarity Reed-Muller) (denoted as *PPRM*) [12].

We have implemented all solutions mentioned above, all of which just consists of combinatorial logic. The correlation coefficients of S-boxes are shown in Table 1.

Table 1. Comparison of correlation coefficients of various S-boxes

LUT	SOP	DSE	PPRM	GF
0.6266	0.6383	0.5894	0.4340	0.0111

Although *GF* S-box has the lowest correlation factor, the result is not sufficient to conclude that the lowest correlation between power trace and hamming distance in S-boxes would lead to the smallest C_p and $\triangle C$ in the AES. The power ratio of S-boxes in different AES implementations should also be considered. Therefore, a variable combined with the effects brought by the correlation coefficient and the power ratio of S-boxes is defined as:

$$f_c = P_{ratio} \cdot \rho_{sbox} \tag{6}$$

where P_{ratio} denotes the power ratio of S-boxes in the AES and ρ_{sbox} denotes the correlation coefficient between power traces and hamming distance of S-boxes. The f_c is more accurate to reflect the effect of S-boxes on the correlation coefficient of AES. The relation among C_p, $\triangle C$ and f_c is shown in Fig. 3.

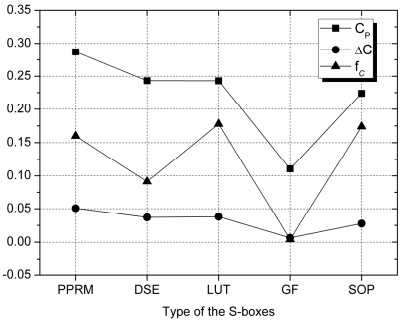


Fig. 3. The relation of C_p, $\triangle C$ and f_c

Fig. 3 illustrate that the big difference between correlation coefficients of AES implementations with LUT-based and GF(2^4)-based S-boxes is related to f_c. So far a relation between the AES system and S-boxes in terms of CPA resistant properties has been proved, and a comparison of abilities of different S-boxes to secure against CPA attack is shown.

Then we extended our research to find the causes of such differences among various S-boxes. The goal of hardware countermeasures against power analysis is to bury the attackable part of the power consumption in different kinds of noise [13]. We found that the glitches generated in S-boxes could be considered as a kind of noise, and we tried to find a relation between glitch power ratios of S-boxes, P_g, and C_s, the correlation coefficient between the total power and hamming distance of S-boxes.

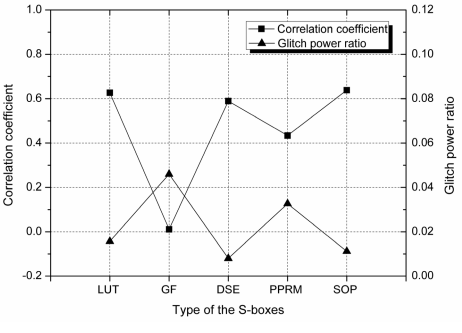


Fig. 4. The relation between C_s and P_g

From Fig. 4 it is obvious that there exists an approximate linear relation between the correlation coefficient and glitch power ratio of S-boxes, and the correlation factor between them is -0.9306. Therefore, the glitch power which can be considered as the main source of noise power in S-boxes greatly affects the ability of S-boxes to secure against CPA attack. According to Equation (4) more noise power would result in a lower *SNR* and further decrease the ρ_{WH}. This would partly explain the reason why the GF(2^4)-based S-box with the highest glitch power ratio can lead to the smallest C_p in the AES.

5 Conclusions and Future Work

In this paper we have examined the ability of various S-boxes to thwart CPA attack. Normally, hardware countermeasures lead to a significant increase of area and power consumption. Our research exploited the internal characteristics of S-boxes to resist CPA attack without any added logic. To our knowledge this is the first comprehensive study on the security aspects of standard cell implementations of AES S-boxes. According to the results of the simulated attacks the security levels of different S-boxes vary greatly, which can directly affect the ability of the AES to secure against CPA attack. Further, by establishing an approximate linear relation between glitch power ratios and correlation coefficients, we introduced some principles of how to build safer S-boxes. Our future work will focus on analyzing the CPA resistant properties of S-boxes from algorithmic views and designing a more secure AES system to resist CPA attack by utilizing the different security properties of S-boxes.

Acknowledgments

The research described in this paper has been supported by the High Technology Research and Development Program of China under grant 2006AA01Z226, and the Scientific Research Foundation of Huazhong Univ. of Sci. & Tech. under grant 2006Z011B.

References

1. “Advanced Encryption Standard (AES)”, Federal Information Processing Standards Publication 197, Nov. 2001.
2. K.Tiri, M.Akmal and I.Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. Proc. of 28th European Solid-State Circuits Conference, pp. 403-406, 2002.
3. K.Tiri, D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont, and I. Verbauwhede. A side-channel leakage free coprocessor IC in 0.18um CMOS for embedded AES-based cryptographic and biometric processing. Proc. ACM/IEEE Design Automation Conference (DAC 2005), pp. 222-227, 2005.
4. D. Suzuki, M. Saeki, and T. Ichikawa. Random Switching Logic: A Countermeasure against DPA based on Transition Probability. Cryptology ePrint Archive, Report 2004/346, 2004.

5. D. Shang, F. Burns, A. Bystrov, A. Koelmans, D. Sokolov, and A. Yakovlev. High-security asynchronous circuit implementation of AES. IEE Proceedings Computers and Digital Techniques, vol. 153, no. 2, pp. 71-77, 2006.
6. J. Golic and C. Tymen. Multiplicative Masking and Power Analysis of AES. CHES 2002, LNCS 2523, pp. 198-212, Springer-Verlag, 2003.
7. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In proceedings of CHES 2004, LNCS 3156, pp. 16-29, Springer, 2004.
8. S. Guilley, P. Hoogvorst, R. Pacalet. Differential Power Analysis Model and some Results. In proceedings of CARDIS 2004, Kluwer Academic Publishers, pp. 127-142, 2004.
9. J. Wolkerstorfer, E. Oswald, and M. Lamberger. An ASIC Implementation of the AES S-boxes. In CT- RSA 2002, LNCS 2271, pp. 67-78, 2002.
10. S. Morioka and A. Satoh. An optimized S-box circuit architecture for low power AES design. In CHES 2002, LNCS 2523, pp.172-186, 2003.
11. G. Bertoni, M. Macchetti, L. Negri, and P. Frangneto. Power-efficient ASIC Synthesis of Cryptographic Sboxes. In GLSVLSI 2004, pp. 277-281, ACM Press, 2004.
12. S. Morioka, and A. Satoh. An optimized S-box circuit architecture for low power AES design. In CHES 2002, LNCS 2523, pp.172-186, 2003.
13. S. Mangard. Hardware Countermeasures against DPA - A Statistical Analysis of Their Effectiveness. In CT-RSA 2004, LNCS 2964, pp. 222-235, 2004.

A Sanitizable Signature Scheme with Aggregation

Tetsuya Izu[1,2], Noboru Kunihiro[2], Kazuo Ohta[2],
Masahiko Takenaka[1], and Takashi Yoshioka[1]

[1] FUJITSU LABORATORIES Ltd.,
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
{izu,takenaka,yoshioka}@labs.fujitsu.com
[2] The University of Electro-Communications,
1-5-1, Chofugaoka, Chofu, 182-8585, Japan
{tizu,kunihiro,ota}@ice.uec.ac.jp

Abstract. A *sanitizable signature scheme* is a digital signature scheme in which, after generating a signer's signature on a document, specific entities (called *sanitizers*) can modify the document for hiding partial information. A verifier can confirm the integrity of disclosed parts of the sanitized document from the signature. The sanitizable signature is quite useful in governmental or military offices, where there is a dilemma between disclosure requirements of documents and privacy or diplomatic secrets. In this paper, we construct an efficient and provably secure sanitizable signature scheme with aggregation from bilinear maps, based on a sanitizable signature proposed by Izu et al, by applying the general aggregate signature by Boneh et al. We also propose some efficiency improvements on the proposed scheme by reducing the number of hash values required as verifiers' input.

1 Introduction

In governmental or military offices, there exists a bothersome dilemma between disclosure requirements of public documents maintained by these offices and privacy or diplomatic secrets recorded in these public documents. In old days, physical masking was a widely-used method for hiding such secrets. However, its analogy for digital documents are not established yet. In addition, in these days, digital documents are stored with digital signatures in order to assure the integrity of documents. Since current signature schemes can not distinguish such appropriate alternations on the original document (sanitizations) from inappropriate alternations (forgeries), a direct analogy of physical masking does not work well.

A *sanitizable signature scheme* is a possible solution for this problem in which, after generating a signer's signature on an original document, specific entities (called *sanitizers*) can modify the document for hiding partial information and generate sanitized documents. Then a verifier can confirm the integrity of disclosed parts of the sanitized document from the signature and

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 51–64, 2007.

the sanitized document. In addition, the secrecy of closed parts is assured, namely, no information of closed parts will be leaked after the sanitizations. Sanitized signatures are so attractive that many constructions have been proposed [SBZ01, JMSW02, MIM+05, ACMT05, IKTY05, MHI06, KL06]. In some schemes, sanitizers are not indetifiable for verifiers in order to keep sanitizers' privacy [SBZ01, MIM+05, MHI06]. However, in these schemes, adversaries can modify and generate forged sanitized documents easily (the additional sanitizing attack [MIM+05]). Ateniese et al. constructed a designated-sanitizer scheme to exclude adversaries' dishonest sanitizations, however, still the same attack can be applied [ACMT05]. On the other hand, Izu et al. proposed a sanitizable signature in which sanitizers are identifiable by verifiers, since all sanitizers use their secret-keys in sanitizations [IKTY05]. However, the scheme has a large overhead since verifiers require verification data linear to both the number of sanitizers and the number of subdocuments in the original document so that it is far from practical.

Contribution of This Paper

In this paper, we construct an efficient and provably secure sanitizable signature scheme with aggregation from bilinear maps, based on a recent sanitizable signature by Izu, Kanaya, Takenaka and Yoshioka (IKTY) [IKTY05], and the general aggregate signature from bilinear maps by Boneh, Gentry, Lynn and Shacham (BGLS) [BGLS03], a natural extension of the short signature by Boneh, Lynn, and Shacham (BLS) [BLS01]. We also provide security proofs of the proposed scheme in the general aggregate chosen-key security model [BGLS03] under co-GDH assumption in the random oracle model.

Proposed scheme has two fundamental functions as a sanitizable signature (the integrity of disclosed subdocuments, and the secrecy of closed subdocuments) and three additional functions (the identification of sanitizers, the identification of dishonest sanitizations, and the alternation of subdocuments). Because of this multi-functional property, verifiers require a large amount of data linear to both the number of sanitizers and the number of subdocuments. In some cases, these input may bring about a heavy overhead in verifications. In this paper, we also propose three efficiency improvements by reducing the number of hash values. As a drawback, some additional functions are lost from the improved schemes. However, the improved schemes have the fundamental functions as a sanitizable signature. Note that our improvements can be applied to the original sanitizable signature proposed by Izu et al.

The rest of this paper is as follows: after an introduction of notations, the general aggregate signature scheme (BGLS scheme), and its security model in section 2, we construct a sanitizable signature scheme with aggregation in section 3. Security discussions are in the same section. Then we propose some efficiency improvements on the proposed scheme in section 4.

2 Preliminaries

This section introduces some notations and the general aggregate signature scheme (BGLS scheme) together with its security model.

2.1 Notations

In this paper, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are multiplicative cyclic groups with order p (prime) and g_1, g_2 are generators of $\mathbb{G}_1$, $\mathbb{G}_2$ (namely, $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$). We assume that Computational Diffie-Hellman (CDH) problem in these groups are hard. Let e be a (cryptographic) bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ such that $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in \mathbb{G}_1$, $c \in \mathbb{G}_2$, $a, b \in \mathbb{Z}$ (bilinearity) and $e(g_1, g_2) \neq 1$ (non-degeneracy).

We also use two secure hash functions $H_1 : \{0,1\}^* \to \{0,1\}^\ell$ and $H_2 : \{0,1\}^* \to \mathbb{G}_2$. H_1 is a standard hash function and we assume that a certain value of ℓ is provided implicitly in the following. For a construction of H_2 in the random oracle model, see [BLS01].

2.2 Aggregate Signature

The aggregate signature scheme is a digital signature scheme in which m signatures generated by m signers on distinct m documents are compressed in an aggregate signature whose length is (almost) same as that of each signature. A verifiers can confirm the integrity of m signatures from the aggregate signature, m documents and m public-keys. If the aggregate signature is invalid, the verifier cannot identify which signatures were invalid.

A concept of the aggregate signature is introduced by Boneh, Gentry, Lynn, and Shacham [BGLS03]. They also constructed a scheme from bilinear maps in the same paper. Since a generation of signers' signatures and an aggregation of signatures are proceeded in separate algorithms, their scheme is called the *general aggregate signature*. On the other hand, Lysyanskaya, Micali, Reyzin, and Shacham constructed another scheme from trap-door permutations [LMRS04]. Since a generation of a signer's signature and an aggregation is proceeded in the same algorithm, their scheme is called the *sequential aggregate signature*. One of the distinguishing property between the general and the sequential scheme is that, when an aggregate signature is valid, sequential verifiers can obtain aggregate signatures output by signers while general verifiers can not.

Note that general aggregate signatures can be used as sequential aggregate signatures. In fact, in our proposed scheme, a general aggregate signature is used as a sequential aggregate signature.

2.3 General Aggregate Signature from Bilinear Maps

We briefly review the general aggregate signature scheme from bilinear maps by Boneh, Gentry, Lynn, and Shacham (BGLS scheme) [BGLS03], a natural extension of the short signature by Boneh Lynn, and Shacham [BLS01]. BGLS scheme consists of four algorithms KeyGen, Sign, Agg, and AggVerify. Fig. 1 shows a description of the BGLS scheme (the aggregate signature scheme), where m signers are assumed. Note that documents $M_1, \ldots, M_m$ should be distinct in order to exclude the potential attack [BGLS03].

In the aggregation algorithm Agg, an input aggregate signature σ must be independent from $\sigma_{j_1}, \ldots, \sigma_{j_k}$. In our proposed scheme, an aggregate signature

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

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

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

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

Fig. 1. A description of BGLS signature scheme

$\sigma^{(j)}$ is aggregated from a signature σ_j generated by the j-th sanitizer and an aggregate signature $\sigma^{(j-1)}$ output by the $(j-1)$-th sanitizer. Here $\sigma^{(j)}$ always equals to a product of $\sigma_1, \ldots, \sigma_j$.

Security of BGLS Scheme

Before discussing the security of the BGLS scheme, we define some related notions. A co-CDH problem is a problem to compute $h^a \in \mathbb{G}_2$ from given g_1, $g_1^a \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, while a co-DDH problem is a decision problem to determine whether $a = b$ or not from given g_1, $g_1^a \in \mathbb{G}_1$ and h, $h^b \in \mathbb{G}_2$ [1]. A group pair $(\mathbb{G}_1, \mathbb{G}_2)$ is called a co-GDH pair if co-CDH problem is hard, but co-DDH problem is easy. A co-GDH assumption is an assumption that $(\mathbb{G}_1, \mathbb{G}_2)$ is a co-GDH pair. In fact, in the BGLS scheme, since a bilinear map e is defined over $\mathbb{G}_1 \times \mathbb{G}_2$, co-DDH problem is easily solved.

Next, let us consider the following game where an adversary $\mathcal{A}$ attempts to forge an aggregate signature in the BGLS scheme: in the setup, $\mathcal{A}$ receives a randomly-generated public key $\mathsf{pk}_1 \in \mathbb{G}_1$. Then, $\mathcal{A}$ requests signatures with pk_1 on adaptively-chosen messages. Finally, $\mathcal{A}$ outputs $m-1$ additional public-keys $\mathsf{pk}_2, \ldots, \mathsf{pk}_m$, messages $M_1, \ldots, M_m$, and an aggregate signature σ. If σ is valid over $M_1, \ldots, M_m, \mathsf{pk}_1, \ldots, \mathsf{pk}_m$, and M_1 is not trivial (namely, $\mathcal{A}$ did not request a signature on M_1 with pk_1), we consider that the adversary wins the game (the general aggregate chosen-key model [BGLS03]).

[1] These problems are generalizations of standard CDH, DDH problems.

It is proved that the BGLS scheme is secure in the general aggregate chosen-key model under co-GDH assumption, namely $\mathcal{A}$'s advantage over coin tosses is negligible. For further security discussions, see [BGLS03].

3 Sanitizable Signature Scheme with Aggregation

In this section, we construct an efficient sanitizable signature scheme with aggregation and provide security proofs in the general aggregate chosen-key model.

As in the previous sanitizable signatures, three types of entities *signers*, *sanitizers* and *verifiers* are considered [2]. A signer generates a signature on an original document. From the original or sanitized document, sanitizers modify the document and generate a new sanitized document. A verifier confirms the integrity of the sanitized subdocuments from the signature and the sanitized document.

3.1 Proposed Sanitizable Signature Scheme with Aggregation

We assume that a document M is identified with an ordered sequence of subdocuments of length n, i.e. $M = (M_1, \ldots, M_n)$ where $M_i \in \{0,1\}^*$. For example, an XML document has such structure. Proposed sanitizable signature scheme consists of four algorithms KeyGen, Sign, Sanitize and Verify. Each algorithm proceeds as in Fig. 2, where m sanitizers are considered. We identify a signer as the 0-th sanitizer. An example with $m = 2$ and $n = 5$ is shown in Fig 3, where verifiers require boxed information as input.

In the proposed scheme, a signer firstly pads random strings to subdocuments M_i as subdocument ID to assure the indistinguishability (and thus secrecy) between subdocuments. Then she computes hash information $h^{(0)}$, a concatenation of a random string $h_0^{(0)}$ and hash values of padded subdocuments $h_1^{(0)}, \ldots, h_n^{(0)}$. Here $h_0^{(0)}$ is padded to assure the distinctness of all hash information. Without this padding, the general aggregate signature may fail when a sanitizer does not sanitize at all (because of the potential attack [BGLS03]).

On input a padded message $\bar{M}^{(0)}$, hash information $h^{(0)}$ and a signature $\sigma^{(0)}$, the first sanitizer determines which subdocuments to be sanitized ($\bar{M}_3$ and $\bar{M}_5$, for example). Then, he generates a new sanitized document $\bar{M}^{(1)} = (\bar{M}_1^{(1)}, \ldots, \bar{M}_n^{(1)})$ by setting $\bar{M}_i^{(1)} \leftarrow \bar{M}_i^{(0)}$ for $i \neq 3, 5$ and $\bar{M}_3^{(1)} \leftarrow H_1(\bar{M}_3^{(0)})$, $\bar{M}_5^{(1)} \leftarrow H_1(\bar{M}_5^{(0)})$. Finally, he generate a signature on value information $h^{(1)}$ generated from a random string $h_0^{(1)}$ and hash values of subdocuments $h_1^{(1)}, \ldots, h_n^{(1)}$ and aggregate it with the signature $\sigma^{(0)}$. The following sanitizers repeat similar procedures. Note that sanitizers can identify which subdocuments were sanitized by comparing the original and the last hash information.

After a verifier verifies the integrity of the sanitized document $\bar{M}^{(m)}$ by comparing $H_1(\bar{M}_i^{(m)})$ and $h_i^{(m)}$, he verifies the aggregate signature $\sigma^{(m)}$. If the verification is well and the proposed scheme is used just as a sanitizable signature,

[2] Signers are called *owners* in [SBZ01].

KeyGen (of the j-th sanitizer)

1. Generate $\mathsf{sk}_j \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$ randomly and set $\mathsf{pk}_j \leftarrow g_2^{\mathsf{sk}}$.

Output: A secret and public key pair $(\mathsf{sk}_j, \mathsf{pk}_j) \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{G}_2$

Sign

Input: A document $(M_1, \ldots, M_n)$ and a signer's secret-key $\mathsf{sk}_0 \in \mathbb{Z}/p\mathbb{Z}$

1. Generate a random value r_i and set $\bar{M}_i^{(0)} \leftarrow r_i || M_i$ for each subdocument M_i $(i = 1, \ldots, n)$.
2. Compute a hash value $h_i^{(0)} \leftarrow H_1(\bar{M}_i^{(0)})$ for each padded subdocument $\bar{M}_i$ $(i = 1, \ldots, n)$.
3. Generate a signature $\sigma^{(0)} \leftarrow H_2(h^{(0)})^{\mathsf{sk}_0}$, where $h^{(0)} = h_0^{(0)} || \ldots || h_n^{(0)}$ with a random value $h_0^{(0)}$.

Output: A document $(\bar{M}_1^{(0)}, \ldots, \bar{M}_n^{(0)})$, hash values $h^{(0)} \in \{0,1\}^\ell$ and a signature $\sigma^{(0)} \in \mathbb{G}_1$

Sanitize (by the j-th sanitizer)

Input: A document $(\bar{M}_1^{(j-1)}, \ldots, \bar{M}_n^{(j-1)})$, hash values $h^{(0)}, \ldots, h^{(j-1)} \in \{0,1\}^\ell$, a signature $\sigma^{(j-1)} \in \mathbb{G}_1$ and a secret key $\mathsf{sk}_j \in \mathbb{Z}/p\mathbb{Z}$

1. Compute an index set (of sanitized subdocuments) $S \leftarrow \{i | h_i^{(0)} \neq h_i^{(j-1)}\}$.
2. Determine a new index set $S' \supseteq S$.
3. Generate a new subdocument $\bar{M}_i^{(j)}$ for each subdocument $M_i^{(j-1)}$ $(i = 1, \ldots, n)$ by setting

$$\bar{M}_i^{(j)} \leftarrow \begin{cases} H_1(\bar{M}_i^{(j-1)}) & \text{if } i \in S' \backslash S \\ \bar{M}_i^{(j-1)} & \text{otherwise.} \end{cases}$$

4. Compute a hash value $h_i^{(j)} \leftarrow H_1(\bar{M}_i^{(j)})$ for each subdocument $\bar{M}_i^{(j)}$ $(i = 1, \ldots, n)$.
5. Generate a signature $\sigma^{(j)} \leftarrow \sigma^{(j-1)} \cdot H_2(h^{(j)})^{\mathsf{sk}_j}$, where $h^{(j)} = h_0^{(j)} || \ldots || h_n^{(j)}$ with a random value $h_0^{(j)}$.

Output: A document $(\bar{M}_1^{(j)}, \ldots, \bar{M}_n^{(j)})$, hash values $h^{(0)}, \ldots, h^{(j)} \in \{0,1\}^\ell$ and a signature $\sigma^{(j)} \in \mathbb{G}_1$

Verify

Input: A document $(\bar{M}_1^{(m)}, \ldots, \bar{M}_n^{(m)})$, hash values $h^{(0)}, \ldots, h^{(m)} \in \{0,1\}^\ell$, a signatures $\sigma^{(m)} \in \mathbb{G}_1$ and public-keys $\mathsf{pk}_0, \ldots, \mathsf{pk}_m \in \mathbb{G}_2$

1. Check whether $h^{(m)} = H_1(h_1^{(m)}) || \ldots || H_1(h_n^{(m)})$. If not, output invalid and terminate.
2. Check whether $e(\sigma^{(m)}, g_2) \neq \prod_{j=0}^{m} e(H_2(h^{(j)}), \mathsf{pk}_j)$. If not, output invalid and terminate.
3. Compute an index set (of sanitized subdocuments) $S \leftarrow \{i | h_i^{(0)} \neq h_i^{(m)}\}$.
4. For a sanitized subdocument $\bar{M}_i^{(m)}$ $(i \in S)$, identify the corresponding sanitizer. Here, if there exists a unique j such that $h_i^{(0)} = \cdots = h_i^{(j-1)} \neq h_i^{(j)} = \cdots = h_i^{(m)}$, we treat the j-th sanitizer as the sanitizer. If such j exists for all sanitized subdocuments, output valid and terminate. If not, output invalid and terminate.

Fig. 2. A description of the proposed sanitizable signature scheme with aggregation

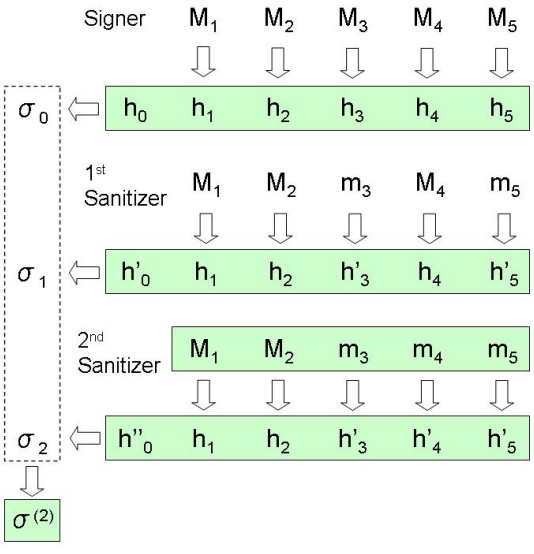


Fig. 3. Proposed scheme

he can terminate here. If further information are required, he can continue and identify who sanitized which subdocuments in sanitizations.

3.2 Security of the Proposed Scheme

Let us discuss the security of the proposed scheme in the general aggregate chosen-key model [BGLS03]. In order to prove the security of a sanitizable signature scheme, the integrity of disclosed subdocuments, and the secrecy of closed subdocuments should be established. In fact, the following theorem holds for the proposed scheme.

Theorem 1 (Security of the Proposed Scheme)

1. *The proposed scheme has the integrity of disclosed subdocuments under* co-GDH *assumption, namely no one can forge a valid aggregate signature and corresponding messages.*
2. *The proposed scheme has the secrecy, namely no information of original subdocuments are leaked from sanitized subdocuments.*

Proof (Sketch) 1. Because of the unforgeability of the BGLS general aggregate signature, adversaries (except sanitizers) can not forge at all. Thus dishonest sanitizers should be considered. Observe that if an adversary $\mathcal{A}$ looses the game in the general aggregate chosen-key model, it implies that $\mathcal{A}$ can not extract a signature corresponding to pk_1 even if an aggregate signature (aggregated from a signature corresponding to pk_1 and other signatures) is given after the setup. Unless, he can generate a new aggregate signature by using the extracted signature and win the game. Consequently, dishonest sanitizers can not forge.

2. Since sanitized documents are replaced by their hash values, no information of original subdocuments will be leaked, if used hash function is one-way.

Remark 1. However, the first sanitizer can forge an aggregate signature since he receives a signer's signature rather than an aggregate signature. For excluding such attacks, it may be better for the signer to double as the first sanitizer.

3.3 Comparison with the IKTY Scheme

This section briefly describes the security difference between the proposed scheme and IKTY scheme. The major algorithmic difference is that, in IKTY scheme, the j-th sanitizer outputs his own signature $\sigma_j = H_1(\bar{M}^{(j)})^{\mathsf{sk}_j}$ rather than an aggregate signature $\sigma^{(j)}$ aggregated from $\sigma_0, \ldots, \sigma_j$. (Moreover, the j-th sanitizer can use arbitrary signature schemes.) Thus the $(j+1)$-th sanitizer receives a sanitized document, $j+1$ hash information $h^{(0)}, \ldots, h^{(j)}$ and $j+1$ signatures $\sigma_0, \ldots, \sigma_j$. Here sanitized information can be changed by following two attacks.

Deletion-of-Intermediate-Sanitizer Attack is a kind of the man-in-the-middle attack. IKTY scheme is vulnerable to this attack: After obtaining the j-th sanitizer's output, an adversary arbitrary delete a hash information and a signature pair $(h^{(a)}, \sigma_a)$ $(0 \leq a < j)$ and passes the forged data to the $(j+1)$-th sanitizer. Then a verifier can not detect such a deletion, since no format error is occurred. Worse yet, this verifier confirms that closed subdocuments, originally sanitized by the a-th sanitizer, are sanitized by the $(a+1)$-th sanitizer, since a-th sanitized information is missing. On the other hand, this attack cannot be applied to the proposed scheme: no one can extract the a-th sanitizer's signature from the aggregate signature.

Deletion-of-Last-Sanitizer Attack is done by dishonest sanitizers. IKTY scheme is vulnerable to this attack: suppose the $(j+1)$-th sanitizer is an adversary. Suppose he deletes a hash information and a signature pair $(h^{(j)}, \sigma_j)$ and generate a new pair $(h^{(j+1)}, \sigma_{j+1})$ on the sanitized document $\bar{M}^{(j)}$ and output them. Then a verifier can not detect such a deletion, since no format error is occurred. Worse yet, this verifier confirms that closed subdocuments, originally sanitized by the j-th sanitizer, are sanitized by the $(j+1)$-th sanitizer, since j-th sanitized information is missing. This attack cannot be applied to the proposed scheme; no one can extract the j-th sanitizer's signature from an aggregate signature.

Remark 2. When IKTY scheme is combined to the sequential aggregate signature, it resists to the deletion-of-intermediate-sanitizer attack. However, it is vulnerable to the deletion-of-last-sanitizer attack since the $(j+1)$-th sanitizer can obtain the aggregate signature generated by $(j-1)$-th sanitizer.

3.4 Functions of the Proposed Scheme

Proposed scheme has two fundamental functions and three additional functions as in the followings.

The next two functions called fundamental functions since they are required as a sanitizable signature.

- **Integrity of disclosed subdocuments:** Because of the unforgeability shown in section 3.2, the proposed scheme assures the integrity of disclosed subdocuments.
- **Secrecy of closed subdocuments:** Because of the secrecy shown in section 3.2, the proposed scheme assures the secrecy of closed subdocuments.

The next three functions called additional functions since they are not required as a sanitizable signature.

- **Identification of sanitizers:** Since each sanitizer generates a signature by using own-secret key, a verifier can identify who sanitized which subdocuments. However, identification of dishonest sanitizer is not possible. For example, if the 2nd sanitizer replaces a hash $h_i^{(1)}$ to other value, the aggregate signature is no more valid. Because of a property of the aggregate signature verification, a verifier can not identify who is a dishonest sanitizer. Note that identification of a dishonest sanitizers is possible in the sanitizable signature by Izu et al. [IKTY05].
- **Identification of dishonest sanitization:** By comparing hash information, a verifier can identify which subdocuments were dishonestly sanitized. In the above example, a verifier can identify the dishonest sanitization on the i-th subdocument, since there does not exist j such that $h_i^{(0)} = \cdots = h_i^{(j-1)} \neq h_i^{(j)} = \cdots = h_i^{(m)}$.
- **Alternation of subdocuments:** When a subdocument $\bar{M}_i$ is sanitized in the proposed scheme, it is replaced by its hash value $H_1(\bar{M})$. Because all sanitizers publishes their signatures, the sanitization can be identified by a verifier. This masking can be considered as a special case of modification: a sanitizer can modify a subdocument $\bar{M}_i$ into any data which a verifier can identify. Thus our scheme can alternate the contents of subdocument.

4 Efficiency Improvements

In this section, we discuss how to improve the proposed sanitizable signature scheme. In the proposed scheme with m sanitizers and n subdocuments, a verifier requires a large amount of input: n subdocuments $\bar{M}^{(m)} = (\bar{M}_1^{(m)}, \ldots, \bar{M}_n^{(m)})$ as a sanitized document, $(m+1)(n+1)$ hash values $h_0^{(0)}, \ldots, h_n^{(0)}, h_0^{(1)}, \ldots, h_n^{(1)}, \ldots, h_0^{(m)}, \ldots, h_n^{(m)}$ as hash information, $(m+1)$ public-keys $\mathsf{pk}_0, \ldots, \mathsf{pk}_m$ and an aggregate signature $\sigma^{(m)}$. In some cases, these input may bring about a heavy overhead in the verification. In the followings, we propose some efficiency improvements by reducing the number of hash values. As a drawback, some additional functions are lost from the scheme. However, the improved schemes have the fundamental functions as sanitizable signatures. Note that the sanitizable signature scheme by Izu et al. [IKTY05] can be improved in the same way.

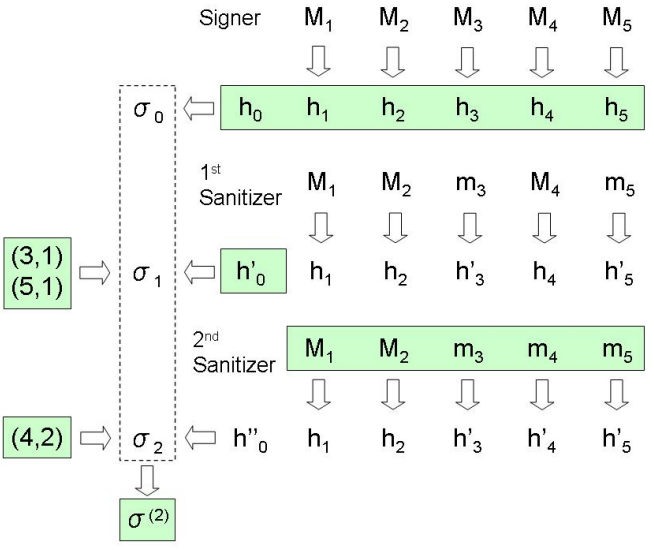


Fig. 4. Improvement 1

4.1 Improvement 1: SCCS-Type Management

In the verification of the proposed scheme, verifiers require $(m+1)(n+1)$ hash values $h_0^{(0)}, \ldots, h_n^{(0)}, h_0^{(1)}, \ldots, h_n^{(1)}, \ldots, h_0^{(m)}, \ldots, h_n^{(m)}$. One can observe that most of these values are same: in fact, we have $h_i^{(0)} = \cdots = h_i^{(m)}$ for each $i = 1, \ldots, n$ if corresponding subdocument is disclosed, and $h_i^{(0)} = \cdots = h_i^{(j-1)} \neq h_i^{(j)} = \cdots = h_i^{(m)}$ if closed (on the other hand, $h_0^{(0)}, \ldots, h_0^{(m)}$ are all distinct since they are ID numbers). If the signer outputs hash values $h_0^{(0)}, \ldots, h_n^{(0)}$, following sanitizers are required only to output hash values $h_0^{(1)}, \ldots, h_0^{(m)}$ since other hash values are recovered from $h^{(0)}$ and $h^{(m)}$ computed from the sanitized document $\bar{M}^{(m)}$. In order to record a sanitization, we use a notation (i, j) which describes that the i-th subdocument is sanitized by the j-th sanitizer. Then output of the j-th sanitizer are a hash value $h_0^{(j)}$, a set of sanitization information $S_j = \{(i_1, j), \ldots, (i_s, j)\}$, an aggregate signature $\sigma^{(j)}$ generated from a previous aggregate signature $\sigma^{(j-1)}$ and a signature on $h_0^{(j)}$ and S_j, and a sanitized document $\bar{M}^{(j)}$. Fig. 4 shows an example with $m = 2$ and $n = 5$. Here verifiers require boxed information as input. Since all hash values can be recovered, no additional functions are lost. With this improvement, required hash values are reduced to $m + n + 1$ from $(m + 1)(n + 1)$. Note that this improvement uses the original hash values as a base point. This idea is based on the source code management system Source Code Control System (SCCS) widely used in UNIX world.

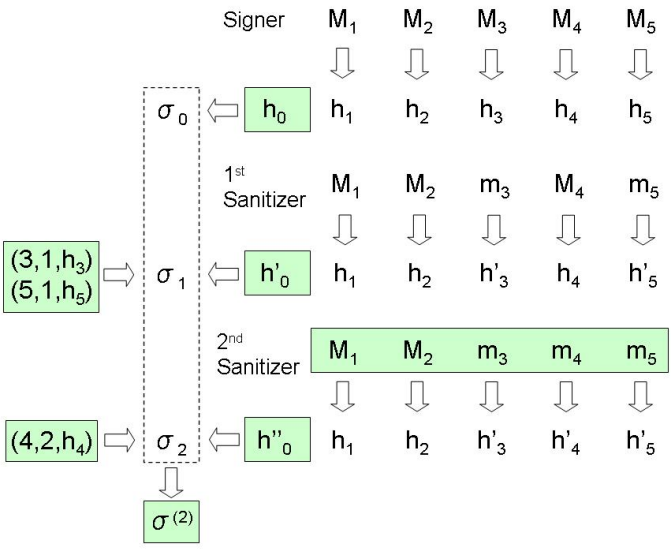


Fig. 5. Improvement 2

4.2 Improvement 2: RCS-Type Management

If a subdocument $\bar{M}_i$ is disclosed (unchanged) in the improvement 1, corresponding hash value is obtained from both $h_i^{(0)}$ and $H_1(\bar{M}_i^{(m)})$ which implies a redundancy. Since the sanitized document is an indispensable information for verifiers, it is desirable to use corresponding hash values as a base point. Similar idea is used in the source code management system RCS (Revision Control System). In this improvement, we use a notation (i, j, h_i) which describes that the i-th subdocument is sanitized by the j-th sanitizer from a subdocument whose corresponding hash value was h_i. Then, output of the j-th sanitizer are a hash value $h_0^{(j)}$, a set of sanitization information $S_j = \{(i_1, j, h_{i_1}), \ldots, (i_k, j, h_{i_k})\}$, an aggregate signature $\sigma^{(j)}$ generated from a previous aggregate signature $\sigma^{(j-1)}$ and a signature on $h_0^{(j)}$ and S_j, and a sanitized document $\bar{M}^{(j)}$. Fig. 5 shows an example. With this improvement, required hash values are reduced to $m+n_S+1$, where n_S is the number of sanitized documents which is at most n. However, dishonest sanitizations can not be identified with this improvement, since original hash values are not stored anywhere.

4.3 Improvement 3: RCS-Type Management with Embedding

In the improvement 2, verifiers require n_C hash values other than the sanitized document as input. Remember in the proposed scheme, a subdocument $\bar{M}_i$ is sanitized by replacing it to an arbitrary value $\bar{M}$. This improvement embeds the hash value into the subdocument: $\bar{M}_i \leftarrow H_1(\bar{M}_i)$. Then, output of the j-th sanitizer are a hash value $h_0^{(j)}$, a set of sanitization information

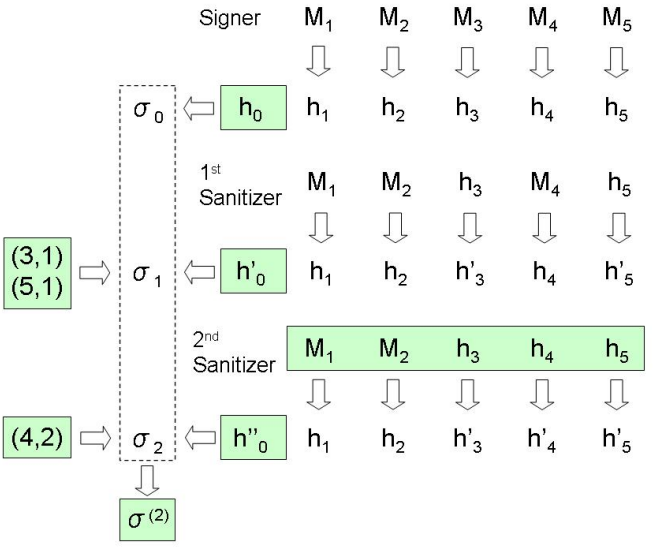


Fig. 6. Improvement 3

$S_j = \{(i_1, j), \ldots, (i_k, j)\}$, an aggregate signature $\sigma^{(j)}$ generated from a previous aggregate signature $\sigma^{(j-1)}$ and a signature on $h_0^{(j)}$ and S_j, and a sanitized document $\bar{M}^{(j)}$. Fig. 6 shows an example. With this improvement, required hash values can be reduced to $m+1$. However, in addition to that dishonest sanitizations can not be identified similar to the improvement 2, subdocuments are not altered freely, since a previous hash value is stored as a sanitized subdocument.

Strongly note that in this improvement, the secrecy of closed subdocument is established because of the preimage-resistance of a secure hash function H_2.

4.4 Comparison

The efficiency and the functions of proposed and improved schemes are summarized in Table 1. Here m, n, n_C denote the number of sanitizers, the number of subdocuments, and the number of sanitized subdocument which is at most n.

From a viewpoint of efficiency, the improvements 1, 2, 3 reduces the number of hash values. Especially, the improvement 3 reduces the number of hash values from $(m+1)(n+1)$ required in the proposed scheme to only $m+1$. On the other hand, while all schemes have fundamental functions as a sanitizable signature (integrity and secrecy), some additional functions are lost in the improvements 2, 3 as drawbacks.

For another comparison, Table 2 summaries the efficiency and the functions of the sanitizable signature by Izu et al. (IKTY) and improved schemes. Since IKTY does not use an aggregate signature, verifiers require n signature as input. However, dishonest sanitizers can be identified by verifying each signature. Other efficiency and functions are similar to Table 1. Note that as in section 3.2, these

Table 1. Comparison of proposed and improved schemes

	Proposed	Improvement 1	Improvement 2	Improvement 3
# of subdocuments	n	n	n	n
# of public keys	$m+1$	$m+1$	$m+1$	$m+1$
# of hash values	$(m+1)(n+1)$	$m+n+1$	$m+n_{\mathrm{C}}+1$ $\leq m+n+1$	$m+1$
# of signatures	1	1	1	1
Integrity of disclosed subdocuments	OK	OK	OK	OK
Secrecy of closed subdocuments	OK	OK	OK	OK
Sanitizer identification	OK	OK	OK	OK
Subdocument alternation	OK	OK	OK	-
Dishonest sanitation identification	OK	OK	-	-
Dishonest sanitizer identification	-	-	-	-

Table 2. Comparison of IKTY and improved schemes

	IKTY	Improvement 1	Improvement 2	Improvement 3
# of subdocuments	n	n	n	n
# of public keys	$m+1$	$m+1$	$m+1$	$m+1$
# of hash values	$(m+1)(n+1)$	$m+n+1$	$m+n_{\mathrm{C}}+1$ $\leq m+n+1$	$m+1$
# of signatures	n	n	n	n
Integrity of disclosed subdocuments	OK	OK	OK	OK
Secrecy of closed subdocuments	OK	OK	OK	OK
Sanitizer identification	OK	OK	OK	OK
Subdocument alternation	OK	OK	OK	-
Dishonest sanitation identification	OK	OK	-	-
Dishonest sanitizer identification	OK	OK	OK	OK

schemes are insecure in the sense that they are vulnerable to the deletion-of-intermediate-sanitizer attack and the deletion-of-last-sanitizer attack.

5 Concluding Remarks

This paper constructs an efficient and provably secure sanitizable signature scheme with aggregation, based on IKTY sanitizable signature and BGLS general aggregate signature from bilinear maps. We also provide security proofs of the proposed scheme. In addition, some improvements are proposed for efficiency.

The deletion-of-last-sanitizer attack is a very powerful attack since an adversary generates a signature as a sanitizer. A main reason why the proposed scheme resists this attack is that the adversary cannot extract a target sanitizer's signature from an aggregate signature. If we assume the power to extract such target sanitizer's signature, the proposed scheme is also vulnerable. This assumption is not so unreasonable: the adversary only needs to obtain an aggregate signature input to the target sanitizer and an aggregate signature output from him. It seems hard to resist such attack. However, further discussions of countermeasures together with the propriety of the assumption.

References

[ACMT05] G. Ateniese, D.H. Chou, B. Medeiros and G. Tsudik, "Sanitizable Signatures", *ESORICS 2005*, LNCS 3679, pp. 159-177, Springer-Verlag, 2005.

[BLS01] D. Boneh, B. Lynn and H. Shacham, "Short Signature from the Weil Pairing", *ASIACRYPT 2001*, LNCS 2248, pp.514-532, Springer-Verlag, 2001.

[BGLS03] D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps", *EUROCRYPT 2003*, LNCS 2656, pp.416-432, Springer-Verlag, 2003.

[IKTY05] T. Izu, N. Kanaya, M. Takenaka and T. Yoshioka, "PIATS: A Partially Sanitizable Signature Scheme", *ICICS 2005*, LNCS 3783, pp. 72-83, Springer-Verlag, 2005.

[JMSW02] R. Johnson, D. Molnar, D. Song and D. Wagner, "Homomorphic Signature Schemes", *CT-RSA 2002*, LNCS 2271, pp.244-262, Springer, 2002.

[KL06] M. Klonowski and A. Lauks, "Extended Sanitizable Signatures", *ICISC 2006*, LNCS 4296, pp.343-355, Springer, 2006.

[LMRS04] A. Lysyanskaya, S. Micali, L. Reyzin and H. Shacham, "Sequential Aggregate Signatures from Trapdoor Permutations", *EUROCRYPT 2004*, LNCS 3027, pp. 74-90, Springer-Verlag, 2004.

[MHI06] K. Miyazaki, G. Hanaoka and H. Imai, "Digitally Signed Document Sanitizing Scheme from Bilinear Maps", *2006 ACM Symposium on Information, Computer ad Communications Security (ACMCCS 2006)*, proceedings, pp.343-354, ACM, 2006.

[MIM+05] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka and H. Imai, "Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control", The Institute of Electronics, Information and Communication Engineers (IEICE) Trans. on Fundamentals, Vol, E88-A, pp. 239-246, No. 1, January 2005.

[SBZ01] R. Steinfeld, L. Bull and Y. Zheng, "Content Extraction Signatures", *ICISC 2001*, LNCS 2288, pp. 285-304, Springer-Verlag, 2001.

A Novel Verifiably Encrypted Signature Scheme Without Random Oracle

Jianhong Zhang[1,2] and Jian Mao[1]

[1] Institute of Computer & Technology, Peking University,
Beijing 100871, P.R. China
{zhangjianhong,manjian}@icst.pku.edu.cn
http://www.icst.pku.edu.cn
[2] College of Science, North China University of Technology,
100041 Beijing, P.R. China
jhzhang@ncut.edu.cn

Abstract. Verifiably encrypted signature is a extended signature type and plays an important role in the constructing optimistic fair exchange. In the work, we propose a novel verifiably encrypted signature scheme without random oracles, and show that the security of the scheme is based on the difficulty of solving the Chosen-Target-Inverse-CDH with square problem. By comparing our scheme with Boneh *et al* scheme and S.Lu *et al* scheme, we show that our proposed scheme has the following advantages: (1) short signature size, only 320 bits; (2)low computation, only 2 pairing operations are needed in the phase of producing and verifying verifiably encrypted signature, respectively. (3)simplification-ability, the creation of verifiably encrypted signature is able to be completed in a logic step.

1 Introduction

As a extended signature, a verifiably encrypted signature (VES), which was proposed by N.Asokan [1], provides a way to encrypt a signature under a designated public key and subsequently proves that the resulting ciphertext indeed contains such a signature. It is often used as a building block of constructing optimistic fair exchange [2][3] over the Internet. It mainly relies on a trusted third party call Adjudicator, to realize fair exchange in optimistic way, that the adjudication is only needed in cases where a participant attempts to cheat the other or simply crashes. Another key feature of VES is that a participant can always force a fair and timely termination, without the cooperation of the other participants. Neither party can be left hanging or cheated so long as the adjudicator is available.

A valid VES can convince the verifier that a given cipher-text is the encryption of a signature on a given message. Alice creates a VES on a message by using her private key and an Adjudicator's public key. Bob is convinced that the encrypted signature is indeed of Alice, which he verifies using the public key of Alice and the Adjudicator. Even though Bob does not have the capability of decrypting the VES, the verification is performed without revealing any information about

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 65–78, 2007.

Alice's signature. If a dispute, the adjudicator can extract Alice's signature from VES on the message.

Since the concept of VES was included, J.Camenish[4] and G.Ateniese [5] proposed a verifiable encryption signature based on the discrete logarithm problem, respectively. In 2003, Boneh *et.al* [11] gave a security model of a verifiably encrypted signature and constructed a scheme satisfying the definitions based on the BLS short signature [9] and Zhang *et.al*[16] proposed a verifiably encryption signature with security proofs in random oracle based on bilinear Pairings. In 2005, by combining ID-based public key cryptography with verifiably encrypted signature, Gu *et.al*[13] proposed a ID-based verifiably encrypted signature scheme based on Hess' signature scheme, and claimed that their scheme was secure in random oracle model. Unfortunately, the scheme was showed to be existentially forgeable attack in [18]. Namely, any one can forge a verifiably encrypted signature on arbitrary a message. Because the security of most of the existing verifiably encrypted signature(VES) schemes are only proven to be secure in random oracle model, but security in the random oracle models does not imply security in the real world.

To address the above problem, in ICDCIT 2005, M.Choudary Gorantla *et.al* [14] proposed the first verifiably encrypted signature without random oracle based on BBS short signature [10], but detail security proof of the scheme has not been presented. Recently, Steve Lu *et al* proposed an efficient verifiably encrypted scheme without random oracles in [19] (for short S.Lu *et al* scheme). In the works, we present a novel and efficient verifiably encrypted signature scheme without random oracles, and show that the security of our proposed scheme depends on the difficulty of solving the Chosen-Target-Inverse -CDH with square problem. Comparing our proposed scheme with S.Lu *et al*scheme [12]and Boneh *et al* scheme [11], we find that our proposed scheme is more efficient in term of signature size and computation. Creation of verifiably encrypted signature is completed in a logic step, however, creation of verifiably encrypted signature in the literature[11][12] is completed in two steps, the one step is to produce the ordinary signature, the other step is to produce a verifiably encrypted signature based on ElGamal encryption. Furthermore, we also construct verifiably encrypted multisignature [12](ring signature, blind signature), which based on Waters' signature, by applying the following our proposed way.

The rest of the paper is organized as follows: Section 2 briefly describes the necessary background concepts; Section 3 presents security model of verifiably encrypted signature; Our verifiably encrypted signature scheme is proposed in section 4; Security proof and performance analysis of the scheme are given in section 5 and section 6. Finally, we conclude our work.

2 Preliminaries

In this section, we first present some background on groups with efficiently computable bilinear pair. Then, we give the corresponding mathematic difficulty problem which our proposed scheme is based on.

2.1 Bilinear Pairing

Definition 1. *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups of prime order q and $P \in \mathbb{G}_1$. A symmetric admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ has the following properties:*

1. *Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for any $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_1$ and $a, b \in Z_p$*
2. *Non-degenerate: It means that $e(P, P) \neq 1$*
3. *Computability: There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.*

Such an e can be constructed by Weil or Tate parings on the elliptic curves. As mentioned in [9][10], the Tate paring on MNT curves [20] gives us the efficient implementation.

Definition 2. *A BDH-parameter-generator is probabilistic algorithm that takes a security parameter λ as input and outputs a $5-$tuple $(q, P, \mathbb{G}_1, \mathbb{G}_2, e)$ where q is a λ-bit prime number, $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ are two groups with order q, $P \in \mathbb{G}_1$ is a generator, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is an admissible bilinear map.*

2.2 Complexity Assumptions

The Chosen-Target-CDH problem is defined as follows: the solver $\mathcal{S}$ receives as input a pair (P, aP), where P is a generator of G with prime order q, and $a \in Z_q$ is a random value. The solver $\mathcal{S}$ has adaptively access to the following two oracles:

- *Target Oracle*: this oracle outputs a random element $Z_i \in G$.
- *Helper Oracle*: this oracle takes as input an element $W_i \in G$ and outputs the element aW_i

We say that the solver $\mathcal{S}$ can $(q_t, q_h, d)-$ solve the Chosen-Target-CDH problem, for $q_t \geq d \geq q_h$, if it makes q_t and q_h queries, respectively, to the target oracle and helper oracles, and after that it outputs d pairs $((V_1, j_1), \cdots, (V_d, j_d))$ such that:

1. all the elements V_i are different;
2. for all $i \in \{1, 2, \cdots, d\}$, the relation $V_i = aZ_{j_i}$ is satisfied, where Z_{j_i} the element output by the target oracle in the $j_i - th$ query.

The Chosen-Target-Inverse-CDH problem is defined as follows: the solver $\mathcal{S}'$ receives as input a pair (P', aP'), where P' is a generator of $\mathbb{G}$ with prime order q, and $a' \in Z_q$ is a random value. The solver $\mathcal{S}'$ can adaptively access to the following two oracles:

- Target Oracle: this oracle outputs a random element $Z_i \in \mathbb{G}$
- Helper Oracle: this oracle takes as input an element $W_i \in \mathbb{G}$ and outputs the element $\frac{1}{a'}W_i$.

We say that the solver $\mathcal{S}$ can $(q_t, q_h, d)-$ solve the Chosen-Target-Inverse-CDH problem, for $q_t \geq d \geq q_h$, if it makes q_t and q_h queries, respectively, to the target oracle and helper oracles, and after that it outputs d pairs $((V_1, j_1), \cdots, (V_d, j_d))$ such that:

1. all the elements V_i are different;
2. for all $i \in \{1, 2, \cdots, d\}$, the relation $V_i = \frac{1}{a'} Z_{j_i}$ is satisfied, where Z_{j_i} is the element output by the target oracle in the $j_i - th$ query.

In [15], Herranz *et al* show that the Chosen-Target-CDH problem is equivalent to the Chosen-Target-Inverse-CDH problem.

To fit our purpose, we modify the Chosen-Target-Inverse-CDH problem, and give a variant. We call the Chosen-Target-Inverse-CDH with square problem, is defined as follows: the solver $\mathcal{S}'$ receives as input triple-tuple (P, aP, a^2P), where P is a generator of $\mathbb{G}$ with order q, and $a \in Z_q$ is a random number. The solver $\mathcal{S}'$ can adaptively access to the following two oracles:

- Target Oracle: this oracle outputs a random element $Z_i \in \mathbb{G}$
- Helper Oracle: this oracle takes as input an element $W_i \in \mathbb{G}$ and outputs the element $\frac{1}{a} W_i$.

We say that the solver $\mathcal{S}$ can $(q_t, q_h, d)-$ solve the Chosen-Target-Inverse-CDH square problem, for $q_t \geq d \geq q_h$, if it makes q_t and q_h queries, respectively, to the target oracle and helper oracles, and after that it outputs d pairs $((V_1, j_1), \cdots, (V_d, j_d))$ such that:

1. all the elements V_i are different;
2. for all $i \in \{1, 2, \cdots, d\}$, the relation $V_i = \frac{1}{a} Z_{j_i}$ is satisfied, where Z_{j_i} is the element output by the target oracle in the $j_i - th$ query.

For the Chosen-Target-Inverse-CDH with square problem, we can regard the problem as a variant, but in fact, the problem is the Chosen-Target-Inverse-CDH problem. Given (P, aP, a^2P), let $P' = aP$, then we can represent (P, aP, a^2P) into (P', aP'). Because P is a generator of $\mathbb{G}$ with prime order q and $gcd(q, a) = 1$, we know that $P' = aP$ is also a generator of $\mathbb{G}$ with order q by the group theory. Thus, the difficulty of solving the Chosen-Target-Inverse-CDH with square problem is equivalent to the difficulty of solving the Chosen-Target-Inverse-CDH problem.

3 Security Model of Verifiably Encrypted Signature

A verifiably encrypted signature scheme consists of six parts: VES.Setup, VES.$\sum$, VES.AdjKeyGen, VES.ESign, VES.EVerify and VES.Adj. Where VES.Setup produces system parameters, VES.$\sum$ is a ordinary signature scheme. VES.AdjKey Gen, VES.ESign, VES.EVerify and VES.Adj are used to provide the verifiably encrypted signature capability. The detail description is as follows:

VES.Setup: This is a polynomially probabilistic time (PPT) algorithm which, on input a security parameter λ, outputs the public parameters of system.

VES.$\sum$: This is a ordinary signature scheme which consists of three algorithms: $\sum.KeyGen$, $\sum.Sign$ and $\sum.Ver$, where the signer's private/public pair is $(a, P_a) \leftarrow \sum.KeyGen(\lambda)$.

VES.AdjKeyGen: This is a polynomially probabilistic time algorithm which, on input a security parameter λ, outputs the Adjudicator's private/public pair (s_a, Q_A).

VES.ESign: This is a PPT algorithm which, on input a message M, the signer's private key a and the adjudicator's public key Q_A, outputs a verifiably encrypted signature σ.

VES.EVerify: This is a polynomial-time deterministic algorithm which, on input a VES σ, the message M and the public key of adjudicator, Q_A, outputs 1 or 0.

VES.Adj: This is a PPT algorithm which, on input a VES σ and the private key s_a of the adjudicator, outputs a valid signature $\hat{\sigma}$.

Definition 3. *(Correctness.)For any message M, $VES.\sum$=($VES.\sum.KeyGen$, $VES.\sum.Sign$ and $VES.\sum.Ver$) is a ordinary signature scheme ; $(s_a, Q_A) \leftarrow VES.AdjKeyGen(\lambda)$, $(a, P_a) \leftarrow VES.\sum.KeyGen(\lambda)$. If the following three equations hold.*

$$\sigma \leftarrow VES.ESign(M, a, Q_A) \ \ and \ \ VES.EVerify(\sigma, Q_A, P_a, M) = 1$$

$$VES.\sum.Ver(M, VES.Adj(\sigma, s_a), P_a) = 1$$

As an extended signature scheme, a secure verifiably encrypted signature scheme should satisfy opacity besides the usual existential unforgeability under chosen message attack[6].

The existential unforgeability of verifiably encrypted signature without random oracle is defined via the following game between the simulator $\mathcal{S}$ and an adversary $\mathcal{F}$, $\mathcal{S}$ acts as the function of the signer and the Adjudicator.

1. the simulator $\mathcal{S}$ runs VES.Setup, and produces the system parameters ϕ, and returns ϕ to $\mathcal{F}$.
2. VESignature queries: $\mathcal{F}$ can query a verifiably encrypted signature to the signer under the public key of adjudicator Q_A, on input the message M, $\mathcal{S}$ runs VES.ESign algorithm to produce a verifiably encrypted signature σ, and returns to $\mathcal{F}$.
3. Adjudication queries: $\mathcal{F}$ can query the adjudicator to extract a signature of the message M, which is produced by the signer, on input a verifiably encrypted signature σ and the message M, $\mathcal{S}$ runs VES.Adj algorithm to extract a valid signature of message M, and returns it to $\mathcal{F}$.

We say $\mathcal{F}$ win the game if $\mathcal{F}$ outputs a forged verifiably encrypted signature (M^*, σ^*), if

- VES.EVerify$(M^*, \sigma^*, P_a, Q_A)$=1, where P_a is public key of the signer, Q_A is public key of the adjudicator.
- M^* has never submitted as one of the VESignature queries.

The success probability of an adaptively chosen message, which the attacker $\mathcal{F}$ win the above game, is defined as $Succ\mathcal{F}_{EUF}^{CMA}$.

Definition 4. *We say $Succ\mathcal{F}_{EUF}^{CMA}$ can $(t, q_s, q_a, \varepsilon)-$break the VES scheme if $Succ\mathcal{F}_{EUF}^{CMA}$ runs in time at most t, makes at most q_s VESignature queries, q_A Adjudication queries and $Succ\mathcal{F}_{EUF}^{CMA}$ is at least ε.*

The opacity of verifiably encrypted signature without random oracle is defined via the following game between the simulator $\mathcal{S}$ and an adversary $\mathcal{F}$.

1. the simulator $\mathcal{S}$ runs VES.Setup, and produces the system parameters ϕ, and returns ϕ to $\mathcal{F}$
2. $\mathcal{F}$ can make VESignature queries and Adjudication queries as ones of the above unforgeability game.

We say $\mathcal{F}$ win the game if $\mathcal{F}$ outputs a valid signature (M^*, γ^*), if

- VES.$\sum$.Ver(M^*, σ^*, P_a)=1, where P_a is public key of the signer.
- the verifiably encrypted signature of message M^* has never submitted as one of the Adjudication queries.

The success probability of an adaptively chosen message, which the attacker $\mathcal{F}$ win the above game, is defined as $Succ\mathcal{F}_{Opa}^{CMA}$.

Definition 5. *We say $Succ\mathcal{F}_{Opa}^{CMA}$ can $(t, q_s, q_A, \varepsilon)$-break the VES scheme if $Succ\mathcal{F}_{Opa}^{CMA}$ runs in time at most t, and makes at most q_s VESignature queries, q_A Adjudication queries and $Succ\mathcal{F}_{Opa}^{CMA}$ is at least ε.*

According to the state above, we know that the property ***opacity*** of verifiably encrypted signature denotes that it is difficult, given a verifiably encrypted signature, to extract an ordinary signature on the same message.

4 Our Proposed Scheme

In the section, we present the design of our new verifiably encrypted signature scheme. The idea of our scheme can been regarded as completing public key encryption of signature and the signature on the message M in a logic step. Our proposed scheme is based on Waters' signature scheme [8]. In our scheme, the messages will be signatures on bitstrings of the form $\{0,1\}^n$ for some fixed n. In practice we can apply a collision-resistant hash function $H_n : \{0,1\}^* \rightarrow \{0,1\}^n$ to sign messages of arbitrary length. For convenience, hashed message $\mathbf{m}$ are represented as $(m_1, m_2, \cdots, m_n)$ with $m_i \in \{0,1\}$ for all $i \in \{1, \cdots, n\}$.

- VES.Setup: public parameters includes the output $(q, P, \mathbb{G}_1, \mathbb{G}_2, e)$ of a BDH-parameter generator as well as as integer n, a collision-resistant hash function $H_n : \{0,1\}^* \rightarrow \{0,1\}^n$, random elements $P', U' \in \mathbb{G}_1$ and a random $n-$tuple $(U_1, \cdots, U_n) \in \mathbb{G}_1^n$. We set a map $F : \{0,1\}^n \rightarrow \mathbb{G}_1$ with mapping string $\mathbf{m}$ on $F(\mathbf{m}) = U' + \sum_{i=1}^{n} m_i U_i$. Finally, the public parameters is as follows:

$$(n, q, \mathbb{G}_1, \mathbb{G}_2, e, P, P', U', U_1, \cdots, U_n, F, H_n)$$

(**Note that**, in the following, $\sum$ is Waters' signature scheme which consists of three algorithms: $\sum$.KeyGen, $\sum$.Sign and $\sum$.Ver).

- VES.$\sum$.KeyGen: a signer, Alice, randomly chooses $a \in Z_q$ as the his private key, and computes the corresponding public key $P_a = aP$.
- VES.$\sum$.Sign: Let $\mathbf{m} = H_n(M)$, for a message $M \in \{0,1\}^*$, $\mathbf{m} = (m_1, m_2, \cdots, m_n)$ with $m_i \in \{0,1\}$. A signature of the message M is produced by picking $r \in_R Z_q$ and setting signature $\sigma = (\sigma_1, \sigma_2)$ with $\sigma_1 = aP' + r(U' + \sum_{i=1}^{n} m_i U_i)$ and $\sigma_2 = rP$.
- VES.$\sum$.Verfy: a purported signature σ on the message M is accepted if on only if

$$e(\sigma_1, P) = e(P_a, P')e(\sigma_2, U' + \sum_{i=i}^{n} m_i U_i) \tag{1}$$

- VES.AdjKeyGen: an adjudicator randomly choose $s_a \in Z_q$ as private key, and computes the corresponding public key $Q_A = s_a P'$.
- VES.ESign: given a signer,Alice's private key a and the adjudicator's public key Q_A, to sign the message M, Alice computes $\mathbf{m} = H_n(M)$ and randomly picks $r_A \in Z_q^*$ to compute the signature

$$\gamma = (\gamma_1, \gamma_2) = (aQ_A + r_A F(\mathbf{m}), r_A P)$$

- VES.EVerify: given a verifiably encrypted signature γ parsed as $(\gamma_1, \gamma_2) \in \mathbb{G}_1^2$. Accept if the following equation (2) holds:

$$e(\gamma_1, P) = e(P_a, Q_A)e(\gamma_2, U' + \sum_{i=1}^{n} m_i U_i) \tag{2}$$

- VES.Adj: given a verifiably encrypted signature γ on the message M, the adjudicator first verifies whether the verifiably encrypted signature γ is valid, and parses it as $(\gamma_1, \gamma_2) \in \mathbb{G}^2$. Compute

$$\gamma_1' = s_A^{-1}\gamma_1 \text{ and } \gamma_2' = s_A^{-1}\gamma_2$$

and output the signature (γ_1', γ_2').

For the scheme above, it is easy to verify that the scheme is valid. If all parties are honest, then the signature (γ_1', γ_2') satisfies the equation above (1). Since

$$\begin{aligned} e(\gamma_1', P) &= e(s_A^{-1}\gamma_1, P) \\ &= e(s_A^{-1}(aQ_A + r_A F(\mathbf{m})), P) \\ &= e(aP' + s_A^{-1} r_A F(\mathbf{m}), P) \\ &= e(aP, P')e(F(\mathbf{m}), s_A^{-1} r_A P) \\ &= e(P_a, P')e(F(\mathbf{m}), \gamma_2') \end{aligned}$$

It shows that the extracted signature by the adjudicator is indeed a valid one. Thus, we can straightforwardly obtain that our proposed scheme satisfies *correctness* of verifiably encrypted signature scheme.

5 Security Analysis

In this section, we will give security analysis and show that our proposed scheme is unforgeable and opaque under adaptively chosen message attack.

Theorem 1. *Our proposed verifiably encrypted signature scheme is* $(t, q_s, q_A, \epsilon)-$ *unforgeable if the Waters' signature scheme[8] is* $(t', q'\epsilon')-$*unforgeable, where*

$$t' = t + O(q_s + q_A) \text{ and } q' = q_s \text{ and } \epsilon' = \epsilon$$

Proof. Assume there exists a $(t, q_s, q_A, \epsilon)-$ adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ as subroutine to forge a Waters' signature with probability at least ϵ'.

Algorithm $\mathcal{B}$ is given a Waters signature public key P_a and other public parameters $(\mathbb{G}_1, \mathbb{G}_2, P, P', U', U_1, \cdots, U_n, H_n(\cdot), e)$. Then it randomly chooses $\alpha_A \in Z_q$ as the adjudicator's private key and sets the corresponding public key $Q_A = \alpha_A P'$, and provides the adversary $\mathcal{A}$ with P_a and Q_A.

$\mathcal{B}$ simulates the signing oracles and adjudicating oracles as follows:

Signing Oracles: when the adversary $\mathcal{A}$ queries a verifiably encrypted signature on a certain message M, the challenger $\mathcal{B}$ queries a signature ,(σ_1, σ_2) on the message M from its own signing oracle, and obtains a signature on the message M. Then the challenger $\mathcal{B}$ computes

$$\gamma_1' = \alpha_A \sigma_1 \text{ and } \gamma_2' = \alpha_A \sigma_2$$

and returns (γ_1', γ_2') to $\mathcal{A}$ as verifiably encrypted signature on M.

Adjudication Queries: When the adversary $\mathcal{A}$ requests adjudication for (γ_1', γ_2'), a verifiably encrypted signature on a message M under key P_a and adjudicator key Q_A. $\mathcal{B}$ responds with $(s_A^{-1}\gamma_1', s_A^{-1}\gamma_2')$ by $VES.Adj$ algorithm of the above scheme. Note that, since $\mathcal{B}$ knows the adjudicator's private key s_A, it can compute $(s_A^{-1}\gamma_1', s_A^{-1}\gamma_2')$.

Output: Finally, $\mathcal{A}$ outputs a valid and nontrivial verifiably encrypted signature (γ_1^*, γ_2^*) on a message M^* at which A must never have made a verifiably encrypted signature.

Obviously, the challenge $\mathcal{B}$ can compute as

$$\sigma_1^* = s_A^{-1}\gamma_1^* \text{ and } \sigma_2^* = s_A^{-1}\gamma_2^*$$

It denotes that (σ_1^*, σ_2^*) is a valid Waters signature on the message M^*. In other words, it means that algorithm $\mathcal{B}$ can forge a Waters signature in the non-negligible probability.

According to the above process, we know that algorithm $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does. Its running time is $\mathcal{O}(1)$ for each of $\mathcal{A}$'s queries and for computing the final output. □

Theorem 2. *If there exists* $(t, q_s, q_A, \epsilon)-$*forger* $\mathcal{A}$ *against opaque of our proposed verifiably encrypted signature scheme, then there exists a* $(q_t, q_h, d)-$ *solver* $\mathcal{B}$ *of the Chosen-Target-Inverse-CDH with square problem, which also succeeds with probability* $\epsilon' \approx \epsilon - \frac{q_A}{q^2}$, $q_t = q_s \geq q_A + 1$, $q_h = q_A$, $d = q_A + 1$.

Proof. Assume there exists a $(t, q_s, q_A, \epsilon)-$ adversary $\mathcal{A}$. We are going to construct a solver $\mathcal{B}$ of the Chosen-Target-Inverse-CDH with square problem, which makes use of $\mathcal{A}$ to solve the Chosen-Target-Inverse-CDH with square problem with probability at least ϵ'.

First of all, $\mathcal{B}$ initializes $\mathcal{A}$, which sets up the system parameters. The solver $\mathcal{B}$ chooses a group $\mathbb{G}_1$ with prime order q which admits a bilinear pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. P is a generator of $\mathbb{G}_1$.

The solver $\mathcal{B}$ asks for an instance of the Chosen-Target-Inverse-CDH with square problem in the group $\mathbb{G}_1$. It receives 3-tuple (P, aP, a^2P) for some random and secret value $a \in Z_q$; it is also provided with access to the target oracles and the helper oracles.

And it randomly chooses $k' \in Z_p$ to compute $P' = k'P$. It sets $Q_A = aP' = k'aP$ and $P_a = aP$ as the public key of adjudicator and the signer, respectively.

Let $\lambda = 2q_A$, algorithm $\mathcal{B}$ picks $\kappa \in_R \{0, \cdots, k\}$. We assume that $\lambda(k+1) < q$ which implies $0 \leq \kappa\lambda \leq q$. Algorithm $\mathcal{B}$ randomly chooses $(x', x_1, x_2, \cdots, x_k) \in_R Z_\lambda^{k+1}$, where $Z_\lambda = \{0, 1, \cdots, \lambda - 1\}$, and $(y', y_1, y_2, \cdots, y_k) \in_R Z_q^{k+1}$ and sets the following relations:

$$u' = (x' - \kappa\lambda)P' + y'P \text{ and } u_i = (x_i)P' + y_iP \text{ for } i = 1, \cdots, k$$

For convenient explanation, we will include two functions:

$$J(\mathbf{m}) = x' + \sum_{i=1}^{n} m_i x_i - \kappa\lambda \quad and \quad K(\mathbf{m}) = y' + \sum_{i=1}^{n} m_i y_i$$

where $H_n(M) = (m_1, \cdots, m_n)$, $m_i \in \{0, 1\}$ and $H_n(\cdot)$ is a collision-resistant hash function.

According to the above relations, we can obtain

$$F(\mathbf{m}) = u' + \sum_{i=1}^{n} m_i u_i = J(\mathbf{m})P' + K(\mathbf{m})P$$

Finally, all public parameters $(\mathbb{G}_1, \mathbb{G}_2, P, P', Q_A, P_a, u', u_1, \cdots, u_k, q, H_n(\cdot), e)$ are passed to $\mathcal{A}$.

Queries: once $\mathcal{A}$ is started with public parameters and public keys P_a, Q_A as input, two kinds of queries may occur.

Verifiably Encrypted Signing Queries: when $\mathcal{A}$ requests a verifiably encrypted signature on M under the challenge key P_a and the adjudicator key Q_A, let $\mathbf{m} = H_n(M) = (m_1, \cdots, m_n) \in \{0, 1\}^k$. If $J(\mathbf{m}) \neq 0 \bmod q$, the solver $\mathcal{B}$ proceeds as follows.

- the solver $\mathcal{B}$ makes a query to its target oracle, and receives a random element $R \in_R \mathbb{G}_1$ as answer from its target oracle. Note that, in essence, for the random element R, there exists a certain $r \in_R Z_q$ which satisfies $R = rP$, only r is unknown to the solver $\mathcal{B}$.

– compute

$$A = r(u' + \sum_{i=1}^{k} m_i u_i) = ru' + r\sum_{i=1}^{k} m_i u_i$$
$$= r(x' - \kappa\lambda)P' + ry'P + \sum_{i=1}^{k} rm_i(x_i)P' + y_i P$$
$$= (x' - \kappa\lambda)k'R + y'R + \sum_{i=1}^{k} m_i(x_i k'R + y_i R) \quad (3)$$

Although r is unknown to the solver $\mathcal{B}$, we can compute A through the most right side of the above equation (3).

– compute

$$\hat{\gamma}_1 = (-K(\mathbf{m})/J(\mathbf{m}))P_a + A$$
$$= (-K(\mathbf{m})/J(\mathbf{m}))P_a + r(u' + \sum_{i=1}^{k} m_i u_i)$$
$$= (-K(\mathbf{m})/J(\mathbf{m}))aP + ((x' - \kappa\lambda)k'R + y'R + \sum_{i=1}^{k} m_i(x_i k'R + y_i R))$$
$$= k'aP + (-a/J(\mathbf{m}))(k'J(\mathbf{m})P + K(\mathbf{m})P) + (k'J(\mathbf{m})R + J(\mathbf{m})R)$$
$$= k'aP + (-a/J(\mathbf{m}))(k'J(\mathbf{m})P + K(\mathbf{m})P) + r(k'J(\mathbf{m})P + J(\mathbf{m})P)$$
$$= k'aP + (r - a/J(\mathbf{m}))(k'J(\mathbf{m})P + K(\mathbf{m})P)$$

– compute

$$\gamma_1 = \hat{\gamma}_1 - k'P_a + k'a^2P \quad and \quad \gamma_2 = (-\frac{1}{J(\mathbf{m})})P_a + R$$

Remark 1. Here, in fact $\gamma_1 = k'a^2P + (r - a/J(\mathbf{m}))(k'J(\mathbf{m})P + K(\mathbf{m})P) = aQ_A + (r - a/J(\mathbf{m}))(k'F(\mathbf{m})P + J(\mathbf{m})P)$.

If $J(\mathbf{m}) = 0$, the challenge $\mathcal{B}$ computes as follows:

– the solver $\mathcal{B}$ makes a query to its target oracle, and receives a random element $R \in_R \mathbb{G}_1$ as answer from its target oracle;
– randomly choose $\tau \in Z_q$ to compute $\gamma_1 = k'a^2P + \tau K(\mathbf{m})R$ and $\gamma_2 = \tau R$

Obliviously, (γ_1, γ_2) is a valid verifiably encrypted signature. Finally, the solver $\mathcal{B}$ returns verifiably encrypted signature of M,(γ_1, γ_2), to $\mathcal{A}$.

Adjudication Queries: Suppose that $\mathcal{A}$ requests adjudication on (γ_1, γ_2) on the message M_i. The solver $\mathcal{B}$ first verifies whether the verifiably encrypted signature (γ_1, γ_2) is valid. If it is valid, then the solver $\mathcal{B}$ sends γ_2 to the helper oracle. And obtain the corresponding answers $\sigma_{i_2} = \frac{1}{a}\gamma_2$ from the help oracle and compute $\sigma_{i_1} = k'aP + F(\mathbf{m})\sigma_{i_2}$, where $\mathbf{m} = H_n(M_i)$. The solver $\mathcal{B}$ returns

$(\sigma_{i_1}, \sigma_{i_2})$ to $\mathcal{A}$ as the extracted signature of the message M_i. In fact, $(\sigma_{i_1}, \sigma_{i_2})$ satisfy $\sigma_{i_1} = \frac{1}{a}\gamma_1$ and $\sigma_{i_2} = \frac{1}{a}\gamma_2$. Since they satisfy the following relations:

$$\begin{aligned}\sigma_{i_1} = \frac{1}{a}\gamma_1 &= aP' + \frac{(r - a/J(\mathbf{m}))}{a}(k'J(\mathbf{m})P + K(\mathbf{m})P) \\ &= aP' + \frac{(r - a/J(\mathbf{m}))}{a}F(\mathbf{m})P \end{aligned} \tag{4}$$

$$\begin{aligned}\sigma_{i_2} = \frac{1}{a}\gamma_2 &= \frac{1}{a}((-\frac{1}{J(\mathbf{m})})P_a + R) \\ &= (-\frac{1}{J(\mathbf{m})})P + \frac{1}{a}R \\ &= \frac{(r - a/J(\mathbf{m}))}{a}P \end{aligned} \tag{5}$$

Obviously, $(\sigma_{i_1}, \sigma_{i_2})$ is a Waters signature.

Output: Finally, algorithm $\mathcal{A}$ outputs a signature (σ_1^*, σ_2^*) on a message M^* from verifiably encrypted signature (γ_1^*, γ_2^*) of the message M^*; it must not have queried its adjudication oracle at M^*, and (γ_1^*, γ_2^*) is ever a queried signature accessing to **Verifiably Encrypted Signing Oracle** . Obviously, the signature (σ_1^*, σ_2^*) on the message M^* is a Waters signature and satisfies the above equation (1).

Thus,the environment of $\mathcal{A}$ is perfectly simulated by $\mathcal{B}$. And the solver $\mathcal{B}$ outputs $q_s(\geq q_A + 1)$ Waters signatures $(\sigma_{i_1}\sigma_{i_2})$ with probability ε. It means, when the solver $\mathcal{B}$ queries its Target oracle, we have R_i $(i = 1, \cdots, q_s)$ which is returned by Target oracle. While the solver $\mathcal{B}$ only make q_A queries for its help oracle. Finally, the extracted signatures $(\sigma_{i_1}\sigma_{i_2})$ on $q_A + 1$ different message M_i are obtained and they satisfy

$$e(\sigma_{i_1}, P) = e(P_a, P')e(\sigma_{i_2}, U' + \sum_{i=i}^{n} m_i U_i) \tag{6}$$

Thus, for $i = 1, \cdots, q_A + 1$, the solver $\mathcal{B}$ outputs the pair (V_i, j_i), where

$$V_i = \sigma_{i_2} + \frac{1}{J(\mathbf{m}^{(i)})}P, \quad where \ \ \mathbf{m}^{(i)} = H_n(M_i)$$

Note that, here $V_i = \frac{1}{a}R_i$. Since all the signature $(\sigma_{i_1}\sigma_{i_2})$ are different, we have that all the values V_i for $i = 1, \cdots, q_A$ are also different.

Because the probability, which the adversary produces a extracted signature of a message from given a verifiably encrypted signature, is $\frac{1}{q^2}$. According to the state above, solver $\mathcal{B}$ makes $q_s(\geq q_A + 1)$ queries to its target oracle, makes q_A queries to its helper oracle, while it output $q_A + 1$ different and valid pair (V_i, j_i) with the probability $\epsilon' \approx \epsilon - \frac{q_A}{q^2}$. It means that solver $\mathcal{B}$ can solve the Chosen-Target-Inverse-CDH with square problem. □

6 Performance Analysis and Further Discussion

6.1 Performance Analysis

In this section, we will analyze efficiency of our proposed scheme by comparing with Boneh *et.al* scheme[11] and S.Lu *et al* scheme [12]. For convenient comparison, we instantiate pairing-based schemes using Barreto-Naehrig curves[10] with 160-bit point representation. We present the detail comparisons in Table 1. The size column gives signature length at the 1024-bit security level. The verification and Generation columns give the computational costs of those operations. "R.O" column denotes if the security proof uses random oracles. Let P_m be scalar multiplication on the curve, P_e be pairings computation. and n is the output length of a collision-resistant hash function. For a given signer and adjudicator, the public key P_a of the signer and the public key Q_A of adjudicator are fixed. Thus, If we pre-computes $e(P_a, Q_A)$, then only 2 pairs computation are executed in the VES.EVerify phase.

Signature Length. A signature size in our proposed scheme only consists of two elements (γ_1, γ_2), where γ_1 and γ_2 are in $\mathbb{G}_1$. When using a supersingular elliptic curve over finite field F_{p^n} with embedding degree $k = 6$ and the modified Weil pairing or Tate pairing [9,20], the length of an element in $\mathbb{G}_1$ can be approximately $log_2 q$ bits, thus the total signature length is approximately $2log_2 q$ bits.

Table 1. Comparison of our proposed scheme with Boneh *et.al* scheme[11] and S.Lu *et al* scheme [12]

Scheme	R.O	Size	Verification	Generations	Adjudication
Boneh *et.al* scheme	Yes	320 bits	$3P_e$	$3P_m$	$2P_m$
S.Lu *et al* scheme	No	480 bits	$3P_e + n/2P_m$	$(n/2 + 5)P_m$	$3P_m$
Our scheme	No	320 bits	$2P_e + n/2P_m$	(n/2+3)P_m	$2P_m$

From Table 1, we know that our scheme is one of the most efficient schemes, the length of the signature is very short, about 320 bits; and the security of the scheme is proven secure in the standard model.

6.2 Further Discussion

From the above our proposed scheme, we know that, given a verifiably encrypted signature (γ_1, γ_2) of message M, any one can produce a new signature on the message M by randomly choosing $x \in_R Z_p$ and computing $\hat{\gamma}_1 = \gamma_1 + xF(\mathbf{m})$ and $\hat{\gamma}_2 = \gamma_2 + xP$ where $\mathbf{m} = H_n(M)$. The reason to appear this problem is that Waters' signature [8] itself doesn't satisfy strong unforgeability, while our proposed VES scheme is based on Waters' scheme, thus, our proposed scheme does't satisfy strong unforgeability. In other words, an adversary can generate a new signature on a previously signed message.

According to the above paragraph, we know if γ_2 of a verifiably encrypted signature $(\gamma_1, \gamma_2)$is fixed, then we can realize strong unforgeability. Thus, to realize strong unforgeability of the scheme, we can consider the following ways.

1. we can change $\mathbf{m} = H_n(M)$ into $\mathbf{m} = H'_n(M, \gamma_2)$, such revision is able to complete strong unforgeability. However, it seems the security proofs of such schemes are different from the description in section 5.
2. another way is that we can introduce a secure signature scheme $\sum$ without random oracle, such as BBS[10], and use $\sum$ to produce a signature on the γ_2 to resist that γ_2 is amended. Then security proof of the revised scheme is similar to the description in section 5.

7 Conclusion

As a building block, verifiably encrypted signature scheme plays an important role in fair exchange. However, most of the existing schemes have been proven secure in random oracle models in [7]. But security in the random oracle models does not imply security in the real world. In the work, we propose a novel verifiably encrypted signature scheme without random oracles, and show that the security of the scheme is based on the Chosen-Target-Inverse-CDH with square problem . By comparing our scheme with Boneh *et.al* scheme[11] and S.Lu *et al* scheme [12], we show that our scheme has the advantage over Boneh *et.al* scheme and S.Lu *et al* scheme with respective to the signature length and computation. Furthermore, we also construct verifiably encrypted multisignature [12](ring signature, blind signature), which based on Waters' signature, by applying our proposed way in this paper.

Acknowledgement

We thanks the anonymous referees for their very valuable comments. This work is supported by China Postdoctoral Science Foundation(NO:20060390007) and the open fund of State Key Laboratory of Information Security.

References

1. Asokan,N., Shoup, V., Waidner,M. Optimistic Fair Exchange of Digital Signature (extended abstract), In: Advances in Cryptology-Eurocrypt'98. LNCS 1403, pp 591-606, springer-verlag, 1998
2. Ateniese.G, Efficient Verifiable Encryption (and Fair Exchange) of Digital Signatures, In: Proc. of the 6th Conference on CCS.ACM Press pp 138-146 1999
3. Bao. F,Deng, R.H, and Mao.W, Efficient and Practical fair exchange protocols with off-line TTP. In IEEE Symposium on Security and Privacy, Oakland, CA, 1998
4. J.Camenisch and Victor Shoup, Practical Verifiable Encryption and Decryption of Discrete Logarithms, CRYPTO 2003, LNCS 2729, pp 126-144, Springer-verlag, 2003

5. G.Ateniese, Verifiable Encryption of Digital Signature and Applications, ACM Transactions on Information and System Security, Vol7, NO.1, February 2004, pp 1-20
6. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal of Computing, 17/2:281C 308, 1988.
7. M.Bellare and P.Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. 1st ACM Conference on Computer and Computer and Communications Security, pp 62-73, 1993
8. B.Waters. Efficient Identity-based encryption without random oracles. In R.Cramer, editor, Procceedings of Eurocrypt 2005, LNCS 3494, pp 114-127, springer, May 2005.
9. D.Boneh, B.Lynn, and H.Shacham. Short signture from the Weil pairing. Journal of Cryptology, Vol 17(4), pp 297-319, September.2004.
10. Dan Boneh, Xavier Boyen, Short signatures without random oracles, In proceedings of Eurocrypt 2004, LNCS 3027, pp. 56-73, 2004, springer-Verlag
11. D.Boneh, C.Gentry, B.Lynn, and H.Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Proceedings of Euro-crypt 2003, LNCS 2656, pp 416-432, springer-verlag, 2003
12. S.Lu, R.Ostrovsky, A.Sahai, H.Shacham and B.Waters, Sequential aggregate signtures and multisignatures without random oracles, In Proceedings of Eurocrypt 2006, LNCS 4004, pp 465-485, springer-verlag, 2006
13. C.X Gu and Y.F Zhu, An ID-based Verifiable Encrypted Signature Scheme Based on Hess's Scheme,CISC 2005, LNCS 3822, pp 42-52, springer-verlag, 2005
14. M.Choudary Gorantla and Ashutosh Saxena, Verifiably Encrypted Signature Without Radom Oracles,ICDCIT 2005, LNCS 3816, pp 357-363, springer-verlag, 2005
15. J.Herranz, and F.Laguillaumie, Blind Ring Signatures Secure under the Chosen-Target-CDH Assumption, Information Security Conference 2006, LNCS 4176, pp117-130 Springer-verlag, 2006
16. Zhang, F., Safavi-Naini, R., Susilo, W., Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairings. In: Progess in Cryptology-Indocrypt'03, LNCS 2904, pp 191-204 Springer-verlag, 2003
17. D.Pointcheval and J.Stern,Security Arguments for Digital Signatures and Blind Signatures,Journal of Cryptology, Volume 13 - Number 3,Pages 361-396.
18. J.Zhang and W.Zou, A Robust Verifiably Encrypted Signature Scheme,Emerging Directions in Embedded and Ubiquitous Computing, LNCS 4097, pp 731-740, 2006, Springer-Verlag.
19. A.Miyaji, M.Nakabayashi, and S.Takano, New explicit conditions of elliptic curve traces for FR-redution, IEICE Trans. Fundamentals, Vol.E84-A, No 5, pp 1234-1243, 2001
20. A.Joux, The Weil and Tate pairings as building blocks for public key cryptosystems, ANTS 2002, LNCS 2369, pp 20-32, Springer-Verlag, 2000.

Certificate Based (Linkable) Ring Signature

Man Ho Au[1], Joseph K. Liu[2], Willy Susilo[1], and Tsz Hon Yuen[1]

[1] Centre for Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
{mhaa456,wsusilo}@uow.edu.au, johnyuenhk@gmail.com
[2] Department of Computer Science
University of Bristol
Bristol, BS8 1UB, UK
liu@cs.bris.ac.uk

Abstract. In this paper, we propose a new notion called *Certificate Based Ring Signature* (CBRS) that follows the idea of Certificate Based Encryption (CBE) presented by Gentry in EuroCrypt 2003. It preserves the advantages of CBE such as implicit certificate and no private key escrow. At the same time it inherits the properties of normal ring signature such as anonymity and spontaneity. We provide its security model and a concrete implementation. In addition, we also propose a variant of CBRS, called *Certificate Based Linkable Ring Signature* (CBLRS). It is similar to CBRS, except with linkability. That is, it allows the public to verify whether two given signatures are generated by the same signer, yet preserves the anonymity of this user. It can be seen as the Certificate Based version of normal linkable ring signature.

Keywords: Certificate Based, Ring Signature, Linkability.

1 Introduction

Public Key Infrastructure (PKI). In traditional public key cryptography (PKC), a user Alice signs a message using her private key. A verifier Bob verifies the signature using Alice's public key. However, the public key is just a random string and it does not provide authentication of the signer by itself. This problem can be solved by using a certificate generated by a trusted party called the Certificate Authority (CA) that provides an unforgeable signature and trusted link between the public key and the identity of the signer. The hierarchical framework is called public key infrastructure (PKI) to issue and manage certificate (chain). In this case, before the verification of a signature, Bob needs to obtain Alice's certificate in advance and verify the validity of her certificate. If it is valid, Bob extracts the corresponding public key which is then used to verify the signature. In the point of view of a verifier, it takes two verification steps for independent signatures. It seems not efficient and not practical enough, especially when the number of users is very large.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 79–92, 2007.

Identity-Based Cryptography (IBC). Identity-based cryptography (IBC), invented by Shamair [25] in 1984, solves this problem by using Alice's identity (or email address) which is an arbitrary string as her public key while the corresponding private key is a result of some mathematical operation that takes as input the user's identity and the master secret key of a trusted authority, referred as "Private Key Generator (PKG)". In this way, the certificate is implicitly provided and it is no longer necessary to explicitly authenticate public keys. The main disadvantage of identity-based cryptography is an unconditional trust to the PKG. This is even worse than traditional PKC since the secret key of every user is generated by the PKG, it can impersonate any user, or decrypt any ciphertext.

Certificate Based Cryptography (CBC). To integrate the merits of IBC into PKI, Gentry [14] introduced the concept of certificate based encryption (CBE). A CBE scheme combines a public key encryption scheme and an identity based encryption scheme between a certifier and a user. Each user generates his own private and public key and request a certificate from the CA while the CA uses the key generation algorithm of an identity based encryption (IBE) [9] scheme to generate certificate. In this way, the certificate is implicitly used as the private key of the user as the signing key (and decryption key). In addition to CBE, the notion of certificate based signature (CBS) was first suggested by Kang et al [15].

In parallel to CBC, certificateless cryptography [2] and self-generated-certificate public key cryptography [16] are another solutions to the key escrow problem inherited by IBC.

Ring Signature. A ring signature scheme [23,20,6,11] allows members of a group to sign messages on behalf of the group without revealing their identities, that is, preserving signer anonymity. In addition, it is not possible to decide whether two signatures have been issued by the same group member. Different from a group signature scheme [10], the group formation is spontaneous and there is no group manager to revoke the identity of the signer. In other words, under the assumption that each user is already associated with a public key of some standard signature scheme, a user can form a group by simply collecting the public keys of all the group members including his own. These diversion group members can be totally unaware of being conscripted into the group.

Linkable Ring Signature. Linkable ring signatures was first proposed by Liu et al [18] in 2004. In this notion, the identity of the signer in a ring signature remains anonymous, but two ring signatures can be linked if they are signed by the same signer. Linkable ring signatures are suitable in many different practical applications, such as e-voting and e-cash [26]. Original ring signatures cannot be used for e-voting because any double votes cannot be detected as they are unlinkable. No one is able to find out whether any two signatures (with two votes) are generated by the same voter or not. Linkable ring signatures solve this problem by allowing the public to detect for any signer producing two or more signatures (votes).

Note that linkability is compulsorily embedded into the signature instead of voluntarily added in linkable ring signatures. If the signer refuses to add

the correct linking information, the whole signature is invalid. In other words, linkability is enforced by the verifier. The signer cannot decline to do so. This is different from voluntarily added linkability. In this case, whether allowing the signature to be linked or not can be decided by the signer. This issue is also explained in [18].

1.1 Problems of Ring Signature in PKI and IBC

As mentioned before, in traditional PKI it takes two verification steps for independent signatures. It is even worse in the case of ring signature. In the verification of a ring signature, one requires the public key of n users. That means one needs to verify the validity of these n public keys using the corresponding certificates in advance. Furthermore, in the signing stage, the actual signer also needs to obtain the public key of other $n-1$ non-signers in order to generate a ring signature. In order to authenticate the ownership of these $n-1$ public keys, the actual signer needs to verify their corresponding certificates before generating the ring signature. Note that this process is only applied to ring signature scheme since normal digital signature does not require the knowledge of public key of other users in the signing stage. This made the problem worse in (linkable) ring signature, especially when n is large. For example, if PKI based linkable ring signature scheme is used in an e-voting system where the number of voters is about one million. Every voter needs to verify one million certificates before the execution of the signing stage. It is extremely inefficient and makes it not practical to be applied to large scale voting event.

On the other hand, in the case of ID-based setting [5], we also have to take extra care for the design of schemes. While some of the existing schemes provide anonymity *unconditionally*, others are computational only. The Private Key Generator (PKG) itself may have extra advantage in breaking the anonymity since it is in possession of all the private keys. This problem does not sound serious in normal ID-based ring signature scheme because almost all existing schemes is unconditionally anonymous. However, in the case of linkable ring signatures [18,27,19,26,3,17] where the verifier is able to determine whether two signatures are signed by the same signer, it is still an open problem to construct one with unconditional anonymity. Within the constraint of computational anonymity, it is a great challenge of providing privacy in an ID-based setting (to the PKG who knows the private key of every user). This issue is also addressed in [4].

Although ring signature schemes and linkable ring signature schemes have been proposed for a few years already, these problems have not been mentioned and investigated before in the literature.

1.2 Contribution

In this paper, we propose a new notion called *Certificate Based Ring Signature* (CBRS). It preserves all properties of a normal ring signature while using certificate based cryptosystem setting. This combination gains the advantages of certificate based systems, by removing explicit certificate chain verification and

key escrow problem. They are especially important in ring signature as mentioned above.

In addition, we also propose its variant, called *Certificate Based Linkable Ring Signature* (CBLRS). It is the linkable version of CBRS such that it allows a verifier to find out whether two given signatures are generated by the same signer or not. It can be regarded as the certificate based version of normal linkable ring signature.

We give security model and concrete implementation of both notions.

Organization. In the rest of the paper, it is organized as follow. We give some mathematical prelminiaries in Section 2. The security model of both notions are presented in Section 3. It is followed by our concrete implementation in Section 4. The paper is concluded in Section 5.

2 Preliminaries

2.1 Notations

Let e be a bilinear map such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic multiplicative groups of prime order p.
- each element of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ has unique binary representation.
- g_0, h_0 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(h_0) = g_0$.
- (Bilinear) $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and a, $b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$.
- (Non-degenerate)$e(g_0, h_0) \neq 1$.

$\mathbb{G}_1$ and $\mathbb{G}_2$ can be same or different groups. We say that two groups $(\mathbb{G}_1, \mathbb{G}_2)$ are a bilinear group pair if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$, the isomorphism ψ and the bilinear mapping e are all efficiently computable.

2.2 Mathematical Assumptions

Definition 1 (Decisional Diffie-Hellman (DDH) Assumption). *The Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}_1$ is defined as follows: On input a quadruple $(g, g^a, g^b, g^c) \in {\mathbb{G}_1}^4$, output 1 if $c = ab$ and 0 otherwise. We say that the (t, ϵ)-DDH assumption holds in $\mathbb{G}_1$ if no t-time algorithm has advantage at least ϵ over random guessing in solving the DDH problem in $\mathbb{G}_1$.*

Definition 2 (q-Strong Diffie-Hellman (q-SDH) Assumption). *The q-Strong Diffie-Hellman (q-SDH) problem [7] in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follow: On input a $(q+2)$-tuple $(g_0, h_0, h_0^x, h_0^{x^2}, \cdots, h_0^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, output a pair (A, c) such that $A^{(x+c)} = g_0$ where $c \in \mathbb{Z}_p^*$. We say that the (q, t, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.*

Definition 3 (External Diffie-Hellman(XDH) Assumption). *[13,24,21,8] For a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, the XDH assumption holds if DDH problem is hard in $\mathbb{G}_1$.*

3 Security Model

3.1 Certificate Based Ring Signatures

The definition of certificate based ring signature schemes parallels the definition of a CBE scheme of Gentry. As stated in [14], it does not necessarily have to be "certificate updating". Two main entities involved are a certifier and a group of users. This model does not require a secure channel between the entities. A certificate based ring signature scheme is defined by six probabilistic, polynomial-time (PPT) algorithms:

- $\mathsf{GEN}_{\mathsf{IBS}}$: takes as input a security parameter 1^{k_1} and (optionally) the total number of time periods t. It returns a certifier's private key SK_C and public parameters $params$ that includes his public key PK_C, and the description of a string space S.
- $\mathsf{GEN}_{\mathsf{PKS}}$: takes as input a security parameter 1^{k_2} and (optionally) the total number of time periods t. It returns a user's secret key SK_U and his public key PK_U.
- Upd1: takes as input a certifier's private key SK_C, the public parameters $params$, a string $s \in S$, a user's public key PK_U at the start of time period i. It returns $Cert'_i$, which is sent to the user.
- Upd2: takes as input the public parameters $params$, $Cert'_i$ and (optionally) an old certificate $Cert_{i-1}$ at the start of time period i. It returns a new certificate $Cert_i$.
- RingSign: takes as input the public parameters $params$, a user's private key SK_U with his certificate $Cert_i$, a set of n users' public keys $\mathcal{P}$ in time period i (with $PK_U \in \mathcal{P}$) and a message $m \in M$. It outputs a signature $\sigma \in S$.
- Verify: takes as input the public parameters $params$, a set of n users' public keys $\mathcal{P}$ in time period i, a message $m \in M$ and a signature $\sigma \in S$. It returns either 1 (valid) or 0 (invalid).

CBRS is designed as a combination of public key signature (PKS), identity based signature (IBS) and ring signatures, where the signer needs both his personal secret key and a certificate from the CA to sign. The string s includes a message that the certifier signs and may be changed depending on the scheme.

Correctness. A certificate based ring signature scheme should satisfy the obvious correctness conditions (that a honestly signed ring signature should be verified as 1).

Unforgeability. As in CBE and CBS, we would like to model two different types of attacks by an uncertified user and by the certifier. Accordingly, we define two different games and the adversary chooses one game to play.

In Game 1, the adversary acts as an uncertified user. After proving knowledge of the secret key corresponding to its claimed public key, it can make RingSign and Cert queries.

In Game 2, the adversary acts as the certifier. After proving knowledge of the master secret corresponding to its claimed *params*, it can make RingSign and SKExtract queries. Let $P_{ID} = (i, PK_C, PK_U, U_{info})$ be a match for a user U's ID in IBC and call it by pseudo ID.

Game 1: The challenger runs $\mathsf{Gen}_{\mathsf{IBS}}(1^{k_1}, t)$, and gives *params* to the adversary. The adversary then issues Cert and RingSign queries. These queries are answered as follows:

- On certification query $\mathsf{Cert}(P_{ID}, SK_U)$, the challenger checks that SK_U is the secret key corresponding to PK_U in P_{ID}. If so, it runs Upd1 and returns $Cert'_i$; else returns $\perp$.
- On sign query $\mathsf{RingSign}(P_{ID}, SK_U, m, \mathcal{P})$, the challenger checks that SK_U is the secret key corresponding to PK_U in P_{ID}. If so, it generates $Cert_i$ and outputs a valid signature $\mathsf{RingSign}(m, params, Cert_i, SK_U, \mathcal{P})$; else it returns $\perp$.

The adversary outputs time i^*, a set of user's public key $\mathcal{P}^*$, a message m^* and a signature σ^*, such that:

- No user in $\mathcal{P}^*$ is the input to Cert oracle.
- (P_{ID}, m^*) are not equal to the inputs of any query to RingSign oracles, where $P_{ID} \in \mathcal{P}^*$.

The adversary wins the game if σ^* is a valid ring signature of m^* for i^* and $\mathcal{P}^*$.

Game 2: The challenger runs $\mathsf{Gen}_{\mathsf{PKS}}(1^{k_2}, t)$, and gives a set of users' public keys $\mathcal{P}$ and the master secret key SK_C to the adversary. The adversary then issues RingSign and SKExtract query.

- On sign query $\mathsf{RingSign}(\mathcal{P}', SK_C, params, m)$ with $\mathcal{P}'$ contains a user's public key $PK_U \in \mathcal{P}$, the challenger checks that SK_C is the secret key corresponding to PK_C in *params*. If so, it generates $Cert_i$ and outputs a valid signature $\mathsf{RingSign}(m, params, Cert_i, SK_U, \mathcal{P}')$; else returns $\perp$.
- On user secret key extract query $\mathsf{SKExtract}(PK_U)$, where $PK_U \in \mathcal{P}$, the challenger returns the corresponding secret key to the adversary.

The adversary outputs time i^*, a set of user's public key $\mathcal{P}^* \subseteq \mathcal{P}$, a message m^* and a signature σ^*, such that

- $(\mathcal{P}^*, m^*)$ is not equal to the inputs of any query to RingSign oracle.
- No user in $\mathcal{P}^*$ is the input to SKExtract oracle.

The adversary wins the game if σ^* is a valid signature of m^* for i^* and $\mathcal{P}^*$.

Definition 4. *A certificate based ring signature scheme is secure against existential forgery under adaptively chosen message and pseudo ID attacks if no PPT adversary has non-negligible advantage in either Game 1 or Game 2.*

Anonymity. An adversary should not be able to tell the identity of the signer with a probability larger than $1/n$, where n is the cardinality of the ring. We also allow the adversary to have the certifier's secret key. We have the following anonymity game:

1. The challenger runs $\mathsf{Gen}_{\mathsf{PKS}}(1^{k_2}, t)$, and gives a set of users' public keys $\mathcal{P}$ to the adversary.
2. The adversary can make a polynomial number of sign query as in Game 2 of unforgeability.
3. In the challenge phase, the adversary picks two public keys $PK_0^*, PK_1^* \in \mathcal{P}$, a message m^*, a set of n public keys $\mathcal{P}^*$ that include PK_0^*, PK_1^* and a certifier's secret key SK_C^* with *params* and gives them to the challenger. Then the adversary receives a challenge signature $\sigma^* = \mathsf{RingSign}(params, SK_b^*, Cert_{i,b}, \mathcal{P}^*, i)$, where $b \in \{0, 1\}$.
4. The adversary can make a polynomial number of sign query.
5. Finally $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

$\mathcal{A}$ wins the above game if $b = b'$. The advantage of A is defined as the probability that $\mathcal{A}$ wins minus $1/2$.

Definition 5 (Anonymity). *A certificate based ring signature scheme is anonymous if no PPT adversary has non-negligible advantage in winning the above game.*

3.2 Certificate Based Linkable Ring Signatures

The definition of certificate based linkable ring signatures is similar to that of CBRS in the previous section. We briefly mention the difference here.

A certificate based linkable ring signature scheme is defined by seven probabilistic, polynomial-time algorithms:

- The first six algorithms are the same as CBRS.
- Link: On input two signatures σ_1, σ_2, it outputs either link or unlink.

Correctness. Besides the verification correctness for CBRS, a certificate based linkable ring signature scheme should also satisfy the linking correctness. That means, if two signatures are linked, they must be generated from the same signer.

Unforgeability. It is the same as CBRS.

L-Anonymity. A crucial difference between Anonymity for ring signatures and L-Anonymity for linkable ring signatures is that in the latter, the adversary cannot query signatures of users PK_1^* and PK_2^* who appears in the challenge phase. The rationale is that if the adversary obtains signature of user i, it can tell if the challenge signature is generated by this user due to the linking property. The other definitions of the game are the same.

Linkability. An adversary should not be able to form two signatures with the same secret key without being linked by the Link protocol.

We have the following linkability game:

1. The challenger runs $\mathsf{Gen}_{\mathsf{PKS}}(1^{k_2}, t)$, and gives the master secret key SK_C and a set of users' public keys $\mathcal{P}$ to the adversary.
2. The adversary can make a polynomial number of sign query and secret key extract query as in Game 2 of unforgeability.
3. The adversary outputs signatures σ_i^* for messages m_i^*, certifiers' public key PK_{C_i} with $param_i$ and users' public keys set $\mathcal{P}_i^* \subseteq \mathcal{P}$ for $i \in \{0, 1\}$.

Let P' be the set of public key input to the secret key extract query. $\mathcal{A}$ wins the game if σ_0^* and σ_1^* are valid signatures, $\mathsf{Link}(\sigma_0^*, \sigma_1^*) = \mathsf{unlink}$ and $|(\mathcal{P}_0^* \cup \mathcal{P}_1^*) \cap P'| \leq 1$. The advantage of A is defined as the probability that $\mathcal{A}$ wins.

Definition 6 (Linkability). *A certificate based linkable ring signature is linkable if no PPT adversary has non-negligible advantage in winning the above game.*

4 The Proposed Scheme

4.1 Construction

Common Parameter. Let λ be the security parameter. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair with computable isomorphism ψ such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p of λ bits. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, be a cryptographic hash function. Also assume $\mathbb{G}_1$ be a group where DDH is intractable. Let g_0, g_1, g_2, u_0 be generators of $\mathbb{G}_1$, h_0, h_1, h_2 be generators of group $\mathbb{G}_2$ such that $\psi(h_i) = g_i$ for $i = 0, 1, 2$ and the relative discrete logarithm of the generators are unknown. This can be done by setting the generators to be output of a hash function of some publicly known seed.

$\mathsf{GEN}_{\mathrm{IBS}}$. The CA chooses a generator h of $\mathbb{G}_2$. Randomly select $q \in_R \mathbb{Z}_p^*$ and compute $q_i = h^{(q^i)}$ for $i = 1 \cdots t_{max}$, where t_{max} is the maximum number of accumulation. It also randomly selects $\gamma \in_R \mathbb{Z}_p^*$ and computes $w = {h_0}^{\gamma}$. The master secret is γ while the public parameters are $(H, \psi, \mathbb{G}_1, \mathbb{G}_2, p, g_0, g_1, g_2, h_0, h_1, h_2, u_0, h, q_1, \ldots, q_{t_{max}}, w)$.

$\mathsf{GEN}_{\mathrm{PKS}}$. The user randomly generates the private key $x_u \in_R \mathbb{Z}_p^*$ and computes the public key $PK_U = g_2^{x_u}$.

Upd1. The user with public information I_U which includes the time period i randomly selects $r' \in_R \mathbb{Z}_p^*$ and sends $C' = g_2^{x_u} g_1^{r'}$, along with the proof $\Pi_0 = SPK\{(r', x_u) : C' = g_2^{x_u} g_1^{r'}\}$ to CA. CA verifies that Π_0 is valid. If it is valid, it randomly selects $r'' \in_R \mathbb{Z}_p^*$ and computes

$$C = C' g_1^{r''} \qquad e = H(I_U) \qquad A = (g_0 C)^{\frac{1}{e+\gamma}}$$

and sends (A, e, r'') to the user. User computes $r = r' + r''$ and checks if $e(A, w h_0^e) = e(g_0 g_2^{x_u} g_1^r, h_0)$. It then stores the certificate $Cert_i := (A, e, r)$.

RingSign. Let $R = \left\{(I_{U_1}, PK_1), \cdots, (I_{U_n}, PK_n)\right\}$ be the set of ring signers. Let the index of the actual signer be u. For signing a message M, the actual signer uses his private key x_u and certificate (A, e, r) to compute

$$v = h^{\prod_{k=1}^{k=|\{R\}|}(q+H(I_{U_k}))} \qquad v_w = h^{\prod_{k=1,k\neq u}^{k=|\{R\}|}(q+H(I_{U_k}))}$$

$$SPK\Big\{(A, e, x_u, r, v_w) :$$

$$A^{e+\gamma} = g_0 g_2^{x_u} g_1^r \ \wedge\ v_w^{e+q} = v \ \wedge\ (\bigvee_{i=1}^{i=|R|} PK_i = g_2^{x_u})\Big\}(M)$$

The signature contains (v) and the transcript of the SPK.

RingSign (Link Version). Let $R = \left\{(I_{U_1}, PK_1), \cdots, (I_{U_n}, PK_n)\right\}$ be the set of ring signers. Let the index of the actual signer be u. For signing a message M, the actual signer uses his private key x_u and certificate (A, e, r) to compute

$$v = h^{\prod_{k=1}^{k=|\{R\}|}(q+H(I_{U_k}))} \qquad v_w = h^{\prod_{k=1,k\neq u}^{k=|\{R\}|}(q+H(I_{U_k}))} \qquad S = {u_0}^{x_u}$$

$$SPK\Big\{(A, e, x_u, r, v_w) :$$

$$A^{e+\gamma} = g_0 g_2^{x_u} g_1^r \ \wedge\ v_w^{e+q} = v \ \wedge\ S = {u_0}^{x_u} \ \wedge\ (\bigvee_{i=1}^{i=|R|} PK_i = g_2^{x_u})\Big\}(M)$$

This can be efficiently constructed as a discrete-log relation SPK, by randomly generating some variables $r_1, r_2 \in_R \mathbb{Z}_p^*$ and computing

$$A_1 = {g_1}^{r_1}, \quad A_2 = A{g_2}^{r_1}, \quad A_3 = g_1^{r_2}, \quad A_4 = v_w g_2^{r_2}, \quad \alpha = r_1 e, \quad \beta = r_2 e$$

$$SPK\Big\{(r_1, r_2, \alpha, \beta, e, x_u, r) : A_1 = g_1^{r_1} \ \wedge\ A_1^e = g_1^{\alpha} \ \wedge\ A_3 = g_1^{r_2} \ \wedge\ A_3^e = g_2^{\beta}$$

$$\wedge\ \frac{e(A_2, w)}{e(g_0, h_0)} = e(g_2, h_0)^{x_u} e(g_1, h_0)^r e(g_2, w)^{r_1} e(g_2, h_0)^{\alpha} e(A_2, h_0)^{-e} \ \wedge\ S = u_0^{x_u}$$

$$\wedge\ \frac{e(A_4, q_1)}{e(v, h)} = e(g_2, q_1)^{r_2} e(g_2, h)^{\beta} e(A_4, h)^{-e} \wedge\ (\bigvee_{i=1}^{i=|R|} PK_i = g_2^{x_u})\Big\}(M)$$

where $(\bigvee_{i=1}^{i=|R|} PK_i = g_2^{x_u})$ can be implemented easily by using standard 1-out-of-n proof of knowledge [12,1].

Note that S is the linkability tag and $v_w^{(q+H(I_u))} = v$. This can be turned into event-oriented version by replacing u_0 with $G(event)$ where G is some suitable hash function. The signature contains (v, S) and the transcript of the SPK (including (A_1, A_2, A_3, A_4)).

Verify. Verify the SPK.

4.2 Security Analysis

The Link Version can be regarded as a generalization of the non-Link Version. Thus we only show the security analysis of the Link Version and the security analysis of the non-Link Version is straightforward followed directly from the Link Version. In rest of this section, we refer "our scheme" as the proposed Certificate Based Linkable Ring Signature scheme.

Theorem 1. *Our scheme is unforgeable under the q-SDH assumption in the random oracle model.*

Lemma 1. *Suppose there exists a PPT $\mathcal{A}$ that could win in Game 1 Unforgeability, then there exists a simulator $\mathcal{S}$ to solve the q-SDH problem.*

Proof. (sketch.) $\mathcal{S}$ receives a q-SDH tuple $(g_1', g_2', {g_2'}^{x}, \ldots, {g_2'}^{x^q})$. $\mathcal{S}$ randomly picks $e_1, \ldots e_{q-1} \in \mathbb{Z}_p^*$ and lets $f(x) = \prod_{i=1}^{q-1}(x + e_i)$. If $x = -e_i$ for some i, $\mathcal{S}$ solves the q-SDH problem directly. Otherwise set $q_i = (h)^{q'^i}$ for some randomly generated $q' \in_R \mathbb{Z}_p^*$ and $h \in_R \mathbb{G}_2$, and compute $h_0 = {g_2'}^{f(x)}$, $w = {g_2'}^{xf(x)}$, $g_0 = \psi(h_0)$. $\mathcal{S}$ picks $e^*, a^*, k^* \in_R \mathbb{Z}_p^*$ and computes $h_1 = [(wh_0^{e^*})^{k^*} h_0^{-1}]^{1/a^*}$ and sets $g_1 = \psi(h_1)$. Randomly pick $\mu \in_R \mathbb{Z}_p^*$ and compute $h_2 = h_1^{\mu}$ and set $g_2 = \psi(h_2)$. $\mathcal{S}$ gives all parameters to $\mathcal{A}$.

Now each cert query is answered as follows. $\mathcal{S}$ computes:

$$\tilde{A}_i = g_0^{1/x+e_i} = \psi({g_2'}^{f(x)/x+e_i})$$

for $1 \leq i \leq q$. For the i-th query, $PK = g_1^{x_i}$ and $C' = g_1^{x_i} g_2^{r_i'}$, uses standard rewind technique to obtain x_i, r_i'. Set $H(I_u) = e_i$. randomly pick $r_i'' \in \mathbb{Z}_p^*$ and computes:

$$\begin{aligned}
A_i &= (g_0 C g_1^{r_i''})^{1/x+e_i} \\
&= (g_0^{1+x_i\mu+\frac{(r_i'+r_i'')[(e^*+x)k^*-1]}{a^*}})^{1/x+e_i} \\
&= \tilde{A}_i^{1+x_i\mu-\frac{(r_i'+r_i'')}{a^*}} g_0^{\frac{(r_i'+r_i'')k^*(e^*+x)}{a^*(e_i+x)}} \\
&= \tilde{A}_i^{(1+x_i\mu-\frac{(r_i'+r_i'')}{a^*})} \big(g_0^{\frac{(r_i'+r_i'')k^*}{a^*}}\big)^{(1-\frac{e_i-e^*}{e_i+x})} \\
&= \tilde{A}_i^{(1+x_i\mu-\frac{(r_i'+r_i'')}{a^*}-\frac{(r_i'+r_i'')k^*(e_i-e^*)}{a^*})} \big(g_0^{\frac{(r_i'+r_i'')k^*}{a^*}}\big)
\end{aligned}$$

$\mathcal{S}$ returns (A_i, e_i, r_i'') to $\mathcal{A}$.

One of the query, denoted as query *, is treated differently as follows. $(A_* = g_0^{k^*}, e^*, r_*'' = a^* - r_*')$ is returned by setting $H(I_*) = e^*$.

RingSign query can be handled using standard technique due to the HVZK property of the SPK.

Finally, $\mathcal{A}$ wins the game by returning a signature σ^* for message m^* and ring L^*. Due to the security of the accumulator from [22] (which is also reduced

to the q-SDH assumption) and the soundness of the SPK, $\mathcal{S}$ can rewind and extracts $(\hat{A}, \hat{e}, \hat{x}, \hat{r})$ such that $\hat{A}^{\hat{e}+x} = g_0 g_1^{\hat{r}} g_2^{\hat{x}}$ and $S = u_0^{\hat{x}}$ and $\hat{PK} = g_2^{\hat{x}}$ for some $\hat{PK} \in L^*$. We have three possibilities.

– Case 1: $\hat{e} \notin \{e_1, \cdots, e_{q-1}, e^*\}$. Then $\mathcal{S}$ computes:

$$\begin{aligned} A &= (g_0 g_1^{\hat{r}} g_2^{\hat{x}})^{1/\hat{e}+x} = (g_0^{1+\hat{x}\mu+\frac{\hat{r}[(e^*+x)k^*-1]}{a^*}})^{1/\hat{e}+x} \\ &= \big(g_0^{\frac{a^*+\hat{x}\mu a^*-\hat{r}}{a^*(\hat{e}+x)}}\big)\big[\big(g_0^{\frac{\hat{r}k^*}{a^*}}\big)^{(1-\frac{\hat{e}-e^*}{\hat{e}+x})}\big] \\ B &= \big(Ag_0^{-\frac{\hat{r}k^*}{a^*}}\big)^{\frac{a^*}{a^*+\hat{x}\mu a^*-\hat{r}-\hat{r}k^*(\hat{e}-e^*)}} \\ &= g_0^{1/\hat{e}+x} \end{aligned}$$

– Case 2: $\hat{e} \in \{e_1, \cdots, e_{q-1}, e^*\}$. With probability $1/q$, $\hat{e} = e^*$, $\mathcal{S}$ computes as in case 1:

$$\begin{aligned} A &= \big(g_0^{\frac{a^*+\hat{x}s\mu a^*-\hat{r}}{a^*(\hat{e}+x)}}\big)\big(g_0^{\frac{\hat{r}k^*}{a^*}}\big) \\ B &= \big(Ag_0^{-\frac{\hat{r}k^*}{a^*}}\big)^{\frac{a^*}{a^*+\hat{x}\mu a^*-\hat{r}}} \\ &= g_0^{1/\hat{e}+x} \end{aligned}$$

– Case 3: $\hat{e} = e_i$ and $\hat{A} = A_i$ for some i. We must have $\hat{A}^{\hat{e}+x} g_1^{-\hat{r}} g_2^{\hat{x}} = A_i^{e_i+x} g_1^{-r_i} g_2^{x_i}$, implies that $\hat{r} + \mu\hat{x} = r_i + \mu x_i$. If $\mathcal{S}$ simulates the game with $\mu = x$ and all other keys and parameters randomly chosen by $\mathcal{S}$, then $\mathcal{S}$ solves the discrete logarithm problem. Hence $\mathcal{S}$ can solve the q-SDH problem.

For Case 1 and Case 2, $\mathcal{S}$ can solve the q-SDH problem from B as follow. We have:

$$B = g_0^{1/\hat{e}+x} = (\psi(g_2')^{f(x)})^{1/\hat{e}+x} = {g_1'}^{f(x)/(x+\hat{e})} = {g_1'}^{\sum_{i=0}^{q-1} c_i x^i + c_{-1}/(x+\hat{e})}$$

where $c_{-1}, c_0, \ldots, c_{q-1}$ can be computed by $\mathcal{S}$ with $c_{-1} \neq 0$. Then $\mathcal{S}$ get:

$${g_1'}^{1/(x+\hat{e})} = \big(B \prod_{i=0}^{q-1} \psi({g_2'}^{x^i})^{-c_i}\big)^{1/c_{-1}}$$

which is the solution to the q-SDH problem together with $\hat{e}$.

Lemma 2. *Suppose there exists a PPT $\mathcal{A}$ that could win in Game 2 Unforgeability, then we there exists a simulator $\mathcal{S}$ to solve the DL problem.*

Proof. (sketch.) By the zero-knowledge property of the SPK, all the oracles can be simulated using standard techniques. Finally, $\mathcal{A}$ outputs a forged signature which involves proof-of-knowledge of the discrete logarithm of one of the public key generated by the simulator. The simulator can thus use standard rewind technique to solve the DL problem.

Theorem 2. *Our scheme is anonymous if the XDH assumption for bilinear pairing e holds (that is, the DDH assumption in $\mathbb{G}_1$ holds) in the random oracle model.*

Proof. (sketch.) By the zero-knowledge property of the SPK in **Sign**, the parameters computed inside the SPK protocol reveal no information about the signer identity. Therefore only the parameters (A_1, A_2, A_3, A_4, S) may reveal such information.

For the case of (A_1, A_2, A_3, A_4) leaking information, suppose we are given a DDH tuple $(g, g^x, g^y, R) \in \mathbb{G}_1^4$ to determine if $R = g^{xy}$. $\mathcal{S}$ picks the master secret key and sets $g_1 = g, g_2 = g^y$. He simulates all oracles correctly with the master secret key. Then at the challenge phase, $\mathcal{A}$ submits two users I_{U_0} and I_{U_1} to $\mathcal{S}$. $\mathcal{S}$ picks $b \in_R \{0, 1\}$ and $z \in_R \mathbb{Z}_p^*$. He sets:

$$A_1^* = g^x, \qquad A_2^* = A_b R, \qquad A_3^* = g^{xz}, \qquad A_4^* = v_{w_b} R^z$$

and simulates the rest of the signature using user I_{U_b}'s private key x_{U_b} and certificate (A_b, e_b, r_b), where $v_w = h^{\prod_{k=1, k \neq U_b}^{k=|\{R\}|} (q + H(I_{U_k}))}$. If $\mathcal{A}$ finally outputs $b' = b$, then $\mathcal{S}$ outputs 1 for the DDH problem. Otherwise, $\mathcal{S}$ outputs 0.

For the case of S leaking information, we can prove similarly by setting $g_1 = g, PK_{\tilde{u}} = g^x, u_0 = g^y$ for a randomly chosen $\tilde{u} \in_R \{1, \ldots, n\}$. $\mathcal{A}$ submits two users I_{U_0} and I_{U_1} to $\mathcal{S}$. If $U_0 \neq \tilde{u}$ and $U_1 \neq \tilde{u}$, $\mathcal{S}$ aborts. Otherwise let $U_b = \tilde{u}$ for $b \in \{0, 1\}$. $\mathcal{S}$ simulates the signature by controlling the random oracle and sets $S = R$ in the output signature. If $\mathcal{A}$ finally outputs $b' = b$, then $\mathcal{S}$ outputs 1 for the DDH problem. Otherwise, $\mathcal{S}$ outputs 0.

Theorem 3. *Our scheme possesses linkability under the DL assumption in the random oracle model.*

Proof. (sketch.) Oracle simulations are straight forward due to the zero-knowledge property of the SPK. $\mathcal{S}$ assigns the DL problem instance (g, y) to one of the public keys and sends it to $\mathcal{A}$, say PK^*. If $\mathcal{A}$ wins, it produces two signatures (σ_1, σ_2) that are unlink. $\mathcal{A}$ is allowed to query at most one private key among the users in the rings of those two signatures. If $\mathcal{A}$ queries corresponding private key of the PK^*, $\mathcal{S}$ aborts.

By the soundness of SPK, a valid signature implies that the discrete log of the linking tag S is equal to the discrete log of one of the public keys in the ring. Since σ_1 and σ_2 are unlink, the linking tag S_1 from σ_1 is not equal to the linking tag S_2 from σ_2. Thus with non-negligible probability, the discrete log of the linking tag of either signature is equal to the corresponding private key of PK^*.

Using standard rewinding technique, $\mathcal{S}$ can extract this value, which is the solution of the given DL problem instance.

5 Conclusion

In this paper, we propose a new notion called Certificate Based Ring Signature, which is the ring signature in certificate based cryptography setting. It solves

the problem of the complicated certificate chain verification which exists in traditional PKI. We also introduce the linkable version of the basic scheme, that allows the verifier to tell whether two given signatures are generated by the same signer. We presented security model and concrete construction for the two notions. We also proved their security in the random oracle model. We believe that these schemes are of practical interest in daily life application, such as e-voting.

Acknowledgements. We would like to thank all anonymous reviewers of ISPEC '07 for their valuable comments and suggestions.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT '02*, pages 415–432. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501.
2. S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In *ASIACRYPT '03*, volume 2894 of *LNCS*, pages 452–473. Springer-Verlag, 2003.
3. M. H. Au, S. S. M. Chow, W. Susilo, and P. P. Tsang. Short linkable ring signatures revisited. In *EuroPKI '06*, volume 4043 of *LNCS*, pages 101–115. Springer, 2006.
4. M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen. Constant-size ID-based linkable and revocable-iff-linked ring signature. In *INDOCRYPT '06*, volume 4329 of *LNCS*, pages 364–378. Springer-Verlag, 2006.
5. M. H. Au, J. K. Liu, T. H. Yuen, and D. S. Wong. ID-based ring signature scheme secure in the standard model. In *IWSEC '06*, volume 4266 of *LNCS*, pages 1–16. Springer-Verlag, 2006.
6. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC '06*, volume 3876 of *LNCS*, pages 60–79. Springer, 2006.
7. D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT '04*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.
8. D. Boneh, X. Boyen, and H. Shacham. Short group signatures using strong Diffie-Hellman. In *CRYPTO '04*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
9. D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO '01*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.
10. D. Chaum and E. V. Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.
11. S. S. M. Chow, J. K. Liu, V. K. Wei, and T. H. Yuen. Ring signatures without random oracles. In *AsiaCCS '06*, pages 297–302. ACM Press, 2006.
12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO '94*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.
13. S. Galbraith. Supersingular cures in cryptography. In *ASIACRYPT '01*, volume 2248 of *LNCS*, pages 495–513. Springer-Verlag, 2001.
14. C. Gentry. Certificate-based encryption and the certificate revocation problem. In *EUROCRYPT '03*, pages 272–293. Springer-Verlag, 2003. LNCS No. 2656.
15. B. G. Kang, J. H. Park, and S. G. Hahn. A certificate-based signature scheme. In *CR-RSA '04*, volume 2964 of *LNCS*, pages 99–111. Springer, 2004.

16. J. K. Liu, M. H. Au, and W. Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In *AsiaCCS 2007*, pages ???–??? ACM Press, 2007. Full Version of this paper can be found on eprint http://eprint.iacr.org/2006/373/.
17. J. K. Liu, W. Susilo, and D. S. Wong. Ring signature with designated linkability. In *IWSEC '06*, volume 4266 of *LNCS*, pages 104–119. Springer-Verlag, 2006.
18. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract). In *ACISP '04*, volume 3108 of *LNCS*, pages 325–335. Springer, 2004.
19. J. K. Liu and D. S. Wong. Linkable ring signatures: Security models and new schemes. In *ICCSA '05*, volume 3481 of *LNCS*, pages 614–623. Springer, 2005.
20. J. K. Liu and D. S. Wong. On The Security Models of (Threshold) Ring Signature Schemes. In *ICISC '04*, volume 3506 of *LNCS*, pages 204–217. Springer-Verlag, 2005.
21. N. McCullagh and P. Barreto. A new two-party identity-based authenticated key agreement. In *CT-RSA '04*, volume 3376 of *LNCS*, pages 262–274. Springer, 2004.
22. L. Nguyen. Accumulators from Bilinear Pairings and Applications. In *CT-RSA '05*, volume 3376 of *LNCS*, pages 275–292. Springer, 2005.
23. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT '01*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.
24. M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002. http://eprint.iacr.org/.
25. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO '84*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.
26. P. P. Tsang and V. K. Wei. Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation. In *ISPEC '05*, volume 3439 of *LNCS*, pages 48–60. Springer, 2005.
27. P. P. Tsang, V. K. Wei, T. K. Chan, M. H. Au, J. K. Liu, and D. S. Wong. Separable Linkable Threshold Ring Signatures. In *INDOCRYPT '04*, volume 3348 of *LNCS*, pages 384–398. Springer, 2004.

An Efficient ID-Based Verifiably Encrypted Signature Scheme Based on Hess's Scheme

Saeran Kwon[1,2] and Sang-Ho Lee[2]

[1] Dept. of Computer Science and Information, Daelim College, Korea
[2] Dept. of Computer Science and Engineering, Ewha Womans University, Korea
sranie@ewhain.net, shlee@ewha.ac.kr

Abstract. As many electronic items are exchanged over the Internet recently, the fair exchange problem becomes of a greater importance. When constructing fair exchange systems, verifiably encrypted signatures are usually used as a building block. Hence, we propose an efficient ID-based verifiably encrypted signature scheme based on Hess's signature scheme because it is known as a concise and secure signature scheme in ID-PKC. Our scheme does not need registrations between users and a trusted third party called an adjudicator, does not need zero-knowledge proof, and uses an optimized adjudicator who participates in the protocol only when problem occurs. Together with a formal model, we analyze security and efficiency of our scheme and show that it is more suitable for communication requirements than previous schemes of same kind.

1 Introduction

Nowadays, according as computer systems are growing explosively and their interconnections via networks are very well equipped, various business is conducted over the Internet under the distributed community. As many electronic items are exchanged over the Internet, for example, electronic checks, electronic airplane ticket, e-mail, electronic contract signing, and so on, fair exchange problem becomes of a greater importance which requires that during the exchange of items, either each party involved in the protocol gets the other's item, or neither of the parties does, even if the protocol is halted by any reason. In digital world, many researchers have been solving these problems in the frame of fair exchange protocol with a participation of a third party who prevents or resolves disputes that may occur between the communication parties [1,2,6,7,8,9,13,14,16,21,24,17].

Asokan et al. [1] introduced formally a fair exchange protocol relying on a trusted third party (TTP) in an optimistic way, that is, the TTP does not participate in the actual exchange protocol in normal cases, but is needed in only abnormal case where one player crashes or attempts to cheat. In such case, the TTP is called an off-line TTP. However, their protocol use highly interactive zero-knowledge proof.

Boneh et al. [6] first proposed a non-interactive verifiably encrypted signature, which is usually used as a building block when constructing optimistic fair

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 93–104, 2007.

exchange, via aggregation of short signatures called BLS scheme [5] based on the bilinear pairing on a gap Diffie-Hellman group (GDH group). Their scheme is very elegant and provably secure in the random oracle model, without registration between users and the TTP, and without zero-knowledge proof which needs many interactions. Their scheme requires special elliptic curve groups with bilinear pairing [11] to enable to solve Decision Diffie-Hellman problem (DDHP) where Computational Diffie-Hellman problem (CDHP) is hard.

In PKC 2004, Zhang et al. [22] presented a new short signature scheme from the bilinear pairing, which unlike BLS short signature scheme [5] uses general cryptographic hash function such as SHA-1 or MD5 and seems to be more efficient than BLS scheme. They also constructed a verifiably encrypted signature scheme [21] based on their above explained signature scheme [22], with the same computational time as the based signature scheme. Recently, Lu et al. [12] also gave an efficient verifiably encrypted signature scheme based on Waters's signature scheme [20], which is secure especially in the standard model without random oracles.

However, all works introduced above are in traditional certificate-based PKI settings where entity's public key is authenticated with its certification generated by the certification authority (CA). We know that the certificate-based PKI system needs many infrastructure in order to manage certifications and related problems. Additionally, though the scheme [12] are driven from the ID-based encryption scheme [20], the signature scheme converted from the ID-based encryption scheme [20] needs a certification for entity's public key.

In 1984, Shamir introduced the concept of ID-based cryptography in [19] in order to simplify certificate management in traditional public key scheme, and Boneh and Franklin presented the first fully practical and secure ID-based encryption scheme (IBE) in 2001 [4]. In the ID-based public key cryptography (ID-PKC), an entity's public key is directly derived from its public information such as name, e-mail address and IP address, and the corresponding private key is generated by a trusted party called a private key generator (PKG).

Saeednia et al. [17] proposed an ID-based optimistic fair exchange protocol based on the Guillou-Quisquater signature scheme that uses off-line TTP. Their scheme is set on RSA type frame. In 2005, Z. Zhang et al. [24], and Gu and Zhu [9] proposed ID-based verifiably encrypted signature schemes (ID-VESS), respectively. Z. Zhang et al. [24] gave an provably secure optimistic fair exchange protocol based on Bellare et al.'s [3] modified Sakai-Ogishi-Kasahara signature scheme [18] called SOK-IBS. Gu and Zhu [9] gave an ID-VESS based on Hess's signature scheme [10]. In both schemes, a semi-trusted off-line TTP is involved, no registration between users and the TTP is needed, and no zero-knowledge proofs are included, where semi-trusted TTP means it may misbehave on its own but will not conspire with either of the two parties involved [8]. Z. Zhang et al.'s scheme has a formal security model, and Gu and Zhu also gave a security proof in the security model specified by Boneh et al's in [6].

However, in [9] there exist some weakness and attacks which they did not consider in its security proof. In 2006, J. Zhang and Zou [23] presented a forgery

on Gu and Zhu's ID-VESS [9]. In addition, they also proposed a verifiably encrypted signature (VES) scheme the size of which is shorter than that of Gu and Zhu.

In this paper, we propose an efficient and concise ID-based verifiably encrypted signature scheme based on Hess's signature scheme which has the smaller size if compared with other ID-based verifiably encrypted signature schemes in ID-PKC, especially the VES scheme of J. Zhang and Zou [23], together with a small size of the based signature scheme i.e. Hess's signature scheme. Furthermore, our scheme does not need registrations between users and a trusted third party called an adjudicator, does not need zero-knowledge proof, and uses an optimized adjudicator. Together with a formal model, we analyze security and efficiency of our scheme and show that it is more suitable for communication requirements than previous schemes of same kind.

The rest of this paper is organized as follows. In section 2, we briefly explain the bilinear pairing, some hard problems and introduce definitions of verifiably encrypted signatures (VES). In section 3, we present our ID-based verifiably encrypted signature based on Hess's signature scheme. In section 4, we investigate efficiency of our scheme, and describe security of VESs in a security model and prove security of our VES scheme in the random oracle model. Finally, we conclude the paper with some remarks.

2 Preliminaries

In this section, we briefly review the basic concepts on bilinear pairings and a definition of Gap Diffie-Hellman group along with some related mathematical problems.

2.1 Bilinear Pairings

Let G_1 be an additive cyclic group of prime order q and G_2 be a multiplicative cyclic group of the same order. A bilinear pairing is a map $e : G_1 \times G_1 \longrightarrow G_2$ with the following properties:

1. Bilinear : $e(P+Q, R) = e(P, R) \cdot e(Q, R)$ and $e(P, Q+R) = e(P, Q) \cdot e(P, R)$ for all $P, Q, R \in G_1$.
2. Non-degenerate : $\exists P, Q, \in G_1$ such that $e(P, Q) \neq 1$.
3. Computability : There is an efficient algorithm to compute $e(R, S)$ for all $R, S \in G_1$.

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve.

2.2 Gap Diffie-Hellman (GDH) Groups and Some Problems

Let G be a cyclic group generated by P, whose order is a prime q. In general implementation [4], G will be the additive group of points on elliptic curve. Now, we describe some mathematical problems.

1. Discrete Logarithm Problem (DLP) : Given P, Q in a cyclic group G, find an integer n such that $Q = nP$.
2. Computational Diffie-Hellman Problem (CDHP) : Given (P, aP, bP) for some $a, b \in Z_q^*$, compute abP.
3. Inverse Computational Diffie-Hellman Problem (Inv-CDHP) : Given (P, aP) for some $a \in Z_q^*$, compute $a^{-1}P$.
4. Decisional Diffie-Hellman Problem (DDHP) : Given (P, aP, bP, cP) for some $a, b, c \in Z_q^*$, decide whether $c = ab$ in Z_q. (If so, (P, aP, bP, cP) is called a valid Diffie-Hellman tuple.)
5. Bilinear Diffie-Hellman Problem (BDHP) : Given (P, aP, bP, cP) for some $a, b, c \in Z_q^*$, compute $e(P, P)^{abc} \in G_2$.

We call G a GDH group if DDHP can be solved in polynomial time but no probabilistic algorithm can solve CDHP with non-negligible within polynomial time [5,15]. We consider additive groups in this paper as the GDH group, which can be found on super-singular elliptic curves or hyper-elliptic curves over finite field. For the details of GDH groups, refer to [4,11,15].

2.3 Definition of Verifiably Encrypted Signatures

We consider verifiably encrypted signature schemes in ID-based conditions. In ID-based public key cryptosystems (ID-PKC), a trust third authority called the Private Key Generator (PKG) issues private keys for users' public keys derived from publicly known information with its secret master key. A verifiably encrypted signature involves a signer, a verifier and a semi-trusted adjudicator TTP, where "semi-trusted" means an adjudicator TTP is honest when resolving dispute but can't make signatures without signers' consent because it does not know user's private key. The TTP chooses randomly its secret key SK and generates the corresponding public key PK. Then during `Setup` procedure, the TTP publishes its public key PK. However, since TTP's key pair (PK,SK) is not in ID-base setting, some modification by a cheating signer may happen with the intention of making resolution procedure unavailable. Thus a certain authentication for TTP's public key PK or alternatively some agreements between related parties must be set ahead of exchanging messages. We define a verifiably encrypted signature similarly to formal definitions proposed in [7,22,24].

Definition 1. *A verifiably encrypted signature scheme consists of seven polynomial-time algorithms namely* `Setup`*,* `Extract`*,* `Sign`*,* `Verify`*,* `VE_Sign`*,* `VE_Verify` *and* `Adjudication`*. Algorithms are described as follows:*

- **Setup:** System parameters `param`, the PKG's master key s and public key P_{pub} are established. Also adjudicator's secrete and public key pair (SK,PK) is set by the adjudicator. Then P_{pub} and PK are published.
- **Extract, Sign, Verify:** These algorithms are the same to those of ordinary ID-based signature schemes.
- **VE_Sign:** Given a private key D of a signer Q, a message m, and an adjudicator's public key PK, a signer computes a verifiably encrypted signature

σ' on m. More precisely, an algorithm `VESign`(m,D,PK) which is run by a signer Q outputs a verifiably encrypted signature σ' .

- **VE_Verify:** Given a public key Q, a message m, an adjudicator's public key PK and a verifiably encrypted signature σ', a verifier checks validity of σ' under the signer Q. More precisely, an algorithm `VEVerify`(m,σ',Q,PK) outputs 1 if accepted or 0 if rejected.
- **Adjudication:** This is a resolution algorithm run by the adjudicator in case a signer Q refuses to open her signature σ to a verifier, who in turn possesses a valid verifiably encrypted signature σ' on m. In this case, given an adjudicator's key pair (PK,SK), a public key Q and a verifiably encrypted signature σ' on m, the adjudicator extract a legal signature σ of m. Algorithm expression is `Adj`(m,σ',Q,SK)=σ for a valid σ'.

Validity of verifiably encrypted signatures requires that verifiably encrypted signatures verify and verifiably encrypted signatures to be resolved by the adjudicator also verify as ordinary signatures. That is, following equations should be satisfied.

$$\begin{aligned}
&\texttt{Veri}(m, \texttt{Sig}(m, D), Q) = 1,\\
&\texttt{VEVerify}(m, \texttt{VESign}(m, D, PK), Q, PK) = 1,\\
&\texttt{Veri}(m, \texttt{Adj}(m, \texttt{VESign}(m, D, PK), Q, SK), Q) = 1,
\end{aligned}$$

where `Sig`, `Veri` are procedures performing `Sign`, `Verify` algorithms on ordinary signatures, respectively.

3 Our ID-Based Verifiably Encrypted Signature Based on Hess's Scheme

3.1 Description

In this section, we will propose an efficient ID-based verifiably encrypted signature based on Hess's signature scheme. Our scheme consists of seven polynomial-time algorithms namely `Setup, Extract, Sign, Verify, VE_Sign, VE_Verify` and `Adjudication`. We briefly describe the scheme:

• **Setup:** Choose two groups $(G_1, +)$ and $(G_2, \cdot)$ of prime order q, a bilinear map $e : G_1 \times G_1 \longrightarrow G_2$ between them, and an arbitrary point $P \in G_1$ to be set a generator of G_1. Also pick three hash functions; $H_1 : \{0,1\}^* \longrightarrow G_1^*$ (extract a point on G_1 from an identity string), $H_2 : \{0,1\}^* \times G_2 \longrightarrow Z_q^*$ and $H_3 : \{0,1\}^* \times G_1 \times G_1 \longrightarrow G_1^*$. The trusted third party called Private Key Generator (PKG) picks its master key $s \in Z_q^*$ at random and computes its public key $P_{pub} = sP$. The adjudicator TTP randomly chooses its private key x and computes the corresponding public key $PK = xP$. The system parameters are $\Omega = (G_1, G_2, q, e, P, P_{pub}, PK, H_1, H_2, H_3)$.

• **Extract:** Given an identity $ID \in \{0,1\}^*$, the PKG computes $Q_{ID} = H_1(ID) \in G_1^*$, and $D_{ID} = sQ_{ID}$. The PKG uses this algorithm to extract the user's private key D_{ID}, and gives D_{ID} to the user by a secure channel.

• **Sign:** Given a private key D_{ID} and a message m, choose an arbitrary point $P_1 \in G_1^*$, pick $k \in Z_q^*$ at random and output a signature $\sigma = (r, V)$, where $r = e(P_1, P)^k$, $h = H_2(m, r)$, and $V = hD_{ID} + kP_1$.

• **Verify:** Given a signature $\sigma = (r, V)$ of an identity ID for a message m, compute $h = H_2(m, r)$, and accept the signature if and only if $r = e(V, P) \cdot e(H_1(ID), P_{pub})^{-h}$.

• **VE_Sign:** The ordinary signature to be encrypted by `VE_Sign` is one based on Hess's ID-based signature scheme-1 [10] described above. However, arbitrary point P_1 to be used when signing is set as an hash value on some points in G_1 together with user identity. Given a private key D_{ID}, a message $m \in \{0,1\}^*$ and adjudicator's public key PK, `VE_Sign` performs as follows:

- choose $k \in Z_q^*$ at random,
- compute $U = kP$, $P_1 = H_3(ID, U, PK)$ and $h = H_2(m, e(P_1, U))$,
- compute $V' = hD_{ID} + kP_1 + kPK$,
- output the verifiably encrypted signature $\sigma' = (V', U)$.

• **VE_Verify:** Given a verifiably encrypted signature $\sigma' = (V', U)$ of message m, compute $P_1 = H_3(ID, U, PK)$ and $h = H_2(m, e(P_1, U))$, and accept the signature if and only if

$$e(P, V') = e(Q_{ID}, P_{pub})^h \cdot e(U, P_1 + PK).$$

• **Adjudication:** Given the adjudicator's secret key x, and a valid verifiably encrypted signature $\sigma' = (V', U)$ of ID for message m, compute $V = V' - xU$, $r = e(P_1, U)$ along with $P_1 = H_3(ID, U, PK)$, and output $\sigma = (r, V)$.

In fact, since $xU = xkP = kPK$ and $r = e(P_1, U) = e(P_1, P)^k$, $\sigma = (r, V)$ is an ordinary signature by Hess's scheme-1.

Here, the set of algorithms **(Setup, Extract, Sign, Verify)** constitutes Hess's scheme-1 in [10].

4 Analysis

In this section, we investigate the efficiency of our VES scheme, describe security demanded in ID-based verifiably encrypted signature (VES) schemes, and finally analyze the security of our ID-based VES scheme.

4.1 Efficiency

We compare the efficiency of our VES scheme to that of recent proposed other VES schemes based on Hess's signature scheme in [23,9]. Since all these VES

schemes are based on Hess's signature scheme, we does not compare the efficiency about signing and verifying procedures. In the following table, we denote by M a scalar multiplication in $(G_1, +)$, and by $\hat{e}$ a computation of the pairing. Since the pairing operation takes the most running time, and secondly scalar multiplication does, we do not take any other operations into account without loss of generality. Some steps can be optimized if the same identities occurs frequently. We indicate optimized cases at the right side of general cases.

	Size	**VE_Sign**	**VE_Verify**	**Adjudication**
C.Gu [9]	$Z_q \times 3G_1$	$2\hat{e} + 5M$ / $5M$	$3\hat{e} + 1M$	$1\hat{e} + 1M$
J.Zhang [23]	$Z_q \times 2G_1$	$1\hat{e} + 4M$	$4\hat{e} + 2M$	$1\hat{e} + 1M$
Proposed	$2G_1$	$1\hat{e} + 4M$	$4\hat{e}$ / $3\hat{e}$	$1\hat{e} + 1M$
	$Z_q \times 2G_1$	$1\hat{e} + 3M$	$3\hat{e}$ / $2\hat{e}$	$1M$

In terms of communication requirements, our VES scheme is at least as efficient as the other two schemes. Our scheme has some trade-off. If we include the value $r = e(P_1, U) = e(P_1, P)^k$ in the VES, the running time of computation is reduced while the size of communication messages extends as in the above table. We conclude that our VES scheme can offer advantages in running time and communication requirements over the other schemes [23,9].

4.2 Security Model of VES

Boneh et al.[4] required three security properties of verifiably encrypted signatures in a certified public key system: validity, unforgeability, and opacity, where validity requires that verifiably encrypted signatures verify and verifiably encrypted signatures resolved by the adjudicator also verify as ordinary signatures, unforgeability requires that it is difficult to forger a valid VES, and opacity requires that extracting an ordinary signature from given verifiably encrypted signatures is hard.

Formal definitions of security on an VES were proposed in [7], which explicitly describe security against all parties involved in the protocols through the attack model and security goals. Also, Z. Zhang et al. [24] considered a similar security model on an VES in ID-based settings. We analyze the security of our VES scheme by security model in [7,24] and in addition one more attack model appended. Followings are aspects of security to be considered against each type of attackers:

Security against a signer with private key: A dishonest signer should not be able to produce a verifiably encrypted signature which are verified but can not be decrypted into an ordinary signature by the adjudicator. In this model, a malicious signer has the strongest power of extracting a private-key for a target-identity ID.

Security against signing without private key: An adversary without a private key for the target ID should not be able to produce a valid verifiably encrypted signature on any message. Of course, in this case, if assuming the original signature scheme encrypted as VES is existential unforgeable in polynomial time, the adjudicator can not extract an ordinary signature from the VES because it means to forge the original signature scheme.

Security against a verifier: A intentional verifier should not be able to transfer any verifiably encrypted signature he got from the signer into an ordinary signature, without explicitly asking the adjudicator to do it.

Security against the adjudicator: The adjudicator should not be able to produce a valid signature on a message m of a signer without explicitly asking the signer to generate a verifiably encrypted signature on m. In fact, a signer does not want the adjudicator to produce valid signatures which he did not intend on producing.

Notes. In fact, since a certain VES can be forged even though a signer does not have a private key as in [23], we consider such an attack model in the above.

4.3 Security Proof of Our Scheme

Theorem 1. *Under the formal model described above, our verifiably encrypted signature scheme based on Hess' signature scheme-1 is provably secure in the random oracle model, if assuming that the CDH problem is hard*

Proof. We shall show that the proposed VES scheme is secure against signer regardless of whether having a private key or not, verifier and adjudicator. Here, we denote by O_{Ext} an oracle simulating the procedure `Extract`, by O_{VSig} an oracle simulating the procedure `VE_Sign`, and by O_{Adj} the oracle simulating the adjudication procedure.

Security against a signer with private key: In this model, we can assume naturally the adversarial signer is given the access to the oracles O_{Ext}, O_{Adj}. Then the dishonest signer's goal is to produce a valid VES $\sigma' = (V', U)$ but from which the adjudicator can not extract an ordinary signature $\sigma = (r, V)$. However, if $\sigma' = (V', U)$ satisfies $e(P, V') = e(Q_{ID}, P_{pub})^h \cdot e(U, P_1 + PK)$ for $P_1 = H_3(ID, U, PK)$, $h = H_2(m, e(P_1, U))$, the adjudicator always can extract an Hess-type signature $\sigma = (\widetilde{r}, \widetilde{V})$ by $\widetilde{V} = V' - xU$ as the follows:

$$e(P, \widetilde{V}) = e(P, V')e(P, -xU) = e(P, V')e(PK, -U) = e(Q_{ID}, P_{pub})^h \cdot e(U, P_1).$$

When $\widetilde{r}$ is set as $e(U, P_1)$, we get from above equation the relation $\widetilde{r} = e(\widetilde{V}, P) \cdot e(Q_{ID}, P_{pub})^{-h}$ where $h = H_2(m, \widetilde{r})$. Hence the signer can not deny $\sigma = (\widetilde{r}, \widetilde{V})$.

Security against signing without private key: Suppose that an adversary without a private key for ID but having access to the oracle O_{VSig} constructs a valid VES $\sigma' = (\widetilde{V}', \widetilde{U})$ on message m under ID which has not been queried

to the oracle O_{VSig}. Then we have $e(P, \widetilde{V}') \cdot e(\widetilde{U}, -PK) = e(Q_{ID}, P_{pub})^h \cdot r$ for $r = e(\widetilde{U}, P_1)$ and $h = H_2(m, r)$. If the adjudicator's public key is set as $PK = zP$ for some $z \in {Z_q}^*$ by simulator, we have $e(P, \widetilde{V}' - z\widetilde{U}) = e(Q_{ID}, P_{pub})^h \cdot r$. Then $\widetilde{\sigma} = (r, \widetilde{V}' - z\widetilde{U})$ is an Hess-type signature. Since an Hess-type signature is proven secure in random oracle model, it is infeasible to construct a valid VES on m without a private key.

Security against a verifier: An advesarial verifier $\mathcal{A}$ can have access to oracles O_{Ext}, O_{VSig} and O_{Adj}. Then, if $\mathcal{A}$ forges a valid signature $\sigma = (r, V)$ for an entity with identity ID on message m, for which the corresponding VES $\sigma' = (V', U)$ has not been queried to O_{Adj}, and ID has not been queried to O_{Ext}, then we can construct a forger algorithm $\mathcal{F}$ to solve the CDH problem through making use of $\mathcal{A}$.

Let $X = aP, Y = bP \in G_1$ be a random instance of the CDH problem taken as input by $\mathcal{F}$. $\mathcal{F}$ takes $z \in {Z_q}^*$ at random and sets $PK = zY$, and then initializes $\mathcal{A}$ with $P_{pub} = X$ and PK as system's public keys. The algorithm $\mathcal{F}$ then starts queries such as those required by the above security model. Without loss of generality, we assume that, for any key extraction query or signature query involving an identity, an H_1 oracle query was previously issued for the same identity. Then these queries are answered by $\mathcal{F}$ as follows.

- ***Queries on oracle*** H_1**:** When an identity ID is submitted to the H_1 oracle, $\mathcal{F}$ flips a coin $T \in \{0, 1\}$ that yields 0 or 1. For random chosen $w \in Z_q^*$, if $T = 0$ then the hash value $H_1(ID)$ is defined as being $wP \in G_1$, or if $T = 1$ then $H_1(ID)$ is defined as being $wY \in G_1$. Both cases, $\mathcal{F}$ inserts a tuple (ID, w, T) in a list L_1 to keep track of the way it answered the query.
- ***Key extraction queries on oracle*** O_{Ext}**:** When $\mathcal{A}$ requests the private key of ID to oracle O_{Ext}, $\mathcal{F}$ recovers the corresponding (ID, w, T) from L_1. If $T = 1$, $\mathcal{F}$ outputs "failure" and halts. If $T = 0$, $\mathcal{F}$ returns wX because $H_1(ID)$ is defined as wP in previous queries on oracle H_1 and $wX = waP = awP = aH_1(ID) = D_{ID}$.
- ***Queries on oracle*** H_2**:** When a tuple (m, r) is submitted to H_2 oracle, $\mathcal{F}$ scans a list L_2 to check whether H_2 was already defined for that input. If it is, the previous defined value is returned. Otherwise, $\mathcal{F}$ picks a random $h \in Z_q^*$, stores the tuple (m, r, h) in L_2 and returns h as a hash value of $H_2(m, r)$ to $\mathcal{A}$.
- ***Queries on oracle*** H_3**:** When a tuple (ID, U, PK) is submitted to H_3 oracle, $\mathcal{F}$ scans a list L_3 to check whether H_3 was already defined for that input. If it is, the previous defined value is returned. Otherwise, $\mathcal{F}$ picks a random $v \in Z_q^*$, stores the tuple (ID, U, PK, v) in L_3 and returns $P_1 = vP$ as a hash value of $H_3(ID, U, PK)$ to $\mathcal{A}$.
- ***VES queries on oracle*** O_{VSig}**:** When $\mathcal{A}$ request the VES on a message m_i for an identity ID to oracle O_{VSig}, $\mathcal{F}$ recovers the previously defined value $Q_{ID} = H_1(ID)$ from L_1. (1) If $Q_{ID} = wP$, $\mathcal{F}$ randomly chooses a $k_i \in Z_q^*$ and set $U_i = k_i P$. Also $\mathcal{F}$ queries (ID, U_i, PK) to H_3 oracle and recovers the previously defined value $H_3(ID, U_i, PK) = v_i P = P_{1i}$ from L_3 if (ID, U_i, PK) already exists in L_3, otherwise, $\mathcal{F}$ picks a $v_i \in Z_q^*$ randomly

and set the hash value $H_3(ID, U_i, PK) = P_{1i}$ as v_iP. Then, after computing $e(k_iP, v_iP) = r_i$, $\mathcal{F}$ requests (m_i, r_i) to H_2 oracle and gets h_i for $H_2(m_i, r_i)$, and set $V_i' = h_iwX + k_iv_iP + k_iPK$. Then (U_i, V_i') is returned to $\mathcal{A}$. $\mathcal{A}$ accepts it as a valid VES because

$$e(P, V_i') = e(P, h_iwX)e(k_iP, v_iP + PK) = e(Q_{ID}, h_iP_{pub})e(U_i, P_{1i} + PK).$$

$\mathcal{F}$ keeps the tuple $(ID, m_i, r_i, U_i, V_i', k_i)$ in a list L_4.

(2) If $Q_{ID} = wY$, then $\mathcal{F}$ randomly chooses $t_i, k_i, h_i \in Z_q^*$, and then sets $V_i' = t_iP_{pub}, U_i = k_iP_{pub}$ and defines the hash value $H_3(ID, U_i, PK) = P_{1i}$ as $k_i^{-1}(t_iP - h_iQ_{ID}) - PK \in G_1$. Next,again $\mathcal{F}$ computes $e(U_i, P_{1i}) = r_i$ and defines the hash value $H_2(m_i, r_i)$ as h_i. If $H_3(ID, U_i, PK)$ or $H_2(m_i, r_i)$ is already defined, $\mathcal{F}$ outputs "failure" and halts. As in the case (1), $\mathcal{A}$ accepts it as a valid VES because

$$\begin{aligned}
& e(Q_{ID}, h_iP_{pub})e(U_i, P_{1i} + PK) \\
&= e(Q_{ID}, h_iP_{pub})e(U_i, k_i^{-1}(t_iP - h_iQ_{ID}) - PK + PK) \\
&= e(Q_{ID}, h_iP_{pub})e(k_iP_{pub}, k_i^{-1}(t_iP - h_iQ_{ID})) \\
&= e(Q_{ID}, h_iP_{pub})e(P_{pub}, t_iP - h_iQ_{ID}) \\
&= e(P_{pub}, t_iP) = e(P, t_iP_{pub}) \\
&= e(P, V_i').
\end{aligned}$$

$\mathcal{F}$ keeps the tuple $(ID, m_i, r_i, U_i, V_i', k_i)$ in L_4.

– ***Adjudication queries on oracle*** O_{Adj}**:** When $\mathcal{A}$ queries O_{Adj} on a VES $\sigma' = (m, U, V')$ for an identity ID, $\mathcal{F}$ first checks its validity and recovers the previously defined value $Q_{ID} = H_1(ID)$ from L_1. If $T = 1$, it halts and outputs "failure". Otherwise, $\mathcal{F}$ looks up the list L_4, find out k_i and answers to $\mathcal{A}$ with $V = V' - k_iPK$ if (m, U, V') is in the list L_4, otherwise halts. However if (U, V') is a valid VES, the probability that m has not been queried to O_{VSig} is negligible.

Suppose $\mathcal{A}$ outputs a fake signature $\sigma = (\widetilde{m}, \widetilde{r}, \tilde{V})$ for an identity $\widetilde{ID}$ eventually. $\mathcal{F}$ recovers the tuple $(\widetilde{ID}, \widetilde{w}, \widetilde{T})$ from L_1. If $\widetilde{T} = 0$, then $\mathcal{F}$ outputs "failure" and stops. Otherwise, it goes on and find out whether $(\widetilde{ID}, \widetilde{m}, \widetilde{r}, \cdot, \cdot, \cdot)$ appear in the list L_4 or not. If it does not appear in L_4, then $\mathcal{F}$ outputs "failure" and stops. In fact, in this attack we assume such condition as $\mathcal{A}$ queries oracle O_{VSig}, in order to prove that it is hard to extract an ordinary signature from a valid VES. In addition, the probability that it does not appear in L_4 is negligible, because it likely forge a Hess-type signature which is proven secure. Also the tuple should not have been queried to the oracle O_{VSig}. If it appear, $\mathcal{F}$ goes through L_4 to find out $\tilde{k}$ for which $U = \tilde{k}P_{pub}$. Since $\widetilde{r} = e(\widetilde{V}, P) \cdot e(Q_{\widetilde{ID}}, P_{pub})^{-h}$ and $e(P, V') = e(Q_{\widetilde{ID}}, P_{pub})^h \cdot \widetilde{r} \cdot e(U, PK)$, we have

$$e(P, V' - \widetilde{V}) = e(U, PK) = e(\tilde{k}P_{pub}, zY) = e(\tilde{k}X, zY).$$

As a result, we get the following solution for the CDH instance ($X = aP$, $Y = bP$)

$$abP = (\tilde{k}z)^{-1}(V' - \widetilde{V}).$$

Security against the adjudicator: We consider an adversarial adjudicator's attack. Adjudicator $\mathcal{A}$ has access to the oracle O_{VSig}, oracle H_3, and has its private key to enable to extract an ordinary signature (r, V) from any VES (U, V') . If $\mathcal{A}$ forges a valid signature $\sigma = (\widetilde{r}, \widetilde{V})$ on message $\widetilde{m}$, while $\widetilde{m}$ has not been queried to the oracle O_{VSig}, then we can construct a forger algorithm $\mathcal{F}$ to forge an Hess-type signature through making use of $\mathcal{A}$.

Here is how $\mathcal{F}$ invokes the adjudicator $\mathcal{A}$. For an O_{VSig}-query on message m, $\mathcal{F}$ chooses $t_i, k_i, h_i \in Z_q^*$ at random, and then sets $V_i' = t_i P_{pub}, U_i = k_i P_{pub}$ and defines the hash value $H_3(ID, U_i, PK) = P_{1i}$ as $k_i^{-1}(t_i P - h_i Q_{ID}) - PK \in G_1$. Next,again $\mathcal{F}$ computes $e(U_i, P_{1i}) = r_i$ and defines the hash value $H_2(m_i, r_i)$ as h_i. If $H_3(ID, U_i, PK)$ or $H_2(m_i, r_i)$ is already defined, $\mathcal{F}$ outputs "failure" and halts. Then adjudicator $\mathcal{A}$ accepts it as a valid VES. When $\mathcal{A}$ outputs a forgery $(\widetilde{m}, \widetilde{\sigma} = (\widetilde{r}, \widetilde{V}))$, where m has not queried to O_{VSig}, then $\mathcal{F}$ just outputs the same $(\widetilde{m}, \widetilde{\sigma})$ returned from $\mathcal{A}$. Hence, we see that $\mathcal{F}$ succeeds in generating a valid forgery if $\mathcal{A}$ succeeds. But since an Hess-type signature scheme is provably secure in the random oracle model under the CDHP assumption, the provability that $\mathcal{A}$ succeeds in above attack is negligible.

By above arguments, we can get that our scheme is provably secure under the hardness assumption of CDHP in the random oracle model. □

5 Conclusion

In this paper, we presented an efficient ID-based verifiably encrypted signature scheme based on Hess's signature scheme. We showed our scheme is more profitable for communication requirements due to smaller size than previous schemes of same kind. Furthermore, we analyzed our scheme's security in a random oracle model along with a formal model and compared its efficiency with other schemes.

References

1. N. Asokan, V. Shoup and M. Waidner, "Optimistic fair exchange of digital signatures," Advances in Cryptology - Proceedings of Eurocrypt'98, **LNCS 1403**, pp.591-606, Springer-Verlag, 1998; IEEE J. on Slected Areas in Communications, 18(4):593-610, 2000.
2. F. Bao, G. L. Wang, J. Y. Zhou and H. F. Zhu, "Analysis and Improvement of Micali's Fair Contract Signing Protocol," ACISP 2004, **LNCS 3108**, pp.176-187, Springer-Verlag, 2004.
3. M. Bellare, C. Namprempre and G. Neven, "Security Proofs for Identity-Based Identification and Signature Schemes," Advances in Cryptology - Proceedings of Eurocrypt'04, **LNCS 3027**, pp.268-286, Springer-Verlag, 2004.
4. D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," Advances in Cryptology - Crypto'01, **LNCS 2139**, pp.213-229, 2001.

5. D. Boneh, A. Lynn and H. Shacham, "Short signatures from the Weil pairing," Advances in Cryptology - Asiacrypt'01, **LNCS 2248**, pp.514-532, 2001.
6. D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," Advances in Cryptology - Eurocrypt'03, **LNCS 2656**, pp.416-432, 2003.
7. Y. Dodis and L. Reyzin, "Breaking and Repairing Optimistic Fair Exchange from PODC 2003," ACM Workshop on Digital Rights Management, pp.47-54, 2003.
8. M.K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party," In the 4th ACM Conference on Computer and Communications Security, pp.1-5, 1997.
9. Chunxiang Gu and Yuefei Zhu, "An ID-Based Verifiable Encrypted Signature Scheme Based on Hess's Scheme," CISC'05, **LNCS 3822**, pp.42-52, 2005.
10. F. Hess, "Efficient Identity based Signature Schemes based on Pairings," Proc. of SAC'02, **LNCS 2595**, pp.310-324, 2003.
11. A. Joux and K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups," Cryptology ePrint Archive, Report 2001/003, 2001. http://eprint.icar.org/.
12. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters, "Sequential Aggregate Signatures and Multisignatures Without Random Oracles," EUROCRYPT 2006, **LNCS 4004**, pp. 465-485, 2006.
13. S. Micali, "Certified e-mail with invisible post offices," presented at the 1997 RSA Security Conference, 1997.
14. S. Micali, "Simple and fast optimistic protocols for fair electronic exchange," In the 22th ACM Symposium on Principles of Dstributed Computing, pp.12-19, 2003.
15. T. Okamoto and D. Pointcheval, "The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes," Proc. of PKC'01, **LNCS 1992** pp.104-118, Springer-Verlag, 2001.
16. J.M. Park, E. Chong, H. Siegel and I. Ray, "Constructing fair exchange protocols for E-commerce via distributed computation of RSA signatures," In the 22th ACM Symposium on Principles of Dstributed Computing, pp.172-181, 2003.
17. Shahrokh Saeednia, Olivier Markowitch and Yves Roggeman, "Identity-based optimistic fair exchange with transparent signature recovery," 9th International Conference on Distributed Multimedia Systems (DMS 2003), pp.718-721, 2003.
18. R. Sakai, K. Ohgishi and M. Kasahara, "Cryptosystems based on pairing," In 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, 2000.
19. A. Shamir, "Identity Based Cryptosystems and Signature Schemes," Advances in Cryptology - Crypto'84. **LNCS 0196**, pp.47-53, 1984.
20. B. Waters, "Efficient identity-based encryption without random oracles," Eurocrypt 2005, **LNCS 3494**, pp. 114-127, Springer-Verlag, 2005.
21. F. Zhang, R. Safavi-Naini and W. Susilo, "Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairings," INDOCRYPT'03, **LNCS 2904**, pp.191-204, 2003.
22. F. Zhang, R. Safavi-Naini and W. Susulo, "An efficient signature scheme from bilinear pairing and its applications," Pulic Key Cryptography - PKC'04, **LNCS 2947**, pp.277-290, 2004.
23. Jianhong Zhang and Wei Zou, "A Robust Verifiably Encrypted Signature Scheme," EUC Workshops 2006, **LNCS 4097**, pp.731-740, 2006.
24. Zhenfeng Zhang, Dengguo Feng, Jing Xu and Yongbin Zhou, "Efficient ID-Based Optimistic Fair Exchange with Provable Security," ICICS 2005, **LNCS 3783**, pp.14-26, 2005.

On the Sequentiality of Three Optimal Structured Multisignature Schemes

Zuhua Shao

Department of Computer and Electronic Engineering
Zhejiang University of Science and Technology
No. 85, XueYuan Road, Hangzhou, Zhejiang
310012, P.R. of China
zhshao_98@yahoo.com

Abstract. A structured multisignature scheme is an order-sensitive multisignature scheme that allows participating signers to sign the same messages in compliance with a specified signing order. In this paper, we find that three optimal structured multisignature schemes cannot keep sequentiality since they cannot prevent partial signers producing a valid partial multisignature in a signing order different from the specified one. Hence, verifies cannot identify the real signing order only by checking verification equations. We guess that it is impossible to design any optimal structured multisignature scheme.

Keywords: Cryptanalysis, structured multisignature, sequentiality, order forge attack.

1 Introduction

A multisignature scheme allows a group of users to jointly sign a document such that any verifier is convinced that each member of the group participated in signing [1]. In some applications, however, co-signers in a signing group may associate with different role/position and therefore have different management liability and authorization capability. Hence multisignatures generated by the same group of co-signers with different signing order often imply different meaning. That is, not only the group of the signers but also its real signing order are important for verifiers. A structured multisignature scheme is an order-sensitive multisignature scheme that only allows participating signers to sign the same messages in compliance with a specified signing order.

A naive, or trivial, solution to this problem is as follows. The first signer generates the first signature on the message m using his private key to obtain (m, σ_1). The second signer generates the second signature on the message (m, σ_1) using his private key to obtain $((m, \sigma_1), \sigma_2)$. The signature $(\ldots((m, \sigma_1), \sigma_2), \ldots, \sigma_n)$ generated by the last signer is the structured multisignature for the message m. It is obvious that this simple scheme will meet the security requirements for structured multisignature schemes if the underlying signature scheme is secure. Its main drawback, however, is that both the

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 105 – 115, 2007.

signature length and the computational cost for verification grow linearly with the number of signers in the group. Harn [2] suggested two additional properties that need to be achieved in the design of an optimal multisignature scheme:

1. The size of a multisignature should be identical to the size of an individual signature.
2. The verification process of a multisignature should be almost identical to the verification process of an individual signature.

There are many structured multisignature schemes proposed in the literature [3 - 8]. Besides them, three schemes enjoy the optimal properties for multisignature schemes. The signature length and verification process are the same as those of the individual signature of the underlying signature scheme. Burmester et al. [9] proposed a structured ElGamal-type signature scheme in the PKC 2000 conference. Their scheme is efficient but requires two rounds of structured signing to generate a valid mulsignature. Later, however, Wu et al. [10] showed that the Burmester et al.'s structured multisignature scheme cannot prevent all participating signers producing a valid multisignature without following the specified signing order. Nevertheless, this is not serious problem from the view of the group-oriented cryptography introduced by Desmedt [11].

Lin et al. [12] proposed a structured multisignature scheme from the Gap Diffie-Hellman Group by extending the short signatures of Boneh et al. [13]. The proposed scheme can resolve signing structures of serial, parallel, and the mix of them.

Harn et al. [14] proposed two structured multisignature algorithms, one based on the RSA signature scheme [15] and the other on an ElGamal-type signature scheme [16]. The latter is more efficient than the Burmester et al.'s scheme since only one-round processing is required to follow the specified signing order.

In this paper, we find that these three optimal structured multisignature schemes cannot prevent partial signers producing a valid partial multisignature in a signing order different from the specified one. Hence, these optimal structured multisignature schemes are only order-free even if they are existential unforgeability under adaptively chosen-message attacks (EUF-CMA) [17].

2 Sequentiality of the Burmester et al.'s Structured Multisignature Scheme

In this section, we first briefly review the Burmester et al.'s structured multisignature scheme, and then propose an order forge attack.

2.1 Brief Review of the Burmester et al.'s Structured Multisignature

We assume that n signers $I_1, I_2, \ldots, I_n$ generate a structured signature on a message M according to order fixed beforehand.

Initialization: A trusted center generates two primes p, q with $q|(p-1)$. The public parameter g is an element of Z_p^* with the order q. $h(\ )$ is a public cryptographic hash function. Each signer I_i generates a random number a_i in Z_q^* as its secret key. The public key of the first signer is $y_1 = g^{a_1} \pmod p$. For $i \in \{2, \ldots, n\}$, the I_i's public key is computed sequentially as follows: $y_i = (y_{i-1} \cdot g)^{a_i} \pmod p$. Finally, the public key of the order group $(I_1, I_2, \ldots, I_n)$ is set to $y = y_n$.

Signature generation:

(1) Generation of r : Signers $I_1, I_2, \ldots, I_n$ generate r together as follows:

1. Signer I_1 selects k_1 in Z_q^* randomly and computes $r_1 = g^{k_1} \pmod p$. If gcd $(r_1, q) \neq 1$, then selects a new k_1 again.
2. For $i \in \{2, \ldots, n\}$, signer I_{i-1} sends r_{i-1} to I_i. Then I_i selects k_i in Z_q^* randomly and computes $r_i = r_{i-1}^{a_i} \cdot g^{k_i} \pmod p$. If $\gcd(r_i, q) \neq 1$, then selects a new k_i again.
3. $r = r_n$.

(2) Generation of s : Signers $I_1, I_2, \ldots, I_n$ generate s together as follows:

1. Signer I_1 computes $s_1 = a_1 + k_1 rh(r, M) \pmod q$.
2. For $i \in \{2, \ldots, n\}$, signer I_{i-1} sends s_{i-1} to I_i. Then I_i verifies that $g^{s_{i-1}} \stackrel{?}{=} y_{i-1} r_{i-1}^{rh(r,M)} \pmod p$, then computes $s_i = (s_{i-1} + 1)a_i + k_i rh(r, M) \pmod q$.
3. $s = s_n$

(3) The structured multisignature on the message M in the order $(I_1, I_2, \ldots, I_n)$ is given by (r, s).

Signature verification:

A structured multisignature (r, s) on the message M is verified by checking

$$g^s = y r^{rh(r,m)} \pmod p$$

Note that if the adversary attacks the key generation, the above scheme is not secure [18]. It is possible to modify the Burmester et al.'s scheme by requiring that each signer I_i provides a zero-knowledge proof of knowledge (ZKPoK) of the discrete log of y_i/y_{i-1} in the base g.

2.2 Order Forge Attack

Besides the order forge attack of Wu et al.'s attack launched by all participating signers, we propose an order forge attack launched by two signers. Suppose that the first two signers I_1 and I_2 want to generate partial signatures in the reversed order.

The first two signers I_1 and I_2 have secret keys a_1, a_2 and public keys $y_1 = g^{a_1} \pmod p$, $y_2 = g^{(a_1+1)a_2} \pmod p$ respectively. The partial signature (r_2, s_2) generated in the specified order by them must satisfy the verification equation $g^{s_2} \stackrel{?}{=} y_2 r_2^{rh(r,M)} \pmod p$.

Now we show that they can generate (r_2, s_2) in the reversed order. The second signer I_2 first selects k_1 in Z_q^* randomly, computes $r_1 = g^{k_1} \pmod p$ and sends r_1 to the first signer I_1. The first signer I_1 selects k_2 in Z_q^* randomly, computes $r_2 = r_1^{(a_1+1)} g^{k_2} \pmod p$ and sends r_2 to the third signer I_3. According to the specified order, other signers generate $r = r_n$.

After receiving r, the second signer I_2 computes his individual signature $s_1 = a_2 + k_1 rh(r, M) \pmod q$ and sends it to the first signer I_1.

The first signer I_1 can verify individual signature (r_1, s_1) of the second signer I_2 by checking

$$g^{s_1} = y_2^{(a_1+1)^{-1}} r_1^{rh(r,m)} \pmod p$$

Then the first signer I_1 computes $s_2 = s_1(a_1 + 1) + k_2 rh(r, M) \pmod q$.

Because $g^{s_1} = y_2^{(a_1+1)^{-1}} r_1^{rh(r,m)} \pmod p$ implies $g^{s_1(a_1+1)} = y_2 r_1^{(a_1+1)rh(r,m)} \pmod p$, (r_2, s_2) satisfies $g^{s_2} = y_2 r_1^{(a_1+1)rh(r,m)} g^{k_2 rh(r,m)} \pmod p$. Hence, $g^{s_2} = y_2 r_2^{rh(r,M)} \pmod p$ would be accepted by the succeeding signers $I_3, \ldots, I_n$. Then they generate s by following the signature generation algorithm.

Hence, the first two signers can generate partial signature (r_2, s_2) in the reversed order.

The attack can be generalized to any adjoining signers.

Therefore, verifies cannot identify in what order the structured mulitisignature (r, s) is generated only by checking the verification equation.

3 Sequentiality of the Harn et al.'s Structured Multisignature Scheme

In this section, we first briefly review the Harn et al.'s structured multisignature scheme, and then propose an order forge attack.

3.1 Brief Review of the Harn et al.'s Structured Multisignature Scheme Based on Elgamal-Type Signatures

Harn et al. [14] proposed two structured multisignature schemes. One scheme is based on the RSA signature scheme, which is not optimal. The other is based on an

ElGamal-type signature scheme, which is optimal. We propose an order forge attack against the optimal one.

3.1.1 Public Parameters

A large prime p, where $p = 2q + 1$ and q is also a prime, and a primitive element α of GF(p) are known to all signers.

3.1.2 Generating Individual and Group Private/Public Key Pairs

Initially all signers in the group $\{U_1, U_2, \ldots, U_t\}$ need to work together to generate their public keys y_i for $i = 1, 2, \ldots, t$ and their group public key y. Each signer randomly selects an odd private key x_i from $[1, q - 1]$. The last signer U_t first computes $y_t = \alpha^{x_t} \pmod p$ and sends it to U_{t-1}; U_{t-1} computes $y_{t-1} == y_t^{x_{t-1}} \pmod p$ and sends it to U_{t-2}; and so on. The group public key y is the public key of the first signer U_1 such that $y = y_1 = \alpha^{x_t x_{t-1} \ldots x_1} \pmod p$. The group private key is $x_t x_{t-1} \ldots x_1 \pmod{p - 1}$, which involves all signers' private keys. It is important to know that each signer U_i needs to prove to all other signers knowledge of the private key x_i before all other signers accepting the revealed value y_i as U_i 's public key. In case a digital certificate is associated with each public key, each signer needs to prove the knowledge of the private key to the certificate authority (CA) before obtaining a digital certificate from the CA.

3.1.3 Generating Individual Signatures

To sign an ElGamal-type signature, there is a pair of short-term private key and public key computed by each signer. This computation is independent of messages and can be precomputed. Hence this process does not need to follow the specified signing order. Each signer U_i randomly selects a short-term private key k_i from $[1, q - 1]$ and computes $r_i = y_{i+1}^{k_i} \pmod p$, where $y_{t+1} = \alpha$. After receiving all r_i for $i = 1, 2, \ldots, t$, each signer can compute $R = r_1 r_2 \ldots r_t \pmod p$.

For a given message m where m is the one-way hash of the message M, following the specified signing order $<U_1, U_2, \ldots, U_t>$, each signer U_i computes an individual signature s_i from the equation $x_i s_{i-1} = k_i R + s_i \pmod{p - 1}$, where $s_0 = m$; s_i is sent to the next signer.

3.1.4 Verifying Individual Signature

On receiving the individual signature s_i from the preceding signer U_i, the current signer U_{i+1} needs to verify that all preceding signers $<U_1, U_2, \ldots, U_i>$ have signed the message m properly. Since all preceding signer's individual signatures satisfy the following equations respectively:

$$y_1^{\,m} = r_1^{\,R} y_2^{\,s_1} \pmod p$$

$$y_2^{\,s_1} = r_2^{\,R} y_3^{\,s_2} \pmod p$$

$$\ldots$$

$$y_i^{\,s_{i-1}} = r_i^{\,R} y_{i+1}^{\,s_i} \pmod p$$

By multiplying all these equations together we obtain the following verification equation as

$$y_1^{\,m} = (r_1 r_2 \ldots r_i)^R \, y_{i+1}^{\,s_i} \pmod p$$

We claim that the signer U_{i+1} can use this verification equation to verify that all preceding signers $<U_1, U_2, \ldots, U_i>$ have signed the message properly.

3.1.5 Generating Group Signature

We claim that (R, s_t) is the multisignature of the message m.

3.1.6 Verifying Multisignature

Similarly, by multiplying all t equations together we obtain

$$y^m = y_1^{\,m} = R^R \alpha^{s_t} \pmod p$$

We claim that any verifier can access the group public key y to verify the multisignature (R, s_t) of the message m according to the verification equation above.

3.2 Order Forge Attack

Because the group public key is $y = y_1 = \alpha^{x_t x_{t-1} \ldots x_1} \pmod p$, with the knowledge of the group private key $x_t x_{t-1} \ldots x_1 \pmod{p-1}$, all participating signers can produce a valid multisignature without following the specified signing order. However, we would like to propose an order forge attack launched by two signers. Suppose that the first two signers U_1 and U_2 want to generate a partial multisignature in the reversed order.

The first two signers U_1 and U_2 have secret keys x_1, x_2 and public keys $y_1 = \alpha^{x_1 x_2 \ldots x_t} \pmod p$, $y_2 = \alpha^{x_2 x_3 \ldots x_t} \pmod p$ respectively. The partial signature (r_2, s_2) generated in the specified order by them needs to satisfy $y_1^{\,m} = (r_1 r_2)^R \, y_3^{\,s_2} \pmod p$.

Now we show that they can generate (r_1, r_2, s_2) in the reversed order. The second signer U_2 first randomly selects a short-term private key k_1 from $[1, q-1]$ and computes $r_1 = y_1^{\,k_1 x_2^{-1}} \pmod p = y_3^{\,k_1 x_1} \pmod p$. The first signer U_1 randomly selects a short-term private key k_2 from $[1, q-1]$ and computes $r_2 = y_3^{\,k_2} \pmod p$. They send (r_1, r_2) to the

other signers. After receiving all r_i for $i = 1, 2, \ldots, t$, each signer can compute $R = r_1 r_2 \ldots r_t \pmod p$.

For a given message m, where m is the one-way hash of the message M, the second signer U_2 computes s_1 such that $x_2 m = k_1 R + s_1 \pmod{p-1}$ and sends s_1 to the first signer U_1.

The first signer U_1 can verify the signature (r_1, s_1) of the second signer U_2 by checking the equation $y_2^{\,m} = r_1^{x_1^{-1} R} y_3^{\,s_1} \pmod p$.

Then U_1 computes s_2 such that $x_1 s_1 = k_2 R + s_2 \pmod{p-1}$ and sends s_2 to the third signer U_3. We claim that (r_1, r_2, s_2) satisfies the verification equation $y_1^{\,m} = (r_1 r_2)^R y_3^{\,s_2} \pmod p$.

From $x_2 m = k_1 R + s_1 \pmod{p-1}$, we have $x_1 x_2 m = k_1 x_1 R + x_1 s_1 \pmod{p-1}$. Adding it to $x_1 s_1 = k_2 R + s_2 \pmod{p-1}$, we have $x_1 x_2 m = (k_1 x_1 + k_2) R + s_2 \pmod{p-1}$. Thus

$$y_3^{\,x_1 x_2 m} = (y_3^{\,k_1 x_1} y_3^{\,k_2})^R y_3^{\,s_2} \pmod p$$

implies the verification equation $y_1^{\,m} = (r_1 r_2)^R y_3^{\,s_2} \pmod p$.

Therefore, the succeeding signers cannot find that the partial signature (r_1, r_2, s_2) are generated by first two signers U_1 and U_2 in the reversed order. Thus, they would generate the group signature (R, s_t) according to the signature generation algorithm.

4 Sequentiality of the Lin et al.'s Structured Multisignature Scheme

In this section, we first briefly review the Lin et al.'s structured multisignature scheme, and then propose an order forge attack.

4.1 Brief Review of the Lin et al.'s Structured Multisignature

The Lin et al.'s structured multisignature scheme is based on a Gap Diffie-Hellman Group G [13], where the Decisional Diffie-Hellman problem can be easily solved while the Computational Diffie-Hellman problem is computationally infeasible. We define the four-tuple of parameters (g, g^a, g^b, g^{ab}) that satisfies the Decisional Diffie-Hellman problem as a valid Diffie-Hellman tuple.

Assume that $Q = \{u_1, u_2, \ldots, u_n\}$ is a group of co-signers to generate structured multisignatures. We define the signing structure Λ for Q as a directed graph with all $u_i \in Q$ as real nodes and u_o and u_∞ as two dummy nodes (the starting node and the terminal node respectively). Moreover, we denote $prev(u_i)$ as the set of nodes directly precede to u_i in Λ. H is a full-domain one-way hash function, where H: $\{0, 1\}^* \rightarrow G\backslash\{1\}$.

The system parameters, q, g, G, H, etc. are as defined in the Gap Diffie-Hellman signature scheme proposed by Boneh et al. [13], where G is a multiplicative cyclic group with prime order q and g is a generator of G. The Lin et al.'s structured multisignature scheme consists of four algorithms: initialization, verification key generation, multisignature generation and multisignature verification stated as follows:

Initialization: For each signer u_i in the system, his secret key x_i is selected at random in Z_q^*, and his public key (certified by the Certificate authority) is computed by $y_i = g^{x_i}$, thus $y_i \in G$.

Verification key generation: Assume that each $u_i \in Q$ agrees upon a structured signing structure Λ, the signature verification key for each of them is generated as follows. Let $v_0 = 1$ be the multiplicative identity in G. Each $u_i \in Q$ generates his/her individual signature verification key v_i in accordance with Λ as:

$$v_i = (g \prod_{u_j \in prev(u_i)} v_j)^{x_i}$$

Finally, the verification key for Q is $v_Q = \prod_{u_j \in prev(u_\infty)} v_j$. Anyone can verify the authenticity of v_i by checking if $(g, y_i, (g \prod_{u_j \in prev(u_i)} v_j), v_i)$ is a valid Diffie-Hellman tuple.

Multisignature generation: For a message m to be signed, each $u_i \in Q$ performs the following steps in accordance with Λ:

Step 1. Compute $M = H(m)$.

Step 2. Verify σ_j from preceding signer u_j, for all $u_j \in prev(u_i)$, by checking if (g, v_j, M, σ_j) is a valid Diffie-Hellman tuple.

Step 3. Compute $\sigma_i = (M \prod_{u_j \in prev(u_i)} \sigma_j)^{x_i}$, where $\sigma_0 = 1$.

Finally, $\sigma_Q = \prod_{u_j \in prev(u_\infty)} \sigma_j$ serves as the structured multisignature generated by all $u_i \in Q$.

Multisignature verification: For a message m and the structured multisignature σ_Q generated by all $u_i \in Q$ in accordance with Λ, the verifier checks if $(g, v_Q, H(m), \sigma_Q)$ is a valid Diffie-Hellman tuple.

4.2 Order Forge Attack

We propose two order forge attacks to multisignatures with different structured signing structures.

4.2.1 The First Attack

Suppose that the first three signers u_1, u_2 and u_3 generate a multisignature in a serial order, that is, $prev(u_1) = \{u_0\}$, $prev(u_2) = \{u_1\}$ and $prev(u_3) = \{u_2\}$.

According to the verification key generation, the signing verification key for u_1 is $v_1 = g^{x_1}$, that for u_2 is $v_2 = g^{(x_1+1)x_2}$ and that for u_3 is $v_3 = g^{((x_1+1)x_2+1)x_3}$ respectively.

Now we show that they can generate partial multisignatures for any message m in the reversed order.

First the third signer u_3 computes $\sigma_3 = (H(m))^{x_3}$ and sends it to the second signer u_2 and the first signer u_1. They can verify it by checking if (g, y_3, $H(m)$, σ_3) is a valid Diffie-Hellman tuple. Then the second signer computes $\sigma_2 = \sigma_1^{x_2}$ and sends it to the first signer u_1. The first signer u_1 can verify it by checking if (g, y_2, σ_3, σ_2) is a valid Diffie-Hellman tuple. Finally the first signer u_1 computes $\sigma_1 = \sigma_2^{(x_1+1)}\sigma_3$. Because $\sigma_1 = (H(m))^{x_3x_2(x_1+1)+x_3} = (H(m))^{x_3x_2x_1+x_3x_2+x_3}$ and $v_3 = g^{((x_1+1)x_2+1)x_3} = g^{x_1x_2x_3+x_2x_3+x_3}$, ($g$, v_3, $H(m)$, σ_1) is a valid Diffie-Hellman tuple. The succeeding signers cannot find that this partial multisignature is generated in the reversed order. Thus, they would generate the structured multisignature σ_Q according to the multisignature generation algorithm.

4.2.2 The Second Attack

Suppose that the first four signers u_1, u_2, u_3 and u_4 have the following signing structure Λ: $prev(u_1) = prev(u_2) = \{u_0\}$, $prev(u_3) = \{u_1, u_2\}$ and $prev(u_4) = \{u_3\}$.

According to the verification key generation, the signing verification key for u_1 is $v_1 = g^{x_1}$, that for u_2 is $v_2 = g^{x_2}$, that for u_3 is $v_3 = g^{(x_1+x_2+1)x_3}$ and that for u_4 is $v_4 = g^{((x_1+x_2+1)x_3+1)x_4}$ respectively.

Now we show that they can generate σ_4 for any message m according to a different signing structure Λ': $prev(u_3) = \{u_0\}$, $prev(u_1) = prev(u_2) = \{u_3\}$, and $prev(u_4) = \{u_1, u_2, u_3\}$.

First the third signer u_3 computes $\sigma_3 = (H(m))^{x_3}$ and sends it to the first signer u_1, the second signer u_2 and the fourth signer u_4. They can verify it by checking if (g, y_3, $H(m)$, σ_3) is a valid Diffie-Hellman tuple. Then the first signer u_1 computes $\sigma_1 = \sigma_3^{x_1}$

sends it to the fourth signer u_4. The second signer computes $\sigma_2 = \sigma_3^{x_2}$ and sends it to the fourth signer u_4. The fourth signer u_4 can verify them by checking if (g, v_3, $H(m)$, $\sigma_1\sigma_2\sigma_3$) is a valid Diffie-Hellman tuple. Finally the fourth signer u_4 computes $\sigma_4 = (\sigma_1\sigma_2\sigma_3 H(m))^{x_4}$. Because $\sigma_3 = (H(m))^{x_3}$, $\sigma_1 = (H(m))^{x_1 x_3}$, $\sigma_2 = (H(m))^{x_3 x_2}$, $\sigma_1\sigma_2\sigma_3 = (H(m))^{x_3 x_1 + x_3 x_2 + x_3}$ and $v_3 = g^{(x_1+x_2+1)x_3}$, (g, v_3, $H(m)$, $\sigma_1\sigma_2\sigma_3$) is a valid Diffie-Hellman tuple and so is (g, v_4, $H(m)$, σ_4). The succeeding signers cannot find that this partial multisignature is generated according to different signing structure. Thus, they would generate the structured multisignature σ_Q according to the multisignature generation algorithm.

Readers can easily give methods to forge multisignures in accordance with different signing structures.

5 Conclusions

We have showed that the three optimal structured multisignature schemes cannot keep sequentiality since these multisignature schemes cannot prevent partial signers producing a valid partial multisignature without following the specified signing order.

Harn et al. [14] also proposed a structured signature scheme based on the RSA signature scheme. This scheme is not optimal. Verifiers need to verify structured signatures by using a verification step, one by one, in an order reversed to the specified signing order, by which they can confirm the real signing order.

In the three optimal structured multisignature schemes above, however, the verification process is almost identical to the verification process of an individual signature. Verifiers check structured signatures only by using a single verification step, once for all, which is independent of the real signing order.

Therefore, we think that this kind of verification process only can validate whether all the signers have signed messages, not can validate the real signing order. This is reason why we guess that it is impossible to design so called optimal structured multisignature schemes.

The authors of these three optimal structured multisignature schemes only discussed the security properties of their schemes heuristically. Hence this kind of ends for these three optimal structured multisignature schemes is natural. This fact shows that the formal models and the security proofs are especially important for any cryptographic scheme.

Acknowledgements. The author would like to thank the anonymous referees for their valuable comments and suggestions that improve the presentation of this paper. This material is based upon work funded by Zhejiang Provincial Natural Science Foundation of China under Grant No.Y104201.

References

1. K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures, NEC Research and Development, Vol.71, 1983, pp.1-8.
2. L. Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature, IEE Proc.-Comput. Digit. Tech., 141(5), 1994, pp.307-313.
3. T. Okamoto and K. Ohta. A digital multisignature scheme based on the Fiat-Shamir scheme, Advances in cryptology - Proceedings of Asiacrypt'91, LNCS 739, Springer-Verlag, 1993, pp.139-148.
4. K. R. P. H. Leung and L. C. K. Hui. Handling signature purposes in workflow systems, The Journal of System and Software, Vol.55, 2001, pp.245-259.
5. C. –Y. Lin, T.-C. Wu and J. –j. Hwang, ID-based structured Multisignature Schemes, advances in Network and distributed Systems security, Kluwer Academic Publisher (IFIP Conference Proceedings 206) Boston, 2001, pp45-59.
6. C. J. Mitchell and N. Hur. On the security of a structural proven signer ordering multisignature scheme, Proceedings of the IFIP TC6/TC11 SixthJoint Working Conference on Communications and Multimedia Security (CMS), 2002, Kluwer Academic Publisher (IFIP Conference Proceedings 228), Boston, 2002, pp.1- 8.
7. S. Mitomi and A. Miyaji. A Mulitisignature Scheme with Message Flexibility, Order Flexibility and Order Verifiability, Information security and privacy-Proceeding of ACISP 2000, LNCS 1841, Springer-Verlag, 2000, pp.298-312.
8. M. Tada. An order-specified one-cycle multisignature scheme secure against active attack, Proceedings of computer security Symposium 2001 (CSS 2001), pp.217-222.
9. M. Burmester, Y. Desmedt, H. Doi, M. Mambo, E. Okamoto, M. Tada, and Y. Yoshifuji. A Structured ElGamal-Type Multisignature Scheme, Advances in Cryptology-Proceedings of PKC'2000, LNCS 1751, Springer-Verlag, pp.466-482.
10. T. C. Wu, C. L. Hsu and C. Y. Lin. On the security of Burmester et al. Structured ElGamal-Type Multisignature Scheme, Proceeding of 11th Conf. On National Security, Tianan, Taiwan, 3-4 May 2001, pp.349-352.
11. Y. Desmelt. Society and group oriented cryptography: A new concept, Advances in Cryptology, Proceedings of Crypto'87, LNCS293, Springer, Berlin, 1988, pp. 120-127.
12. C.-Y. Lin T.-C. Wu, and F. Zhang. A structured multisignature scheme from the Gap Diffie-Hellman Group, eprint.iacr.org/2003/090.
13. D. Boneh, B. Lynn, and H. Shacham. Short signature from the Weil pairing, In Proceeding of Asiacrypt'01, LNCS 2248, Springer-Verlag, Berlin, 2001, pp.514-532.
14. L. harn, C.-y. Lin and T. –C. Wu. Structured multisignature algorithms, IEE. Proc-comput. Digit. Tech., 151(3), May 2004, pp.231-234.
15. R. L. Rivest, A. Shamir and L. Adelman. A method for obtaining digital signatures and public-key cryptosystem, Commun. ACM, 21, (2), 1978, pp.120-126.
16. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Tran. , 1985, IT-31, pp. 469-472.
17. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks, SIAM Journal on Computing, 17(2), pp.281-308, 1988.
18. Z. Dong and K. Chen. An attack on a multisignature scheme, eprint.iacr.org/2003/201.

Secure Feedback Service in Wireless Sensor Networks

Di Ma

University of California, Irvine
dma1@ics.edu.uci

Abstract. In many sensor network applications, it is critical for the base station to know the delivery (or execution) status of its broadcast messages (or commands). One straightforward way to do so is to let every sensor node send an authenticated acknowledgement (ACK) to the BS directly. However this naive solution is highly communication inefficient and may result in severe ACK implosion near the BS. In this paper, we propose a communication efficient scheme to provide secure feedback service in sensor networks. In our basic scheme, we use ACK aggregation to reduce the ACK traffic. Meanwhile we store audit information for each aggregation operation so that the BS can use the audit information to locate errors in the network. We further improve the basic scheme by constructing a balanced aggregation tree to reduce localization delay and using Bloom filters to reduce storage requirement in each sensor for storing audit information. We analyze the performance of the proposed scheme and show it achieves good bandwidth gain over the naive approach.

Keywords: Feedback Service, Sensor network, ACK aggregation, ACK implosion, MAC, Bloom Filter, Authentication.

1 Introduction

Wireless sensor networks (WSNs) enable data gathering from a vast geographical region and thus present unprecedented opportunities in a wide range of tracking and monitoring applications from both civilian and military domains. In a WSN, there exist hundreds or thousands of low-cost sensors which sense and collect data from the environment for some given tasks. The sensed data is then forwarded to the base station (BS) or sink in a hop-by-hop manner for further processing. A BS is a powerful control unit for WSNs which processes the data gathered by the sensors and operate the WSN by issuing commands/queries to the sensors. In many WSN applications, reliable data delivery is critical [18]. In these applications, the BS needs to know whether the sensors (the intended receivers) have received its broadcast/multicast messages or performed certain actions as it commanded.

For example, in a security application where image sensors are used to detect and identify the presence of critical targets [18], the BS may send one of the following three classes of messages, all of which have to delivered reliably to the sensors and thus message delivery status is wanted:: (i) Control-data. The BS may want to send a particular (say upgraded) image detection/processing software to the sensors which are configurable; (ii) Query-data. The BS may have to send a database of target images

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 116–128, 2007.

to the sensors, to help them in the image recognition triggered by subsequent queries; (iii) Query. The BS may send out one or more queries requesting information about the detection of a particular target. The sensors can then match targets detected with the pre-stored images, and respond accordingly.

Another example of explicit feedback is the request for expected action acknowledgement. In some wireless sensor and actuator networks, the BS may send (broadcast/multicast) commands to the sensors periodically and expect the sensors to perform certain actions. For security purpose, the BS expects ACKs from sensors. In this way, the BS knows the current network status which helps the BS to monitor the whole network.

A third example of explicit feedback is to detect the Denial-of-Message (DoM) attack [14]. Message broadcast is a fundamental communication primitive in most sensor networks. Sensors in an adversarial environment might be deprived of broadcast messages under the DoM attack. To detect the existence of DoM attack, every broadcast recipient, the sensor, is required to send an authenticated ACK to the BS. If the BS did not receive an ACK from a sensor node in a certain period, it will assume that this sensor node is under DoM attack.

The above examples clearly clarify both the necessity and importance of a secure feedback service in some WSN applications. A naive solution is for each sensor node to send its authenticated ACK to the BS directly. However, this may results in ACK implosion near the BS as one broadcast message may result in thousands of ACKs. When thousands of ACKs are forwarded to the BS at the same time, ACK implosion will occur near the BS. At the same time, transmission of thousands of ACKs is very expensive. According to [2], wireless transmission of a bit can require over 1000 times more energy than a single 32-bit computation. Communication inefficiency associated with the naive solution also shortens the lifetime of the whole WSN. In the naive approach there is a widely differing data communication load amongst sensors at different levels. Sensors closer to the BS have to send significantly larger amounts of data than their descendants and hence they use up their batteries and die sooner. When a level of nodes closer to the BS stop functioning, then the whole WSN stops functioning as well. Therefore, nodes would have to either be swapped around manually or replaced upon failure, both tasks being quite impractical when considering the number of nodes at the various levels.

To prevent ACK implosion, we propose to use ACK aggregation to reduce the ACK traffic. In an ACK aggregation scheme, multiple ACKs are compressed into a single aggregated tag and verifying this aggregated tag equals to verify all the component ACKs. Related cryptographic primitives such as *multisignature* [15,17,4,6] and *aggregate signature* [5,12,11] can be used to aggregate signatures. However these signature schemes are not suitable in the WSN setting because of their expensive public-key operations. In the symmetric key setting, Exclusive-Or (XOR) has been used in [14,1] as the aggregation function to aggregate ACKs from sensors. However, the security of XOR-based aggregation is not clear (we discuss this in detail in Section 3.2). We propose to use collision resistent hash function to do the aggregation and the security of our hash-based aggregation scheme is based on the collision-resistance property of the underlying hash function.

Although ACK aggregation reduces ACK traffic, it loses detailed network information. When errors happen, the final aggregate arriving at the BS only tells the BS the fact that there is something wrong. It cannot tell the BS how many nodes are in problem if the BS wants to know the severity of the problem. Furthermore WSNs are usually location-aware in nature. Specifically, if data is obtained without the corresponding location information, the data may be useless. Sensors positioned in different geographical areas may have different application importance. In this case the BS may want to know the distribution of nodes in problem, whether it is network wide or just a small area of network failing in function. The aggregate simply cannot answer these questions for the BS.

In this paper, we propose a communication efficient scheme to provide secure feedback service in WSNs. In our scheme, sensors in adjacent area are implicitly grouped together through construction of an aggregation tree among all the group members. ACKs from group members are aggregated together and the final aggregate from the root of the aggregation tree is sent to the BS. To provide detailed status information to the BS when the final aggregate value fails in the verification process, audit information for aggregation operation are stored in intermediate nodes in the aggregation tree. The audit information allow the BS to isolate faults in the network. Therefore our scheme consists of two main functioning parts: ACK aggregation and audit, fault localization. These two functioning parts work complementarily to provide detailed feedback service to the BS.

Our aggregation scheme is different from data aggregation [10, 8, 20, 7, 21]. In our scheme, authentication objects of messages, MACs, are aggregated while individual messages are kept intact. In a data aggregation scheme, individual data information is lost and the aggregate gives statistical information like MAX, MIN, AVG or Median to the verifier. Data aggregation schemes cannot be used in applications, for example, temperature developing pattern sensing in a nuclear reactor, which require the presentence of individual data record.

The remaining part of this paper is organized as follows: Section 2 formulates the system assumptions and security model. Section 3 presents our basic scheme and Section 4 gives improvements. Section 5 analyzes the proposed scheme. We conclude the paper in Section 6.

2 System Assumptions and Security Model

We assume a general WSN with n sensors and a single BS. We assume that all n nodes are alive. As sensors are unusually put in an unattended and adversary environment because of application nature, we require end-to-end security a must: a node's ACK should be able to uniquely identified by the BS and no one can cheat the node's status. In our scheme, we assume each sensor node has a unique identifier S and it generates its ACK in the form of a MAC with a unique secret symmetric key K_S shared with the BS. Thus we assume a secure key-management protocol is present to establish and manage the secure pair-wise key between each node and the BS.

We further assume the existence of a broadcast authentication primitive such that every node will receive the broadcast message from the BS in an authenticated fashion.

This broadcast authentication could, for example, be performed using μTESLA [19] or the one-time-signature based broadcast authentication scheme proposed in [9]. To prevent re-play attack, there is a nonce or a unique message ID associated with each broadcast message requesting acknowledgement. A sensor node replies the BS with its MAC after it receives the broadcast message. All ACKs are supposed to arrive at the BS within a fixed period.

Nothing can be made to tell whether an authenticated ACK is generated by a node or an attacker who obtains the secret key of the node. Instead we assume only a small portion of the total number of sensors can be compromised and their secret keys are exposed to the attacker. The attacker has a network-wide presence and can record, modify, inject or delete at will. The goal of an attacker is to report falsified information in order to hide the real network status information from the BS. The attacker may forge a bogus ACK of a sensor node which cannot respond because of being denied from receiving broadcast messages; the attacker may modify the ACK aggregate and pass the altered aggregate in the delivery network; the attacker may also want to drop some ACKs. A secure feedback service should be able to detect and locate such attacks.

3 The Basic Scheme

Our secure feedback scheme has four main phases: aggregation tree construction, ACK aggregation and audit, aggregate verification and fault localization.

Aggregation tree construction. In our scheme, ACK aggregation is performed over an aggregation tree rooted at the BS. An intermediate node in the aggregation tree acts as an aggregator which aggregate ACKs from its child nodes. The aggregation tree construction process is a process of topology discovery. It determines which nodes are aggregators and which are not. The construction is performed once in the system initialization stage and it can be performed again whenever needed to reflect any network changes caused by mobility of certain nodes, or to the addition or deletion of sensor nodes.

ACK aggregation and audit. After the authentication tree is constructed, a node knows whether it is an aggregator or not. If a node is an aggregator, it has the information about its child nodes. Once the ACKs from its child nodes arrive, an aggregator aggregates them with the ACK of its own using an aggregation function ($Agg(\cdot)$) and then passes the aggregate value to its parent node in the tree. At the same time, the aggregator stores audit information for this operation. The audit information will be used in the fault localization stage to prove that the aggregator does perform its aggregation task honestly and to help the BS to locate errors in lower levels.

Aggregate verification. Upon receiving an aggregate, the BS, knowing the topology of the aggregation tree, reconstructs the aggregation tree. It then compares the computed aggregate with the one it received. If the two values are equal, the verification is successful and a successful verification shows that all the nodes acknowledged and all the ACKs arrived at the BS. Otherwise, the verification fails and the BS conducts a fault localization process to isolate nodes who fails in acknowledgement.

Fault localization. When an aggregate fails in the verification process, the BS asks the aggregator who generats the value for its audit information. The audit information

allows the BS to check whether this aggregator does its aggregation task honestly. The audit information also helps the BS to find which components of the aggregate are not correct. Then the BS asks the nodes responsible for these incorrect components in the lower level of the authentication tree for audit information. This process repeats until the leaf level of the authentication tree is reached. In each round, the BS narrows down its localization search one level down toward the leaf nodes in the aggregation tree.

3.1 Aggregation Tree Construction

We use the method described in TaG [13] to construct an aggregation tree. In TaG, the BS broadcasts a message asking sensors to organize into a routing tree. It specifies its own ID and its level (or distance from the root, in this case, zero) in that message. Any sensor node without an assigned level that hears this message assigns its own level to be the level in the message plus one. It also chooses the sender of the message as its parent, through which it will route messages to the root. Each of these sensor nods then rebroadcasts the construction message, inserting their own IDs and levels. The construction message floods down the tree in this fashion, with each node rebroadcasting the message until all nodes have been assigned a level and a parent. The BS can initiate the construction process periodically so that to keep the latest network topology. To maintain stability in the network, parents are retained unless a child does not hear from them for some long period of time, at which point it selects a new parent using this same process. In Tag, sensors in adjacent area are implicitly grouped together through the construction process of the aggregation tree.

An example aggregation tree constructed using the method described above is shown in Figure 1. The tree is rooted at the BS. Intermediate nodes in the tree, such as nodes A, B, C, or E, act as aggregators. The tree construction process divides sensor nodes into groups and each group forms a sub-tree of the authentication tree. A group consists of sensor nodes which are geographically neighbored to each other. The root of the sub-tree performs the final aggregation operation and reports the result to the BS directly. For example, node A, M and N are the roots of such sub-trees and they report to the BS directly.

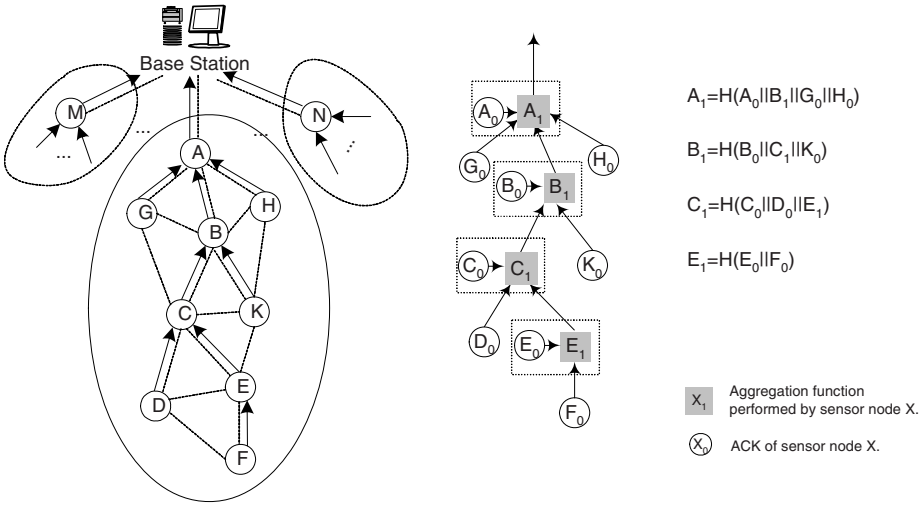
3.2 ACK Aggregation and Audit

Once the structure of the aggregation tree is known, ACKs can be aggregated when they are routed along the aggregation tree towards the root. Now we need to choose a proper aggregation function which an aggregator can use to combine ACKs coming from its child nodes.

Exclusive-Or (XOR) has been used in [14,1] as the aggregation function to aggregate ACKs from sensors. An aggregator, upon receiving the ACKs from its child nodes, computes the XOR of all these ACKs with its own ACK. If all nodes acknowledged, the final aggregate value arrived at the BS is in the form of

$$ACK_1 \oplus ACK_2 \oplus \cdots \oplus ACK_n$$

The XOR aggregation scheme is straightforward and independent on the structure of the aggregation tree. However, the security of the scheme is unclear as we do not know



(a) Example sensor network graph where arrows indicate aggregation direction.

(b) A partial aggregation tree rooted at node A.

Fig. 1. Construction of an ACK aggregation tree

how strong the weak collision resistance of Exclusive-Or is. Weak collision resistance of Exclusive-Or in this setting can be defined as:

> Given n messages $\{m_1, \cdots, m_n\}$, the probability of finding n' messages $\{m'_1, \cdots, m'_{n'}\}$ such that $m_1 \oplus \cdots \oplus m_n = m'_1 \oplus \cdots \oplus m'_{n'}$, $\{m_1, \cdots, m_n\} \neq \{m'_1, \cdots, m'_{n'}\}$ and $n' \leq n$.

We do not know how to calculate this probability. It is hard to evaluate the security strength of the XOR-based scheme. Instead we use a collision resistent hash function to aggregate ACKs. As in the XOR scheme, each sensor node generates its ACK in the form of a MAC: $ACK_S = MAC(K_S, N, S)$ where K_S is the secret key shared between the node S and the BS, N is the nonce or message ID unique to each broadcast message requesting acknowledgement and $MAC(\cdot)$ is any secure MAC scheme. The message a leaf node S sends to its parent is a tuple: (S, N, ACK_S). An aggregator, upon receiving the ACKs from its child nodes, concatenates all the ACKs with its own in order of IDs and then calculates the hash value over the concatenation. The hash value of the concatenation is the aggregate value which the aggregator further forwards to its parent in the tree. The message an aggregator A sends to its parent is a tuple (A, N, AGG_A) where AGG_A denotes the aggregate value. Figure 1 (b) shows the aggregation function performed by aggregators A, B, C and E and the corresponding aggregate values calculated by them. Meanwhile, every aggregator stores all the ACKs used in the aggregation as audit information. That is, A stores B_1, G_0, H_0 and its own ACK A_0; B stores B_0, C_1 and K_0; C stores C_0, D_0 and E_1; E stores E_0 and F_0.

We can view the process of ACK aggregation as constructing a hash tree distributively among all sensor nodes. To verify an aggregate, the BS reconstructs the sub-tree

by using the topology information and the secret keys shared with each sensor node in the group and compares its own calculated root value of the sub-tree with the aggregate value it received.

3.3 Fault Localization

When the verification of an aggregate failed, some nodes in the group do not acknowledge normally. BS initiates an iterative localization process among group members to locate nodes in problem. The localization begins with the root of a sub-tree who generates the final aggregate. The BS asks the root of the sub-tree for audit information which contains all the components in the final aggregate. The BS first checks whether the aggregator behaved honestly. It hashes the components extracted from the audit information and compares the hash result with the final aggregate it received. If the root is honest in aggregation, the two values will match. Next the BS checks whether a component matches the value it locally computed. An unmatched value tells the BS the fact that some nodes, who do not acknowledge normally, are in the sub-tree where this component value is the root value. Hence the BS will ask the node who generates the component for its audit information. By checking the audit information of an aggregator, the BS will narrow its search one level below this aggregator. That is, a test on the audit information of a level l node will let the BS locate some level $l + 1$ subgroups. The BS repeats this localization process one level further down to the leaf level until it reaches the leaf level.

We use an example to illustrate the concept of this iterative localization process. Suppose node F is under DoM attack. Its parent E either did not receive anything from it or received a forgery from the attacker. This error propagates in the aggregate values of E_1, B_1 and A_1. When the BS finds that A_1 does not match A_1^c which is locally computed by the BS (we use superscript ^{c} to denote that a value is a locally computed value by the BS), it asks A for audit information on A_0, B_1, G_0 and H_0. It computes the hash over these components: $A_1' = H(A_0||B_1||G_0||H_0)$. If $A_1' = A_1$, A performed aggregation honestly. Next the BS checks whether any component of A_1 matches the value it locally computed: $A_0 \stackrel{?}{=} A_0^c$, $B_1 \stackrel{?}{=} B_1^c$, $G_0 \stackrel{?}{=} G_0^c$,and $H_0 \stackrel{?}{=} H_0^c$. The value B_1 will not match B_1^c in this case. Then the BS asks B to send its audit information on B_0, C_1 and K_0. It computes $B_1' = H(B_0||C_1||K_0)$ and compares B_1' with B_1 it received from A to check whether A indeed aggregated ACKs from its child nodes. By inquiring on B's audit information, the BS finds that C_1 does not match C_1^c and asks C for audit information. Finally the BS finds that no ACK is from node F or $F_0 \neq F_0^c$ (F_0 is a forgery). That is, after four rounds of investigation the BS locates the error at sensor node F.

Security. An attacker is unable to make an individual ACK forgery without knowing the secret key because the underlying MAC scheme is unforgeable. An attacker is unable to make an aggregate forgery if at least one node is not compromised by the attacker. Otherwise we can either break the collision-resistance of the underlying hash function or the unforgeability of the MAC scheme.

In our scheme, an aggregator's role is just computing an aggregation operation. An aggregator is not necessarily to be trusted more by the BS than any other leaf nodes. An aggregator can not inject a forged ACK, modify an ACK or drop an ACK from its child nodes since any of these operations can be detected by the BS in the fault localization process. An aggregator can not provide irresponsible audit information to the BS as the BS will check its honesty in the localization process.

4 Improvements

4.1 Reducing Localization Delay

To locate an error originated from a leaf node in level l, the BS needs to perform l-rounds investigation process to locate the error. The height of the aggregation tree h determines the maximal delay in the localization process. For an aggregation tree constructed with TaG, the height of the aggregation tree depends on the node density and the total number of nodes n in the network. In a two-dimensional deployment area with a constant node density, the best bound on the diameter of the network is $O(\sqrt{n})$ if the network is regularly shaped. In irregular topologies the diameter of the network may be $\Omega(n)$. As the aggregation tree constructed by TaG may be arbitrarily unbalanced, the performance of the investigation process (in number of rounds) to locate errors in different levels varies dramatically. For example, to locate an error originated from F, the BS needs to perform four-rounds investigation process; on the other hand, to locate an error originated from H, the BS only needs to perform one round investigation process.

This motivates us to construct a more balanced aggregation tree to have localization delay bounded to $O(logn)$. We use the *delayed aggregation* idea described in [1] to construct a balanced aggregation tree. In the delayed aggregation approach, an aggregator only computes the aggregate of *some* (not *all* as in our basic scheme) of its child ACKs and passes the other ones to its parent for aggregation. Child nodes whose ACKs are not aggregated and passed to its parent's parent are moved one level up towards the root. Hence, delayed aggregation helps reducing the height of the aggregation tree. It trades off increased communication during aggregation phase in return for a more balanced aggregation tree with a height of $O(log_d n)$ where d is the degree of a node in the tree, and hence a better performance in the number of rounds in the fault localization phase.

Now we describe the algorithm for producing balanced aggregation trees with node degree of d. Our algorithm extends the algorithm which is used to construct a balanced binary tree [1]. We use the same strategy to construct a balanced d-ary aggregation tree: an aggregation operation is performed if and only if it results in a complete, d-ary aggregation tree. We assume each internal node keeps a small, fixed size list of neighborhood to maintain network topology.

We define a d-ary *forest* as a set of d-ary complete trees such that no d trees have the same height: $\{tree_1, tree_2, \cdots\}$. A tree in the forest is represented by its root node and the number of leaf nodes in the tree: $tree = (ID, count)$. A leaf node V in the aggregation tree generates a forest with a single node tree in the form $\{(V, 1)\}$ and sends it to its parent. An internal node S generates its own forest in the similar way. In

addition, S also receives forests from its children. S combines all the forests to form a new forest as follows. Suppose S wishes to combine q forests $\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_q$. Let $\mathcal{F} = \mathcal{F}_1 \bigcup \mathcal{F}_2 \bigcup \cdots \bigcup \mathcal{F}_q$. If there are less than d trees in $\mathcal{F}$, S simply passes $\mathcal{F}$ to its parent and no aggregation operation is performed by S. The height of a tree in $\mathcal{F}$ can be determined by inspecting the *count* field of the tree. Let h be the smallest height such that more than d trees in $\mathcal{F}$ has height h. S picks up d trees $T_1, T_2, \cdots, T_d$ of height h, and merge them into a tree of height $h+1$. The rule for S to pick up d trees of the same height to form a new tree is it always picks up trees with root nodes closer to itself. S obtains the adjacent nodes information from its neighborhood list. This process repeats until no two trees are the same height in $\mathcal{F}$. Then S forwards the new forest to its parent. Aggregation happens when the combination of trees happens.

Figure 2 shows an example of the process to construct a balanced 3-ary aggregation tree rooted at node A. When E receives a forest $\mathcal{F}_F = \{(F_0, 1)\}$ from its child F, as only two trees are in the combined forest $\mathcal{F}_E = \{(E_0, 1), (F_0, 1)\}$, E simply forwards $\mathcal{F}_E$ to its parent C. C picks three trees, $(C_0, 1)$, $(D_0, 1)$ and $(E_0, 1)$, of height 0 and whose roots are closer to it to form a tree of height 1: $(C_1, 3)$. C then forwards the forest $\mathcal{F}_C = \{(C_1, 3), (F_0, 1)\}$ to its parent B. B forms a new tree $(B_1, 3)$ of height 1 with trees $(B_0, 1)$, $(F_0, 1)$ and $(K_0, 1)$ and sends the forest $\mathcal{F}_B = \{(B_1, 3), (C_1, 3)\}$ to A. A combines trees $(A_0, 1)$, $(G_0, 1)$ and $(H_0, 1)$ to forms a tree $(A_1, 3)$ of height 1. A further combines trees $(A_1, 3)$, $(B_1, 3)$ and $(C_1, 3)$ to form a tree $(A_2, 9)$ of height 2.

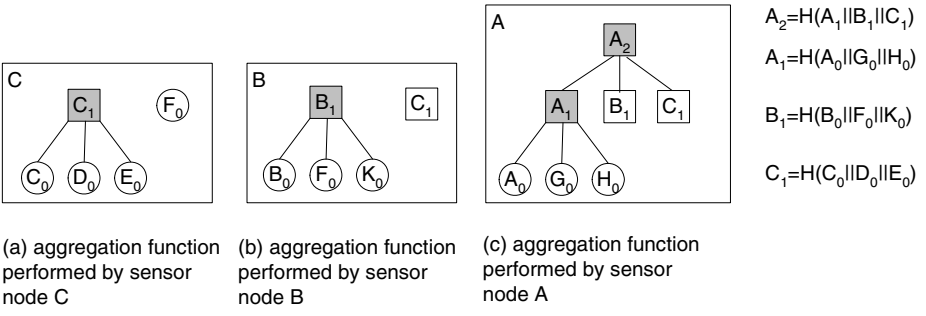


Fig. 2. Improved construction of balanced aggregation tree

In the new balanced aggregation tree rooted at A of height 2, E is no longer an aggregator. The three aggregators now are A, B and C. A performs two aggregation operations. It first aggregates ACKs from G, H with its own ACK to get aggregate value A_1. It then further aggregates A_1 with B_1 and C_1. Aggregate values generated by each aggregator are listed in Figure 2. Note in the new balanced aggregation tree, F is now in level 2 and its parent is B. Using the same example shown in Section 3.3, the BS now only needs to perform two rounds of the investigation process to locate an error originated from F.

4.2 Reducing Space for Storing Audit Information

An aggregator needs to store at least d ACKs as audit information (in Figure 2, A needs to store $2d$ ACKs as it performs aggregation twice). Instead of storing these ACKs separately, we can use a space-efficient data structure, Bloom filter, to store those ACKs.

The Bloom filter is conceived by Burton H. Bloom in 1970 [3]. It is a probabilistic data structure for testing membership of a set [16]. A Bloom filter for representing a set $S = \{s_1, s_2, \cdots, s_n\}$ of n elements is described as an array of m bits, initially all set to 0. A Bloom filter uses k independent hash functions $h_1, h_2, \cdots, h_k$ which map each a key value to one of the m positions uniformly. For each element $s \in S$, the bits $h_i(s)$ are set to 1 for all $1 \leq i \leq k$. A location can be set to 1 multiple times, but only the first change has an effect. To check if an item x is in S, we check whether all $h_i(x)$ are set to 1. If not, then clearly x is not a member of S. If all $h_i(x)$ are set to 1, we assume that x is in S. A Bloom filter may yield a false positive (but no false negative error), where it suggests that an element x is in S even though it is not. The probability of a false positive for an element not in the set, or the false positive rate p, is calculated as:

$$p = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-kn/m})^k$$

For a given m and n, the value of k that minimizes the probability is

$$k = \frac{m}{n} ln2 \approx 0.7\frac{m}{n}$$

which gives a probability of

$$p = (2^{-ln2})^{m/n} \approx 0.62^{m/n}$$

Bloom filters have a strong space advantage over other data structures for representing sets. A Bloom filter with 1% error and an optimal value of k, on the other hand, requires only about 9.6 bits per element - regardless of the size of the elements. This advantage comes partly from its compactness and partly from its probabilistic nature. If a 1% false positive rate seems too high, each time we add about 4.8 bits per element we decrease it by ten times.

We modify our fault localization process to accommodate the use of Bloom filters to store audit information. For each aggregation operation, an aggregator computes a Bloom filter with all its input ACKs as members. When the Bloom filter of node S at level l arrives at the BS, the BS tests the Bloom filter with its locally computed ACKs from S's child nodes at level $l+1$ (inputs to S's aggregation function). If the test shows that some ACKs are not in the Bloom filter, it tells the BS that some errors are in the lower groups with these level $l+1$ nodes as roots. In other words, a test on an level l Bloom filter let the BS narrow its search to some level $l+1$ subgroups. The BS repeats the investigation process one level further down to the leaf level until it reaches the leaf level. A Bloom filter has false positives. A false positive happens when the BS cannot find any missed ACK value and thus it cannot locate a subgroup in a lower level to proceed the localization process. In this case, the BS simply asks all the child nodes of that aggregator to send their individual ACKs directly to the BS. We will analyze how the false positive affect the performance of the localization process in Section 5.

5 Analysis

We compare the communication overhead of our secure feedback service scheme *SFSS* with the naive scheme *No-Agg* where individual ACKs are sent to the BS directly without aggregation. To simplify the measurements, we envision a WSN, with numerous sensor nodes and only one base station, organized into a balanced d-ary aggregation tree of height h. The number of bits sent by individual nodes and the overall bandwidth in the WSN are measured over this tree.

5.1 Per Node Communication Cost

For the *No-Agg* scheme, a leaf node only needs to transmit its own acknowledge message and the message length is $|M| = |ID| + |Nonce| + |ACK|$ in bits. All internal nodes need to forward the packets sent to them by their children, and the number of packets received grows exponentially as we move higher in the tree or closer to the root. The number of bits a node at level l needs to transfer is calculated as $\frac{d^{h-l+1}-1}{d^l-1} * |M|$.

For our *SFSS* scheme, in the ACK aggregation phase, each node forwards the same number of bits to its parent; in the fault localization phase, an aggregator involved in the investigation process needs to send its Bloom filter containing its audit information to the BS. Thus for leaf nodes and aggregators not involved in the investigation process, the number of bits they need to send is $|M|$; an aggregator involved in the investigation process needs to send $|M| + |BF|$ bits where $|BF|$ denotes the length of a Bloom filter in bits. Considering the situation when a false positive happens to an aggregator's Bloom filter, ACKs from its child nodes are required to be sent to the BS directly and this imposes $(d-1) * |M|$ communication load for level l nodes. Suppose there are n_l^{invt} nodes in level l involved in the investigation process, the average communication cost per node at level l is calculated as $(1 + \frac{n_l^{invt} * p * d}{d^l}) * |M| + \frac{n_l^{invt}}{d^l} * |BF|$. If we ignore the false positive of the Bloom filter such that $p = 0$, the per node cost is $|M| + \frac{n_l^{invt}}{d^l} * |ACK|$. If we further assume that all nodes acknowledged correctly such that $n_l^{invt} = 0$, we get the per node cost in this ideal situation as $|M|$.

From the analysis above, we know for the *No-Agg* approach there is a widely differing data communication load amongst sensors at different levels. The nodes closer to the sink die sooner as they have to send significantly larger amounts (d times) of data than their descendants. Instead ACK aggregation in *SFSS* mitigates the burden of higher level nodes on forwarding packets so that all the nodes roughly have the same transmission load. A level 1 nodes in *SFSS* only have a transmission load approximately d^h times less than that of a level 1 node in *No-Agg*.

5.2 Overall Communication Cost and Bandwidth Gain

The overall communication cost in the WSN is computed as the sum of the communication cost at each level in the tree. For the *No-Agg* scheme, the overall communication cost is calculated as: $C_{No-Agg} = \sum_{l=1}^{h} d^l * \frac{d^{h-l+1}-1}{d^l-1} * |PKT|$. For the *SFSS* scheme, the overall communication cost is calculated as: $C_{SFSS} = \sum_{l=1}^{h}(1 + \frac{n_l * p * d}{d^l}) * |M| + \frac{n_l}{d^l} * |BF|$. The bandwidth gain of *SFSS* over *No-Agg* is defined as $\frac{C_{No-Agg}}{C_{SFSS}}$.

Now we use concrete examples to show the bandwidth gain of *SFSS* over *No-Agg*. We consider a balanced aggregation tree of degree of 10 and height of 3, that is $d = 10$ and $h = 3$. Based on the same tree topology, we calculate the bandwidth gain with three different Bloom filter false positive rates: $p = 1\%, 0.1\%, 0.01\%$. When $p = 1\%$, the length of a bloom filter $|BF|$ containing 10 elements is 96-bits; when $p = 0.1\%$, $|BF| = 144$-bits; when $p = 0.01\%$, $|BF| = 192$-bits.

We consider three scenarios: when (1) all the nodes reply; (2) 90% of the nodes reply and (3) 70% of the nodes reply. The bandwidth cost in *Agg+Invt* is dependent on the distribution of nodes who fail in reply. Therefore we consider two extreme cases: (1) the worst case when errors occur in the maximum number groups; (2) the best case when errors occur in the minimum number of groups. This translates that in the 90% case, MAX(n_1^{invt})=10, MAX(n_2^{invt})=100 and MIN(n_1^{invt})=1, MIN(n_2^{invt})=1; in the 70% case, MAX(n_1^{invt})=10, MAX(n_2^{invt})=100 and MIN(n_1^{invt})=1, MIN(n_2^{invt})=3. We use 90%-MAX, 90%-MIN, 70%-MAX, and 70%-MIN to denote these different cases. Bandwidth gain with error distributions different from these two extreme distributions is in between the bandwidth gains of these two extremes. The calculation results are listed in Table 1.

Table 1. Bandwidth Gain (*Agg+Invt* vs. *No-Agg*.)

d	fp	m	100%	90%-MIN	90%-MAX	70%-MIN	70%-MAX
10	1%	96	2.7	2.43	2.14	1.88	1.66
10	0.1%	144	2.7	2.42	2.04	1.88	1.58
10	0.01%	192	2.7	2.42	1.93	1.87	1.5

From Table 1, we see *SFSS* has a good bandwidth gain over the naive scheme *No-Agg*. In the MAX distribution case, although decreasing false positive rate of the bloom filter decreases the probability for nodes to re-send their ACKs directly to the BS when their parent's Bloom filter has a false positive, it increases the length of all the bloom filters and hence leads to a decrease of the total bandwidth gain. In the MIN distribution case, false positive rate does not affect the bandwidth gain too much as in this case investigation traffic only contributes a very small percentage of the total bandwidth cost.

The results shown in this section are very encouraging since they confirm that aggregation is a useful technique for reducing the total bandwidth usage and therefore extend the overall lifetime of the network.

6 Conclusion

In this paper, we proposed a secure feedback scheme to provide secure feedback service in some sensor applications. In our basic scheme, ACKs are aggregated when they travel along the aggregation tree rooted at the BS. Each aggregator stores audit information which allows the BS to locate errors in the network. Improvements are made to reduce fault localization delay and storage overhead for audit information. Performance analysis showed that our scheme achieves good bandwidth gain over the naive scheme and enable a longer life of the WSN.

References

1. H. Chan, A. Perrig, and D. Song. “Secure hierarchical in-network aggregation in sensor networks”. *ACM CCS’06*, Nov. 2006.
2. K. Barr, and K. Asanovic. “Energy aware lossless data compression”. In *Proc. of MobiSys’03*. San Francisco, CA, May 2003.
3. B. Bloom. “Space/time tradeoffs in hash coding with allowable errors”. *Communication of the ACM*, 13(7):422-426, July 1970.
4. A. Boldyreva. “Efficient threshold signature, multisignature and blind signature scheme based on the gap-Diffe-Hellman-group signature shcme”. In *Proc. of PKC 2003*, LNCS 2567, pp. 31-46, Springer-Verlag, 2003.
5. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. “Aggregate and verifiably encrypted signatures from bilinear maps”. In *Proc. of Eurocrypt 2003*, LNCS 2656:416-432, May 2003.
6. C. Castelluccia, S. Jarecki, J. Kim, and G. Tsudik. “A robust multisignature scheme with application to acknowledgement aggregation”. In *Security in Communication Networks ’04*, 2004.
7. C. Castelluccia, E. Mykletun, and G. Tsudik. “Efficient aggregation of encrypted data in wireless networks”. In *Mobile and Ubiquitous Systems: Networking and Services MobiQuitous 2005*. July 2005.
8. L. Hu, and D. Evans. “Secure aggregation for wireless networks”. In *Workshop on Security and Assurance in Ad Hoc Networks*, 2003.
9. W. Lin, S. M. Chang, S. P. Shieh. “An efficient broadcast authentication scheme in wireless sensor networks”. *ASIACCS 2006.*
10. C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. “Impact of network density on data aggregation in wireless sensor networks”. In *ICDCS’02*, pp. 457-458. 2002.
11. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. “Sequential aggregate signatures and multisignatures without random oracles”. In *Prof. of Eurocrypt 2006*, May 2006.
12. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. “Sequential aggregate signatures from trapdoor permutations”. In *Proc. of Eurocrypt 2004*, LNCS 3027:245-254, Nov. 2001.
13. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. “TAG: a tiny aggregation services for ad-hoc sensor networks”. *SIGOPS Oper. Syst. Rev.* 36(SI):131-146, 2002.
14. J. McCune, E. Shi, A. Perrig, and M. Reiter. “Detection of Denial-of-Message Attacks on Sensor Network Broadbasts”. *IEEE Symposium on Security and Privacy*, 2005.
15. S. Micali, K. Ohta, and. L. Reyzin. “Accountable-subgroup multisignatures”. In *Proc. of CCS 2001*, pp. 234-54, 2001.
16. M. Mitzenmacher. “Compressed bloom filters”. *IEEE/ACM Trans. on Networks*. pp 613-620. Oct 2002.
17. T. Okamoto. “A digital multisignature scheme using bijective public-key cryptosystems”. *ACM Trans. Computer Systems*, 6(4):432-441, 1998.
18. S. Park, R. Vedantham, R. Sivakumar, and I. Akyildiz. “A scalable approach for reliable downstream data delivery in wireless sensor networks”. *MobiHoc 2004*. May 24-26, 2004.
19. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. “SPINS: Security protocols for sensor networks”. *Wirel. Netw.*, 8(5):521-534, 2002.
20. D. Wagner. “Resilient aggregation in sensor networks”. In *Workshops on Security of Ad Hoc and Sensor Networks.* 2004.
21. Y. Yang, X. Wang, S. Zhu, and G. Cao. “SDAP: a secure hop-by-hop data aggregation protocol for sensor networks”. In *ACM MOBIHOC’06*. May 2006.

An Economical Model for the Risk Evaluation of DoS Vulnerabilities in Cryptography Protocols

Zhen Cao, Zhi Guan, Zhong Chen, Jianbin Hu, and Liyong Tang

School of Electronics Engineering and Computer Science
Peking University, Beijing 100871, China
{caozhen, guanzhi, chen, hjbin, tangly}@infosec.pku.edu.cn

Abstract. Denial of Service (DoS) attacks are a virulent type of attack on the availability of networks' intended services and resources. Defense against DoS attacks has been built into the cryptography protocols intended for authentication and establishment of communications. However the cryptography protocols have their own vulnerability to DoS. Consequently it is desirable to provide a methodology to evaluate the cryptography protocols' resistance to DoS attacks. In this paper, we propose an economical model for the risk evaluation of Denial of Service vulnerabilities in cryptographical protocols. By characterizing the intruder capability with a probability model, our risk evaluation model specifies the Value at Risk (VaR) for the cryptography protocols. The Value at Risk does the very job answering the question that how much computing resources are expected to lose with a given level of confidence. The proposed model can help the common users to have a better knowledge of the protocols they are using, and in the meantime help designers to examine their designs and get clues to improve them. We validate the applicability and effectiveness of our risk evaluation model by applying it to analyze two related protocols.

1 Introduction

Recent years have witnessed the proliferation of network Denial of Service (DoS) attacks, which are any malicious actions that degrade networks' intended service to legitimate users. One of the most common and devastating types of DoS attack is the resource exhaustion attack, in which an attacker, by initiating a large number of instances of a protocol, causes the victim to deplete resource. These DoS attacks are usually carried out by intruders taking advantage of the vulnerabilities of the very protocol that intends to establish or authenticate communications following up. As a result, defenses against Denial of Service attacks should be built into the protocols themselves as much as possible.

Using cryptography protocols for authentication before communication establishment is a widely accepted mechanism defending against DoS attacks. However the cryptography protocols may introduce DoS vulnerability themselves, for some verifications involve resource consuming computations which may cause victims to be exhaustive of resources. Consequently, protocol designers should be

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 129–144, 2007.

on the alert for this problem and make their cryptography protocols invulnerable to DoS attacks as much as possible.

As the defense against DoS has been built into the protocols which have their own vulnerabilities, it is desirable to evaluate the resilience of cryptography protocols to DoS attacks. As a saying goes: if you can not evaluate it, you can not improve it. Not until we can express in numbers what we are speaking about is our knowledge of something becoming satisfactory and valuable. The cryptography protocol is no exception. Although formal methods [1] have achieved great success during the last two decades in evaluating whether or not cryptography protocols satisfy their security goals, little effort has been made for the risk evaluation of DoS vulnerabilities in cryptography protocols, the circumstance of which makes the very problem of DoS risk evaluation an important and urgent one.

Risk is the probability that a hazard will turn into a disaster. With protocol analysis, we can only find out potential vulnerabilities of certain protocol, namely that what kind of attackers under what kind of circumstance can intrude the system. But the notion of risk management urges the need for a framework analyzing the impact of those threats on system resources as well as a probability model analyzing the likelihood of those threat being realized. Fortunately, a cost-based framework for analyzing vulnerabilities to network DoS attacks in protocols was proposed by Meadows [2] [3]. This cost-based framework provided an excellent starting point for understanding and quantifying Denial of Service resilience in protocols. But without a probability model characterizing the likelihood of those threats turning into a realistic loss, we can never step towards the paradigm of risk evaluation and then risk management. In this paper, we present an economical model for the risk evaluation of Denial of Service vulnerabilities in cryptography protocols by introducing a probability model into Meadows' cost-based framework and adopting the model of Value at Risk (VaR) [4] which is widely used in financial literature.

The contributions of this paper can be summarized as follows:

- An economical model is specified for the risk evaluation of Denial of Service vulnerabilities in cryptography protocols. To our best knowledge, this is the first model for the risk evaluation of cryptography protocols;
- Value at Risk (VaR) for cryptography protocols is defined and utilized as risk evaluation method, which aggregates all the risks under DoS attacks into a single number. An algorithm for the computation of VaR in cryptography protocols is present as well;
- The applicability of our model is validated by applying it to the analysis of CCITT X.509 authentication protocol as well as its revised version. The evaluation result indicates the effectiveness as well as the validity of the proposed model.

The rest of this paper is organized as follows: in Section 2 we elaborate on the motivation of this paper. Then the system model of DoS risk evaluation is specified in Section 3, followed by a case study of our model to validate its

applicability in Section 4. In Section 5 we summarize related works on DoS analysis and evaluation, and finally we conclude this paper in Section 6.

2 Motivation

We will elaborate on the motivation of our work in this section before going on to introduce the proposed economical model. Although formal analysis of protocols have achieved great success during the last two decades, it has been carried out from experts' perspective, but fail to contribute much to the understanding of common protocol customers who have little knowledge of cryptography and information security. For instance, after a formal analysis tool is applied to the protocol, the experts can tell to some extend whether or not the protocol is vulnerable to certain kind of attacks, but for customers who has no idea of protocol analysis, it is really hard for them to understand whether it is proper to use this protocol. That is to say, common customers of the protocols do not benefit from the protocol analysis directly. That is not to say that protocol analysis is not helpful and necessary, but an implication that we should bridge the gap between the analysis result and common customers' comprehension.

Risk evaluation is the very methodology to bridge this gap. The concept of risk evaluation has undergone a long history. Bernstein [5] asserted that the revolutionary idea that defined the boundary between modern times and the past is the mastery of risk. Risk evaluation helps us to put into practice what is known as sustainable development, which means we can make a good living when what we have prepared for potential hazards is sufficient for the expected losses. For DoS attacks, risk evaluation of cryptography protocols can tell us how much is exposed to DoS attacks with a given level of confidence, and this evaluation result will help common customers a lot. As for cryptography protocols, let us settle down to reflect what is required from common customers' perspective. Common customers always want everything set up as simple as possible with the help of protocol analysis. For instance, they do not want to know what kind of attacks can be potentially dangerous, but they care about how much computation resource are exposed to these attacks; they do not want to understand why this protocol is better than others, but they are curious about how much one protocol will behave more secure and robust than the others. Risk evaluation of cryptography protocols meets this requirement quite well, for common customers can get to know the probability of harmful consequences or expected losses resulting from using the protocol, and they can easily compare different protocols with the risk evaluation results.

The same story goes for the companies. The boss who has been reading about derivatives which potentially suffer from losses, wants to know just how much market risk the company is taking in the company's foreign exchange. Many years passed before we can start the best answer that "the Value at Risk is . . . ". In a nutshell, subject to the simplifying assumptions used in its calculation, Value at Risk (VaR) aggregates all of the risks in the portfolio into a *single number* suitable for use in the boardroom, report to regulators, or disclosure

in an annual report. VaR answers the very question that "What is the most the entity can, with a 95% or 99% level of confidence, expect to lose in dollars over the next month". Value at Risk has been called the "new science for risk management", and it has achieved great success in financial risk evaluation and has been mandated by the Basel Committee on Banking Supervision [6].

The success of Value at Risk in financial community has inspired many researches in applying it to the risk management of computer and networking systems [7] [8]. This paper is dedicated to propose an economical model based on Value at Risk to evaluate the risk of Denial of Service in cryptography protocols. The evaluation result will benefit both common users and designers. With the proposed model, common users can be aware of the risk of their protocols: what is expected to lose in their computing resources or anything else with a certain level of confidence. Taking advantage of this evaluation model, protocol designers and analysts can evaluate the resilience of their protocols to Denial of Service attacks, and get clues as to how to make their designs better.

3 System Model

In this section, we will present our economical model for the risk evaluation of cryptography protocols. The specification used in our analysis is specified first, after which the risk evaluation model based on Value at Risk is present.

3.1 Protocol Specification

The specifications used in our model is the same as what is specified in [3].

The popular Alice-and-Bob specification of cryptography protocol will be used across the whole paper.

Definition 1. *An Alice-and-Bob specification is a sequence of statements of the form* $A \rightarrow B : M$ *where* A *and* B *are processes and* M *is a message.*

Annotated Alice-and-Bob specification style, which is the basis of high level protocol description languages such as CAPSL [9] and Casper [10], includes message processing steps at both the protocol initiator and responder, as defined below.

Definition 2. *An annotated Alice-and-Bob specification is a sequence of statements of the form* $A \rightarrow B : T_1, \ldots, T_k \parallel M \parallel O_1, \ldots, O_n$

The sequence $T_1, \ldots, T_k$ represents the sequence of operations performed by A in producing M, while the sequence $O_1, \ldots, O_n$ represents the sequence of operations performed by B in processing and verifying M. More closely study of each line leads to the definition of event.

Definition 3. *Let* $\boldsymbol{L} = A \rightarrow B : T_1, \ldots, T_k \parallel M \parallel O_1, \ldots, O_n$ *be a line in an annotated Alice-and-Bob specification. We say that* X *is an event occurring in* $\boldsymbol{L}$ *if*

1. *X is one of the T_i or O_i, or;*
2. *X is 'A sends M to B' or 'B receives M from A'.*

There are two kinds of events: normal events and verification events. Normal events can occur at either sender or receiver, and have only one outcome: success, while verification events occur only at the receiver, and can come out with success or failure. To describe B's intention to proceed with the protocol after successfully verifying a message, an *accept event* is attached to the end of each line. Section 4 gives an example of this specification.

3.2 Intruder Capability and Its Probability Distribution

Definition 4. *We define an **intruder action** to be an event engaged in by an intruder that affects messages received by legitimate participants in a protocol. We define an **intruder capability** to be a set of actions available to an intruder, partially ordered by set inclusion.*

Examples of intruder capability would include such cases as an intruder who could send messages but not read messages that were not addressed to it, an intruder who can impersonalize as the other entities, an intruder who can generate valid time stamp for establishing communications, and an intruder who can generate valid signatures of legitimate participants, and so forth.

Intruder capability characterizes the intruders' ability to persuade one participant of the protocol to consume resources participating in the protocol. Because different kinds of intruders distribute with different probabilities, we are going to introduce the definition of *Intruder Capability Probability Distribution Function* which characterize the probability of intruders with different capabilities.

Definition 5. *Let θ be an **Intrude Capability Probability Distribution Function** from the set of intruder capability to an probability value within* $[0, 1]$.

This function describes the probability distribution of intruders' capability. We take it for granted that the more powerful the capability is, the less possible that intruders will own the capability. For example, if we can divide the intruder capabilities into n different sets, and the probability of intruders who have capability IC_i is p_i, i.e., $\theta(IC_i) = p_i, P(intruder \in ICS_i) = p_i$, for $i = 1, \ldots, n$. Assume that the n events of owning capability $IC_1, \ldots, IC_n$ are all independent, then the probability of intruders who have only capabilities of $IC_1, \ldots, IC_k$ is $p_1 p_2 \ldots p_k (1 - p_{k+1}) \ldots (1 - p_n)$. (where ICS_i denotes the set including all the intruders that own capability IC_i)

Setting up the probability model of intruder capability is a crucial process for our risk evaluation model. As attackers with different capabilities can cause the victim to stop at different steps of the protocol and thus consuming different levels of computation resource under DoS attacks, we will arrive at the definition of the probability distribution of DoS loss after the cost set and the protocol engagement cost is defined.

3.3 Cost Set and Protocol Engagement Cost

In this subsection, we will study into the cost of participating in the cryptography protocol which includes the cost of event execution, the cost of message acceptance and the cost of protocol engagement.

Definition 6. *A **cost set C** is a partially ordered set with partial order $<$ together with a function $+$ from $C \times C$ to C such that $+$ is associative and commutative, and $x + y \geq max(x, y)$, along with an zero element 0 such that $x = 0 + x = x + 0$, for all x in **C**.*

An examples of cost set would be the set including all the positive integers with 0 as the zero element, and the common addition function as the + function, and partially ordered by "less than" ($<$).

Definition 7. *A function δ from the set of events defined by an annotated Alice-and-Bob specification to a cost set C which is 0 on the accept events is called an **event cost function**.*

Note that the cost of a verification event is expected to express the expense of performing the verification, and the cost of sending a message is expected to express the expense of preparing that message.

Definition 8. *Let P be an annotated Alice-and-Bob protocol, let C be a cost set, and let δ be an event cost function defined on P and C. We define the **message acceptance cost function** associated with δ to be the function δ' on events following the receipt of a message as follows:*

If the line $A \rightarrow B : O_1, \ldots, O_k \parallel M \parallel V_1, \ldots, V_n$ appears in P, then for each event V_j:

$\delta'(V_j) = \delta(V_1) + \ldots + \delta(V_j)$.

The message acceptance cost function specifies the cost of processing messages up to reaching a failed verification event.

Meadows [2] [3] went on to introduce protocol engagement cost based on event cost function and message acceptance cost function. But Meadows' protocol engagement cost function is only defined on accept events. We extend the definition of protocol engagement cost to include all the valid events occurring at the defender of the protocol.

Definition 9. *We define the **protocol engagement cost function** associated with δ to be the function Δ defined on all the events as follows:*

For each event V_m in line $A \rightarrow B : O_1, \ldots, O_k \parallel M \parallel V_1, \ldots, V_n$:

1. *If V_m is not an accept event, then $\Delta(V_m)$ is the sum of the costs of all operations occurring at B desirably-preceding V_m plus the cost of V_m (i.e. $\delta(V_m)$);*
2. *If V_m is an accept event and there are no lines $B \rightarrow X : O'_1, \ldots, O'_{k'} \parallel M' \parallel V'_1, \ldots, V'_{n'}$, then $\Delta(V_m)$ is the sum of all the costs of all operations occurring at B desirably-preceding V_m;*

3. *If V_m is an accept event and there is a line $B \rightarrow X : O'_1, \ldots, O'_{k'} \parallel M' \parallel V'_1, \ldots, V'_{n'}$, then $\Delta(V_m)$ is the sum of the costs of all operations occurring at B desirably-preceding V_m plus the sum of the costs of the O'_i ($\delta(O'_1) + \ldots + \delta(O'_{k'})$).*

Note as well the notion of *desirably-precedes* is the same as what is defined in [2]. This protocol engagement cost reflects one of the most common ways in which Denial of Service attacks can proceed: to persuade a principal to waste resources participating in a bogus instance of the protocol. The more capable the intruder is, the more steps the victim will be persuaded to take engaged in the protocol. As a result, the protocol engagement cost represent the victim's loss under Denial of Service attacks.

3.4 DoS Loss Probability Distribution

Before defining the DoS Loss Probability Distribution, we give the definition of *fail point*, which characterizes the fail model of the cryptography protocols. The participant stops proceeding to participate in the protocol until it reaches a fail point, where the verification event come out unsuccessfully.

Definition 10. *A **fail point** $\mathcal{P}$ is a pair (L, E) denoting the place where the protocol will fail in verification at event E in line L.*

If the responder of the protocol fails in the verification of the first event in the first message, we say it fails at point $\mathcal{P}(L1, E1)$; If the responder proceeds to participate in the protocol until the last event in the last message, we say it fails at the last accept event because the cost of accept event is zero ($\delta(acceptevent) = 0$). We will use $\mathcal{P}.E$ to denote the event in fail point $\mathcal{P}$.

Definition 11. *A function η defined from the set of intruder capabilities to the set of fail points is called **Intruder Fail Point Function**.*

Definition 12. *The loss under Denial of Service attacks L_{DoS} is defined as the sum of the costs of all operations occurring at the principal participating in the protocol until it fails at point $\mathcal{P}(L, E)$.*

If an intruder with capability IC_i persuades the responder to participate in the protocol until the responder fails at point $\mathcal{P}_i(L, E)$, the *intruder fail point function* η maps IC_i to fail point $\mathcal{P}$, i.e., $\eta(IC_i) = \mathcal{P}_i$, and the DoS Loss of the defender is $\Delta(\eta(IC_i).E)$, i.e., $L_{DoS} = \Delta(\eta(IC_i).E)$.

Since we have all the definitions above, we arrive at the very point to figure out the DoS Loss Probability Distribution as follows.

Definition 13. *The **DoS Loss Probability Distribution Function** is defined from the set of DoS Loss (L_{DoS}) to a probability value within $[0, 1]$.*

Assume there are n different intruder capabilities $IC_i, IC_2, \ldots, IC_n$ with the probability of $\theta(IC_1), \theta(IC_2), \ldots, \theta(IC_n)$, respectively. Intruders with those n

capabilities can persuade the legitimate entity to participate in the protocol until failing at points $\eta(IC_1), \eta(IC_2), \ldots, \eta(IC_n)$, respectively.

The DoS Loss Probability Distribution is computed as follows.

$$Pr(L_{DoS} = loss) = \sum_{i=1,\ldots,n} \{\theta(IC_i) | \Delta(\eta(IC_i).E) = loss\} \tag{1}$$

Since we have arrived at the probability distribution of DoS losses, we can take Value at Risk as the method to evaluate the risk of Denial of Service in cryptography protocols.

3.5 Risk Evaluation with VaR

Before giving the Value at Risk (VaR) definition of DoS risk in cryptography protocols, we should recall the definition of VaR in financial language.

Definition 14. *Using a probability of α percent and holding period of t days, an entity's Value at Risk is the loss that is expected to be exceeded with a probability of only α percent during the next t-day holding period.*

Mathematically, VaR is the α-quantile of the Probality&Loss(P&L) distribution, i.e., it satisfies the relation:

$$Pr(\nu(\omega) \geq VaR) = \alpha \tag{2}$$

where we assume that the P&L distribution is a continuous and strictly monotone function, and both $\nu(\omega)$ (the financial loss function) and VaR are the absolute value of loss.

There are two key factors in the definition of VaR: the loss probability α and the time interval t. Typically values for the probability α are 1, 2.5, or 5 percent, while common holding period are 1, 2, and 10 business days, or 1 month. The choice of probability α is determined primarily by how the designer and/or user of the risk management system wants to interpret the Value at Risk: is an "abnormal" loss one that occurs with a probability of α. That means the probability of loss greater than VaR will be less than α. Because the risk of financial markets highly correlates with the holding time, the time interval t cannot be neglected. But when we are evaluating the risk of DoS attacks in cryptography protocols, the holding time is not inevitable, for the vulnerabilities in the cryptography protocols do not vary with respect to time.

Now that we have recalled the definition of VaR in financial language, we are ready for the definition of Value at Risk for DoS vulnerabilities in the language of cryptography protocols.

Because the loss under Denial of Service attacks in our model is discretely distributed, the definition of VaR should be modified to accommodate the discretely distributed variables.

Definition 15. *Using a probability of α, an entity's Value at Risk is the maximum of the DoS loss value that is expected to be exceeded with a probability of equal to or greater than α.*

Mathematically, VaR is the value satisfying the relation:

$$VaR = \max L_i \qquad s.t. \quad Pr(L_{DoS} \geq L_i) \geq \alpha \tag{3}$$

where $L_1, L_2, \ldots, L_n$ *are the* n *discretely distributed loss value with probability* $\gamma(L_1), \gamma(L_2), \ldots, \gamma(L_n)$.

Based on this definition of VaR in cryptography protocols, we give an algorithm for the computation of VaR value as Alg. 1. In Alg. 1, we are to find a value i that the probability of DoS loss greater than L_i is less than the predefined confidence α. At the beginning, we sort $L_1, \ldots, L_n$ so that $L_i \leq L_j$ for every $i < j$, and i is assigned n (Line 1–2). Then P_r, the sum of the probability of DoS loss greater than L_i is computed in Line 3–5. If P_r is greater than α, the algorithm returns L_i. (Line 6–8), and otherwise i is decreased by 1 and the algorithm goes to Line 2.

Alg 1. VaR Computation

```
1: sort(L_i, L_2, ..., L_n)
2: i ⇐ n
3: P_r ⇐ 0
4: for j = i to n do
5:     P_r ⇐ P_r + γ(L_j)
6: if P_r ≥ α then
7:     VaR = L_i
8:     return
9: i ⇐ i − 1
10: goto Line 2
```

Definition 16. *For the same probability* α*, the less the VaR value computed in our evaluation model is, the stronger the protocol is resistant to Denial of Service attacks.*

Because the VaR is the absolute value for the risk of the protocol under Denial of Service attacks, the less the risk, the stronger the protocol is resistant to Denial of Service attacks. As a result, Definition. 16 is self-evident.

We summarize the procedure of risk evaluation for cryptography protocol with the proposed model as follows.

1. Use the annotated Alice-and-Bob specifications to describe the cryptography protocol we want to analyze;
2. Chose a Cost Set C and specify an event cost function δ for each event in the annotated Alice-and-Bob specifications;
3. Following the second step, go on to figure out the message acceptance function δ' and protocol engagement cost function Δ for each event occurring at the defender.

4. Analyze the intruders. Specify all the intruder capabilities that threat the protocol and give the intruder capability probability distribution function θ;
5. For each intruder capability, determine the fail point where the intruders with this capability will fail at participating the protocol, then we get the intruder fail point function η;
6. Figure out the DoS Loss Probability Distribution Function from Equation 1;
7. Chose a probability value α, and take Alg. 1 to figure out the Value at Risk;
8. Use the VaR to evaluate the protocol: compare with other protocols or tell whether the system can survive under such risk.

Since we have defined the economical model for the risk evaluation of cryptography protocols based on Value at Risk, we are ready to apply the model to existing protocols to validate its applicability.

4 Applicability

In this section, we will show how we can apply the proposed economical model to the CCITT X.509 [11] authentication protocol (three messages version) and its enhanced version with client puzzle scheme to evaluate the risk of DoS attacks.

The CCITT X.509 authentication protocol can be annotated by the Alice-and-Bob specifications as follows.

1. $L_1 : A \rightarrow B : generatenonce_1, encrypt_1, sign_1 \parallel$
   $A, \{T_a, N_a, B, X_a, \{Y_a\}_{K_b}\}_{K_a^{-1}} \parallel$
   $checkname_1, checksig_1, checknonce_1, checktime_1, decrypt_1, accept_1$
2. $L_2 : B \rightarrow A : generatenonce_2, encrypt_2, sign_2 \parallel$
   $B, \{T_b, N_b, A, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b^{-1}} \parallel$
   $checkname_2, checksig_2, checknonce_2, checktime_2, decrypt_2, accept_2$
3. $L_3 : A \rightarrow B : sign_3 \parallel$
   $A, \{N_b\}_{K_a^{-1}} \parallel$
   $checkname_3, checksig_3, checknonce_3, accept_3$

A revised version of CCITT X.509 authentication protocol proposed by Wei et al. [12] use the client puzzles to enhance its defense against DoS attacks. Note that the puzzle is to find a *solution* so that the left k bits of $hash(S_{ii} \parallel S_{ir} \parallel solution)$ are all zeros. The evaluation result indicates the effectiveness of this enhancement against Denial of Service. This protocol is also described using the annotated Alice-and-Bob specification as follows:

1. $L_1 : A \rightarrow B : generateS_{ii} \parallel$
   $S_{ii} \parallel$
   $storeS_{ir}, accept_1;$
2. $L_2 : B \rightarrow A : generateS_{ir}, generatepuzzle \parallel$
   $S_{ir}, k \parallel$
   $storeS_{ir}, accept_2;$

3. $L_3 : A \rightarrow B : solvepuzzle, encrypt_1, sign_1 \parallel$
   $S_{ii}, S_{ir}, solution, A, \{S_{ii}, S_{ir}, T_a, N_a, B, X_a, \{Y_a\}_{K_b}\}_{K_a^{-1}} \parallel$
   $checksolution, checkname_1, checksig_1, checknonce_1, checktime_1, decrypt_1,$
   $accept_3$
4. $L_4 : B \rightarrow A : retrieve(S_{ii}, S_{ir}), encrypt_2, sign_2 \parallel$
   $S_{ii}, S_{ir}, B, \{S_{ii}, S_{ir}, T_b, N_b, A, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b^{-1}} \parallel$
   $checkname_2, checksig_2, checknonce_2, checktime_2, decrypt_2, accept_4$
5. $L_5 : A \rightarrow B : sign_3 \parallel$
   $S_{ii}, S_{ir}, A, \{S_{ii}, S_{ir}, N_b\}_{K_a^{-1}} \parallel$
   $checkname_3, checksig_3, checknonce_3, accept_5$

The cost set C is defined on all the positive integers including zero, where operation $+$ is the addition function, and $\leq$ is the 'less than' relationship.

We give an instance of event cost function δ. The cost is evaluated by the computation resource of doing the verifying computation. We have done some evaluation of the benchmarks for some well known cryptography algorithms with OpenSSL 0.9.8a [13] on Pentium M 1.6GHz, 512MB RAM, Linux 2.6.15-27-386, which is listed in the Appendix. The evaluation results show that in software implementation, and symmetric key algorithms are approximately 10 times slower than the hash algorithms. It is the observation of [14] that the asymmetric key cryptography is approximately 100 times slower than the symmetric key cryptography. So here we assume carrying out the simple verification event cost such as $checkname_i$ $checknonce_i$ and $generatepuzzle$ costs 1 unit of computation resource, and the algorithms containing $hash$ computation cost 10 units of computation resource such as $checksolution$, and the symmetric key algorithms such as $decrypt_i$ and $encrypt_i$ cost 100 units of computation resource, and signature algorithm such as $sign_i$ and $checksig_i$ verifications which involve public key computation cost 10000 units of computation resource. The event cost function δ is summarized in Table. 1 as below.

Table 1. Event Cost Function (δ)

Event	$checknonce_1$	$checksig_1$	$checkname_1$	$decrypt_1$	$encrypt_1$	$sign_1$
Cost	1	10000	1	100	100	10000
Event	$checknonce_2$	$checksig_2$	$checkname_2$	$decrypt_2$	$encrypt_2$	$sign_2$
Cost	1	10000	1	100	100	10000
Event	$checknonce_3$	$checksig_3$	$checkname_3$	$sign_3$	$generatepuzzle$	$checksolution$
Cost	1	10000	1	10000	1	10

From the event cost function, we can figure out the message acceptance cost function δ' and the protocol engagement cost function Δ. The details are neglected here.

Following the risk evaluation procedure defined in the Section 3.5, we now come to the tough job of analyzing the capability of intruders.

As far as these two protocols are concerned, we can classify different intruders into the following seven different intruder capabilities.

1. IC_1: denoting the capability with which intruders are able to impersonalize as a legitimate initiator of the protocol, e.g, getting a valid identity that the responder is willing to communicate with. This is a trivial ability for intruders, so we assign $\theta(IC_1) = p_1 = 0.5$;
2. IC_2: denoting the capability with which intruders are able to forge a valid signature of the corresponding entities; this is a much more powerful capability, so we assign a relatively small probability to it. $\theta(IC_2) = p_2 = 0.1$;
3. IC_3: denoting the capability with which intruders are able to forge a valid nonce accepted by the protocol responder. $\theta(IC_3) = p_3 = 0.6$;
4. IC_4: denoting the capability with which intruders are able to synchronize a valid time with the responder, and $\theta(IC_4) = p_4 = 0.2$;
5. IC_5: denoting the capability with which intruders are able to solve the puzzle challenged by the responder. Because this capability is difficult to get, we assign a small probability to it, i.e, $\theta(IC_5) = p_5 = 0.05$;
6. IC_6: denoting the capability with which intruders are able to tamper with the encrypted data in the first message ($\{Y_a\}_{K_b}$), and $\theta(IC_6) = p_6 = 0.1$;
7. IC_7: denoting the capability with which intruders are able to generate the valid cookie for communication in the revised version of CCITT X.509 authentication protocol, and $\theta(IC_7) = p_7 = 0.3$;

For the original version of CCITT X.509 authentication protocol, the intruders with capability IC_1 but without capability IC_2 will persuade the responder to participate in the protocol until it fails at point $(L_1, checksig_1)$. The intruders with capabilities IC_1, IC_2 but without IC_3 will fail in verification at point $(L_1, checknonce_1)$. The intruders with capabilities IC_1, IC_2, IC_3 but without IC_4 will fail at point $(L_1, checktime_1)$. The intruders with capabilities IC_1, IC_2, IC_3, IC_4 but without IC_6 will fail at point $(L_1, accept_1)$. The intruders with capabilities $IC_1, IC_2, IC_3, IC_4, IC_6$ will persuade the responder to finish all the operations, and the corresponding DoS loss is $\Delta(accept_3)$. With this, we can arrive at the DoS Loss Probability Distribution Function.

Lemma 1. *For the original version of CCITT X.509 authentication protocol, for probability* $\alpha = 0.03$*, the Value at Risk (VaR) under DoS attacks equals to* $\Delta(checktime_1)$*, which is* 10003 *units of computation resource.*

Proof. Since we have got the DoS Loss Probability Distribution $Pr(L_{DoS})$, we can arrive at $VaR = \Delta(checktime_1)$ after carrying out Alg. 1.

For the CCITT X.509 authentication protocol modified with client puzzles, our analysis on the intruders is a bit different. The intruders with capability IC_7 but without capability IC_5 will fail at point $(L_1, accept_1)$; The intruders with capabilities IC_7, IC_5 but without capability IC_1 will fail at point $(L_3, checkname_1)$; The intruders with capabilities IC_7, IC_5, IC_1 but without capability IC_2 will fail at point $(L_3, checksig_1)$; The intruders with capabilities IC_7, IC_5, IC_1, IC_2

but without IC_3 will fail at point $(L_3, checknonce_1)$; The intruders with capabilities $IC_7, IC_5, IC_1, IC_2, IC_3$ but without IC_4 will fail at point $(L_3, checktime_1)$; The intruders with capabilities $IC_7, IC_5, IC_1, IC_2, IC_3$ and IC_4 but without IC_6 will fail at point $(L_3, decrypt_1)$; The intruders with all the capabilities $IC_7, IC_5, IC_1, IC_2, IC_3, IC_4, IC_6$ will persuade the responder to finish all the operations, and the corresponding DoS loss is $\Delta(accept_5)$. This relationship indicates the DoS Loss Probability Distribution Function of this protocol.

Lemma 2. *For the CCITT X.509 authentication protocol modified with client puzzles, for probability* $\alpha = 0.03$*, the Value at Risk (VaR) under DoS attacks equals to* $\Delta(checkname_1)$*, which is* 14 *units of computation resource.*

Proof. Since we have got the DoS Loss Probability Distribution $Pr(L_{DoS})$, we can arrive at $VaR = \Delta(checkname_1)$ after carrying out Alg. 1.

From Lemma. 1 and Lemma. 2, and Definition. 16 since the VaR of the revised version of CCITT X.509 protocol is smaller than the original protocol, we can arrive at Proposition. 1.

Proposition 1. *From Lemma. 1 and Lemma. 2, the enhanced version of X.509 authentication protocol is more resistant to Denial of Service attacks than the original one.*

The evaluation result shows that more computation resource is exposed to DoS attacks in the CCITT X.509 protocol than its modified version. The risk evaluation result is self-evident and easy to understand. For common users without prerequisites of protocol analysis, they can get a comprehensive knowledge of the security performance of the protocol they are using: compare the security performance of different protocols that they are choosing from, and tell whether their systems will survive DoS attacks under such risk. In this example for instance, common customers get to know that the revised version of CCITT X.509 protocol is more robust than original one, and if more resource is prepared than what the VaR indicates, the system is survivable and sustainable. For the protocol analysts and designers, they can know whether their designs have met the security requirements as well as get clues to improve their jobs or test whether their ideas of security promotion really make sense with respect to DoS resilience. In this example for instance, protocol analysts and designers get to know that the client puzzle scheme has effectively enhanced the protocol's resilience to DoS attacks.

5 Related Work

Hamdi and Boudriga [15] gave a survey on the theory, challenges and countermeasures of computer and network security management. They reviewed the well-known risk management approaches and some shortcomings of the existing methodologies. They also set out common requirements that must be respected by any risk management frameworks, among which cost estimation

and attack modeling requirements are covered. As for DoS risk management, a lot of researches fall into the category of measuring and quantifying DoS impact [16] [17] [18], which are dedicated to measuring the impact of DoS attacks. On the DoS evaluation of cryptography protocols, a cost-based framework for analyzing vulnerabilities to network DoS attacks in protocols was proposed by Meadows first in [2] and then refined in [3]. Taking advantage of this evaluation framework, the protocol designer specifies a tolerance relationship and tells whether the protocol's resilience to DoS is within its tolerance. The tolerance relation matrix describes how much effort he or she believes it should be necessary to expend against an attacker of given strength. Smith [19] extended Meadows' framework to analyze JFK, an Internet key agreement protocol. Those researches have shed light on the evaluation of DoS vulnerabilities in protocols, however, without a probability model, they have not stepped towards the notion of risk evaluation. By characterizing the attackers with a probability model, this paper specifies how to evaluate the risk of DoS vulnerabilities in protocols, which is indicated by the Value at Risk (VaR), a widely accepted approach in financial risk management. Although proposed in financial community, VaR is not a new comer for computer scientists and engineers. Kleban and Clearwater did the first job employing the idea of VaR to evaluate the risk of computer systems [7] [8], however, little effort has been made for applying VaR to the risk evaluation of cryptography protocols since then. Value at Risk has a solid mathematical foundation and has achieved great success in financial risk evaluation. As a result, we adopt the idea of Value at Risk to evaluate the risk of DoS attacks in cryptography protocols in this paper.

6 Conclusion

In this paper, we propose an economical model for the risk evaluation of DoS vulnerabilities in cryptography protocols. Value at Risk (VaR) is defined and utilized to do the job of risk evaluation. To our best knowledge, this is the first work on the risk evaluation of DoS vulnerabilities in security protocols. The applicability and effectiveness of the proposed model is validated by applying it to analyze the CCITT X.509 authentication protocol and its modified version with client puzzles. Evaluation result shows that the modified version of the CCITT X.509 has enhanced the protocol's resistance to DoS. With the help of the model, common customer can get a comprehensive knowledge of the security performance of the protocol without any prerequisites of protocol analysis, and protocol analysts and designers can know whether their designs are effective and get clues to improve their work as well.

Acknowledgement

This work is part of Project No. 60673182 supported by the National Natural Science Foundation of China. The authors would like to thank the anonymous reviewers of ISPEC 2007 for providing valuable feedbacks.

References

1. Meadows, C.: Formal methods for cryptographic protocol analysis: Emerging issues and trends. IEEE Journal on Selected Areas in Communications **21**(1) (2003) 44–54
2. Meadows, C.: A formal framework and evaluation method for network denial of service. In: Proceedings of The 12th Computer Security Foundations Workshop. (1999) 4–13
3. Meadows, C.: A cost-based framework for analysis of denial of service networks. Journal of Computer Security **9**(1) (2001) 143–164
4. Holton, G.A.: Value-at-Risk Theory and Practice. Elsevier (2003)
5. Bernstein, P.: Against the gods: The remarkable story of risk. John Wiley and Sons Inc (1996)
6. Basel-Committee: Consultative document: The new basel capital accord. http://www.bis.org/publ/bcbsca03.pdf (2001)
7. Kleban, S., Clearwater, S.: Computation-at-risk: Assessing job portfolio management risk on clusters. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, IEEE Computer Society Press (2004) 254–260
8. Kleban, S., Clearwater, S.: Computation-at-risk: Employing the grid for computational risk management. In: Proceedings of the 18th IEEE International Conference on Cluster Computing, IEEE Computer Society Press (2004) 347–352
9. Lowe, G.: Casper: a compiler for the analysis of security protocols. In: Proceedings of 10th IEEE Computer Security Foundations Workshop, IEEE Computer Society Press (1997) 18–30
10. Millen, J.K.: Capsl: Common authentication protocol specification language. Technical Report MP 97B48, The MITRE Corporation. http://www.csl.sri.com/users/millen/capsl/ (1997)
11. CCITT-Committee: Ccitt recommendation x.509: The directory authentication framework. http://www.lsv.ens-cachan.fr/spore/ccittx509_3.html (1988)
12. Wei, J., Chen, Z., al. et: A new countermeasure for protecting authentication protocols against denial of service attack. Acta Electronia Sinica **33**(2) (2005) 288–293
13. OpenSSL: The open source toolkit for ssl/tls. (http://www.openssl.org)
14. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. John Wiley and Sons Inc (1996)
15. Hamdi, M., Boudriga, N.: Computer and network security risk management: Theory, challenges, and countermeasures. International Journal of Communication Systems **18**(8) (2005) 763–793
16. Mirkovic, J., et al: Measuring denial of service. In: Proceedings of the 2006 Quality of Protection Workshop, ACM Press (2006)
17. Mirkovic, J., Fahmy, S., Reiher, P.: Measuring impact of dos attacks. In: Proceedings of the DETER Community Workshop on Cyber Security Experimentation. (2006)
18. Chen, Y., Bargteil, A., Bindel, D., Katz, R.H., Kubiatowicz, J.: Quantifying network denial of service: A location service case study. Lecture Notes on Computer Science **2229** (2001) 340–351
19. Smith, J., Gonzalez-Nieto, J.M., Boyd, C.: Modelling denial of service attacks on jfk with meadows's cost-based framework. In: Proceedings of the Fourth Australasian Information Security Workshop, Australian Computer Society, Inc. (2006)

Appendix: Benchmarks for Cryptography Algorithms

We present our benchmarks for both hash function and symmetric key cryptography algorithms in this appendix. This evaluation result shows in software implementation, the symmetric algorithms are approximately 10 times slower than the hash algorithms. The experiment is done on a PC with Pentium M 1.6GHz, 512MB RAM, Linux 2.6.15-27-386, and OpenSSL 0.9.8a, and the x-axis represents the buffer size used by the algorithm, and the y-axis represents the size of data processed by the algorithm in 1 second.

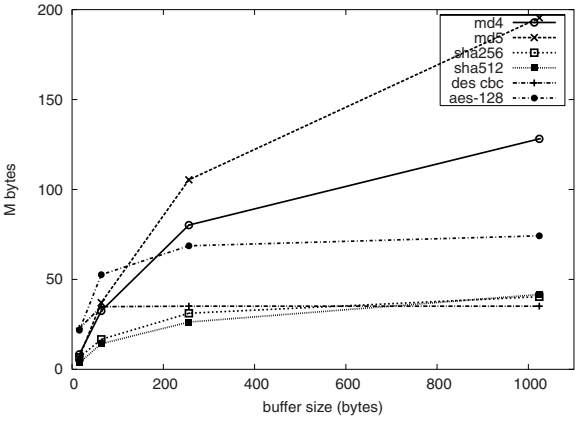


Fig. 1. Benchmarks for Symmetric Key Cryptography Algorithms

Practical Uses of Virtual Machines for Protection of Sensitive User Data

Peter C.S. Kwan[1] and Glenn Durfee[2]

[1] Electrical Engineering Department, Princeton University*
pkwan@princeton.edu
[2] Palo Alto Research Center
gdurfee@parc.com

Abstract. Systems running commodity software are easily compromised with malware, which may be used by attackers to extract personal information of the users of the systems. This paper presents Vault – a system that uses a trusted software component to prevent the exposure and abuse of sensitive user data in the presence of malware. Users input and store their sensitive data only in the trusted component, which is separated from the commodity system by a virtual machine monitor. We define a protocol framework for the interactions required between different system components in order to protect user secrets, even if the user is running a commodity operating system with arbitrary (and possibly malicious) software load, while introducing minimal changes to the user experience. Our design takes advantage of the isolation guarantees and safe I/O multiplexing of virtual machine technology to attain a high degree of security under a severe threat model.

We demonstrate that our approach is practical by implementing prototypes for two applications: (1) submission of long-term secrets, such as password and credit card data, to a web server, and (2) SSH user authentication using `ssh-agent`. In both cases we made minimal changes to existing software components.

1 Introduction

The widespread use of personal computers running vulnerable commodity operating systems (OSes) has put the personal data of millions of users at risk – data that is easily exploited for identity theft or other fraudulent activities [17]. Attacks that harvest sensitive data[1] from users' computers take advantage of two crucial weaknesses in modern commodity OSes: First, it is notoriously easy to introduce malicious software into a commodity OS through viruses, worms, Trojan horses, and spyware. Second, once running locally, malicious software can easily obtain sensitive information through the use of powerful APIs exposed by the OS, such as keystroke interception and disk I/O. Many

* Much of this work was done while the author was an intern at Palo Alto Research Center.

[1] In this paper, we use the phrases "sensitive data", "secrets", "sensitive information" and "personal data" interchangeably to refer to a broad class of data users would like to keep private (such as passwords, credit card numbers, and cryptographic keys).

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 145–161, 2007.

security practices, such as the use of secure network protocols and security tokens, become much less effective when the attackers can simply sniff at every key the users type for their passwords, PIN, and credit card numbers, or when the attackers can read any file on the file system. While this is well-known, the superior functionalities and price advantages of modern commodity systems mean they will continue to be in widespread use despite their vulnerabilities.

To address these concerns, we introduce Vault, a virtual-machine-based security system designed to protect sensitive data on commodity systems. Vault uses a virtual machine monitor (VMM) to compartmentalize a physical machine into two virtual machines (VMs). Sensitive data are stored and handled only in the *trusted VM*, while all other computing activities occur in the *untrusted VM*. Users are free to configure the untrusted VM with a commodity OS and a software load of their own choosing. On the other hand, the trusted VM runs a minimal OS with a restricted set of functionalities. To give an idea of what the user experience is using Vault, consider an online shopping scenario. First, a user starts an online shopping session with a web merchant, using their favorite web browser in the untrusted VM. During checkout, instead of entering her credit card number into the browser, she *explicitly* switches to the trusted VM, and inputs it there, where it is then securely transmitted to the merchant's server. Afterward, the system automatically switches back to the untrusted VM to continue the checkout process.

This design is an example of a broader class of systems that protect sensitive data using of small, isolated, trusted components. The trusted components in Vault are the VMM and the trusted VM. Crucially, the trusted VM has a trusted I/O path to the user, especially for receiving confirmations for actions involving the use of sensitive data. This is because the VMM controls the multiplexing of I/O devices, and thus is able to separate the user interactions with the trusted VM from those with the untrusted VM. As our main contribution, we define a protocol framework for the delegation of the handling of sensitive data to the trusted VM. This protocol framework prevents attacks from untrusted components and allows users to guard the use of their sensitive data.

We further show that this framework is practical and can be readily integrated with existing applications. We built a prototype for two of the most common online applications involving user secrets: Web-based online shopping and the `ssh-agent` authentication module used in SSH logins.

The rest of the paper is organized as follows. Section 2 surveys past work in protection of user data in untrusted environments. It is followed by our assumptions on threat and trust in Section 3. Section 4 describes the design of Vault and how different components in the systems interacts to achieve secure use of sensitive data. Then we describe our prototypes in Section 5. In Section 6, we discuss the requirements for widespread adoption of our solution, and argue that it is realistic and achievable. We conclude in Section 7.

2 Related Work

Using separation to improve security of complex systems has had a long history. Small, trusted and tamper-resistant hardware components are used to store sensitive data and handle their operations. Smartcards and secure co-processors [23, 27] are examples of

small trusted components designed to be deployed in untrusted hosts. In those designs, no sensitive data are stored in the host. Instead, the host requests the use of sensitive data, typically cryptographic keys, via an API exposed by the component. However, there is currently no viable way for the trusted component to determine the legitimacy of requests coming from the host, if the component does not have prior state about what can be trusted. For example, the authentication services provided by a smartcard are guarded by a PIN. But since the PIN must be entered by the user *via* the host, it can easily be intercepted by a keystroke-logging malware. Although still unable to access the private key stored in the smartcard, the malware can now make requests, using the sniffed PIN, to the smartcard for operations involving the private key. We have addressed this problem by providing a trusted I/O path between the user and the trusted VM. The trusted VM can then obtain user confirmations for operations involving sensitive data. The work in [10] also employs user confirmation to prevent misuse of sensitive data. However, being a hardware solution, they can only make use of primitive LED and push buttons for user interactions. Our VMM-based approach provides a much more viable user interface for the trusted component.

Isolation mechanisms integrated in the processors have also been proposed. Lee et al. [14], Suh et al. [24], and Lie et al. [15] use cryptographic methods to create isolated and tamper-evident execution environments. Intel [12] and AMD have also proposed curtained memory for the creation of a protected compartment, possibly for a secure kernel enforcing security policies. We focus in this paper not on the mechanisms for isolation, but on how to make use of the isolation guarantees. In this spirit, Jiang et al explore the use of a co-processor to build trust into the services provided by remote servers [13]. Marchesini et al propose the use of the attestation functionality of the TPM [26] chip to attest to the authenticity of a security "Enforcer", which in turn attests to the configuration of the software platform [16]. These work protect users against malicious server operators. We instead protect user's data from malware running on their own computers.

Another approach to provide separation is virtualization. Terra [7] and NetTop [18, 19] use VMM to separate compartments of differing levels of trustworthiness. For example, one VM may be used only for trusted applications handling top secret documents, while another VM, considered less trustworthy, may be used for web browsing. Successful compromise of the web browsing VM does not affect the security of the other VM. The use of VMM for isolation is similar to our work. However, the aforementioned work fails to protect sensitive data that cannot be restricted to a trustworthy VM. For instance, online shopping typically requires entering passwords and credit card information into a web browser loaded with third-party plug-ins. Because of their notorious vulnerability, such browsers usually operate only in the least trusted VMs. Thus, despite the availability of more trusted VMs, users are nevertheless required to input sensitive data into a VM that is much more likely to be compromised.

Proxos [25] allows application designers to specify a subset of the system calls of a commodity OS that their applications do not trusted, and delegates those calls to a trusted private OS, which is separated from the commodity OS by a VMM.

The `factotum` component in the Plan 9 OS represents the closest concept to our work [4]. `Factotum` handles all authentication requests on behalf of the user,

optionally requiring user confirmations. Applications wishing to take advantage of `factotum` needs to be redesigned to delegate the authentication process to `factotum`. However, `factotum`'s trust model includes the Plan 9 OS. Vault addresses a larger set of threats by considering a commodity system in the untrusted VM that may be fully compromised.

3 Security Model

We consider the threats posed by remote attackers on user's long-term secrets used for authentication and e-commerce purposes. We assume attackers can launch attacks from over the network. By exploiting bugs, attackers can compromise a system to install malicious code, such as a keystroke logger. Attackers may also introduce malicious code by employing social engineering techniques. Once installed, we assume that such malicious code may run at the same processor privilege level as the OS kernel, giving it unrestricted access to the private data of all applications, as well as the ability to arbitrarily manipulate application execution. Malware gaining access at this level is not uncommon because most users run with administrator privileges, allowing the installation of kernel components including device drivers. In addition, we also assume that the attackers can intercept and modify all network traffic.

However, we do not consider physical attacks on the hardware platform, such as probing of system buses and keyboard. Also, we do not directly address phishing attacks. Methods to protect against phishing [5, 21] are orthogonal and can be combined with this work.

Under this threat model, we trust the hardware platform. We also trust the VMM and the software that runs in the trusted VM. The remote servers (e.g. Amazon.com), to which the user intends to communicate with and prove possession of his long-term secrets, are also trusted. This means that genuine servers are trusted to handle the secrets correctly. However, the identities of such servers upon connections remain untrusted until proven using certification-authority-based mechanisms provided in secure communications protocols, such as Transport Layer Security (TLS) [6].

4 Design of the Vault System

Figure 1 illustrates the main entities and software components in our design. Vault makes use of a type 1 VMM to provide virtualized hardware. A type 1 VMM runs directly on the hardware [9], whereas a type 2 VMM, such as VMware Workstation, runs atop a host OS. The correct functioning of a type 1 VMM does not depend on an underlying host OS.

The VMM supports two VMs. The untrusted VM runs a commodity OS and an arbitrary application load, chosen by the user. The user conducts most of her computing activities using this VM. The freedom to configure the untrusted VM allows the user to continue to enjoy the superior functionality of a commodity system. Such configuration, however, may contains vulnerabilities. We therefore emphasize that it is completely untrusted, i.e., any malicious software may be operating in the untrusted VM. In contrast, the trusted VM runs a minimally configured OS and only the Vault application. The

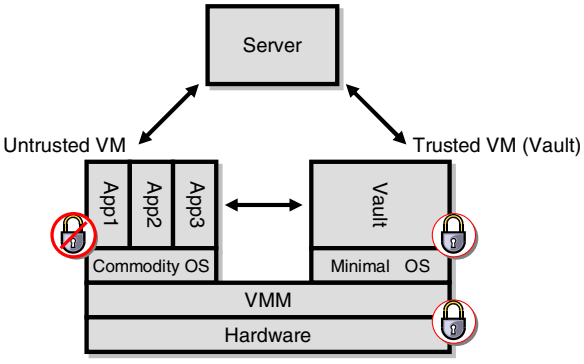


Fig. 1. The software components in Vault. The hardware, VMM, and the software stack of the trusted VM are trusted (indicated by the lock).

Vault application handles long-term secrets of the user. Its provides a graphical user interface (GUI) for user to input the secrets, or to authorize the use of secrets previously stored.

With this organization, the user's secrets are prevented from observation and tampering by any malicious software in the untrusted VM. This guarantee, however, depends on the correctness of the VMM and the software in the trusted VM – they must be correctly designed and implemented in such a way they do not leak any information about the user's secrets. In other words, the VMM, the minimal OS, and the Vault application forms a trusted code base (TCB) of Vault. While it is extremely difficult to fully verify that a piece of software is free of vulnerabilities using today's software engineering methodology, we describe in Section 5 how we take steps to approach this requirement for the TCB.

As mentioned earlier in Section 2 and in [10], the trusted VM needs to be able to differentiate between legitimate and malicious requests from the untrusted VM, in order to prevent misuse of user's sensitive data. Although this is undecidable in general [2, 3], we can make forward progress by making the definition of legitimacy more precise. We say that a request to use or input sensitive data is legitimate when *it is explicitly approved by the user after he or she has been presented with all relevant information associated with the request.*

We recognize that not all users can make good security decisions, even when presented with relevant information. Nevertheless, we view the problem of designing a user interaction model that encourages correct user decisions as a complementary usability problem, a solution to which would work in concert with the software architecture outlined in this paper. The focus of this work is on how to ensure the genuineness of the information presented.

4.1 Trusted I/O and Transition to Vault

In presenting the information and receiving input from the user, we take advantage of the VMM's ultimate control over the I/O devices to present a trusted GUI to the user.

There are two aspects to the realization of a trusted I/O path. First, it is enabled by the trusted multiplexing of the keyboard, video, and mouse by the VMM. Second, and more importantly, we must ensure that malicious software cannot easily spoof a Vault-look-alike in the untrusted VM, enticing sensitive data from the user. In order words, the user must be able to establish which VM she is interacting with.

This can be accomplished by associating special user actions with the transition from the untrusted VM to the Vault. One option to do so is the use of attention key sequences, such as `Ctrl-Alt-Del` required on the Microsoft Windows™ login screen. In Windows, users are trained to associate the attention key sequence with the display of the password prompt. If malicious software spoofs the prompt, the user realizes this by the absence of her special action. Similar to the Windows OS, the VMM can intercept a pre-defined sequence to trigger the transition to the trusted VM, without passing the sequence to the untrusted VM. A more intuitive interface might be a dedicated key for switching to Vault, similar to the password key in [21]. Alternatively, a graphical VM switch can be displayed at the top the screen, as is employed in NetTop [19]. This region must be controlled by the VMM and cannot be obstructed or spoofed by the VMs.

4.2 Protocol Framework for Delegation

To use Vault, the user first initiates a session with a remote server in the untrusted VM. When a long-term secret is requested by the server, she switches to the Vault application in the trusted VM. The Vault application takes over the session from the untrusted VM and requests the secret from the user via a trusted I/O path. The user inputs the secret, which is then transmitted via a secure connection from the Vault application to the server. The system then switches back to the untrusted VM, to continue the session.

In this framework, to ensure that the user is not enticed to submit secrets to malicious servers, the Vault application must (1) verify the information received from the untrusted VM; (2) present relevant information (such as server name) for user confirmation; and (3) establish a secure tunnel to the remote server for the transmission of secrets.

The protocol described below achieves these requirements. To make use of the Vault, existing applications need to be modified to delegate the handling of long-term secrets to the Vault. Figure 2 illustrates the exchanges taken in the protocol.

1. The user begins a session of interaction with the remote server via an application, e.g. a web browser, running in the untrusted VM. She proceeds to a point where long-term secrets is requested. This requests can be a password, a credit card number, or a cryptographic response to a challenge.
2. The user explicitly initiates the transition from the untrusted VM to the trusted VM by pressing a special attention key sequence.
3. In the trusted VM, the Vault application detects that it has been activated. It requests identifiers for the session and the server (e.g. an IP address or a URL) from the application running in the untrusted VM. We refer to these identifiers as `sessionID` and `servername`. Both are sent from the untrusted VM to the Vault application.
4. Using `servername`, the Vault application establishes a secure tunnel with the server. This can be set up using various secure communication protocols, such as

TLS [6]. In the secure tunnel, Vault sends the server `sessionID`. Note that this secure tunnel may optionally be relayed via the untrusted VM.

5. The server verifies the `sessionID`. Only if it is valid, the server sends the Vault application information pertaining to that session, as well as a request for the long-term secret. This request may be formatted as an XML form according to some extensible protocols between Server and Vault. Verification by the server prevents the untrusted VM from supplying a malicious `sessionID` at Step 3 above.
6. The Vault application displays information about the server, derived from the secure tunnel. For example, in TLS, the information is the name of the server embedded in its digital certificate. This allows the user to confirm that she is interacting with the intended server. Together with Step 5, this completes the verification of the `sessionID` and `servername` received from the untrusted VM.
7. Once the information is confirmed to be correct, the user responds to the request either by entering the requested secret, or authorizing the use of a secret stored previously by the Vault application. Because of the trusted I/O path, no malicious software in the untrusted VM can eavesdrop or tamper with the user's interaction with the Vault application.
8. The Vault application sends the user's responses to the server.
9. The server concludes by sending the Vault an instruction intended for the application in the untrusted VM. For example, this can be the next URL to be loaded in a browser.
10. The Vault application relays this instruction to the untrusted VM, and signals the VMM to transition back to the untrusted VM.[2]
11. The application in the untrusted VM executes the instruction issued by the server. It continues the session, which is now in a new state after the successful input of the long-term secrets. The user continues to use the untrusted VM for the rest of this session.

With this protocol framework, together with a trusted I/O path between the Vault application and the user, we are able to present genuine information about the session to the user and allow her to approve the use of her secrets, preventing misuses. In addition, the change to user experience is minimal. The only extra step for the user is the transition to the Vault application at Step 2.

5 Implementation

We prototyped Vault along with changes to two popular applications that use long-term secrets. One application is the submission of passwords and credit card data for online commerce with web merchants. The second is public-key user authentication in SSH. We found that only minimal modifications are required to adapt these applications to use Vault. The next section discusses the underlying infrastructure we built to support the Vault prototype.

[2] The reason we relay an instruction from the server to the untrusted VM via the Vault, as opposed to sending it directly from the server to the untrusted VM, is that in stateless applications such as web browsing, a connection between the server and the untrusted VM may no longer exist.

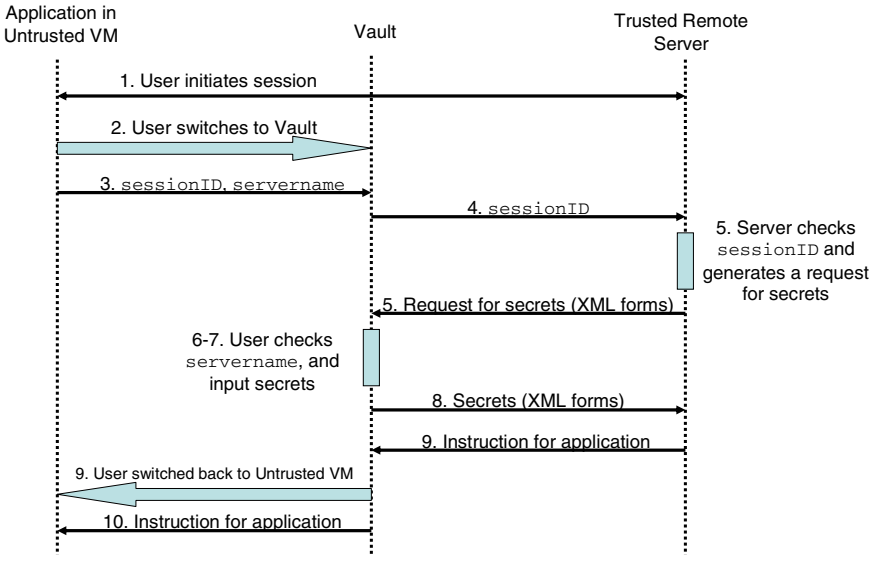


Fig. 2. The protocol framework for delegating the handling of long-term secrets to the Vault application

5.1 The Virtual Machine Infrastructure

The Xen Virtual Machine Monitor Our design is VMM-agnostic. In the prototype, we use Xen, an open-source type 1 VMM [1] that has improving support for true virtualization using Intel virtualization technology [11]. In Xen's organization, Domain-0 is a trusted management VM, and is started by the Xen VMM at boot time. This VM is trusted to control the rest of the system, such as starting and stopping other VMs. In addition, only Domain-0 has access to all devices on the system. We therefore consider Domain-0, and the services provided by it, a trusted extension of the VMM. In other VMMs, such as VMware ESX Server, the devices are typically controlled directly at the VMM layer.

Operating Systems and Applications in the VMs. Both the untrusted VM and trusted VM run Linux in the prototypes. The installation in the untrusted VM is a full client workstation, whereas the trusted VM contains a minimally configured Linux installation with limited network connectivity. Although even our minimial Linux installation probably still contains vulnerabilities, we use this for our research prototype to approximate the requirement of a TCB. (In a production-quality version of Vault, we imagine that a carefully designed and vetted TCB would be used instead. Since very few OS services are required, this could be a very small TCB.) In our prototype, only the Vault application runs in the trusted VM, and network connectivity is limited to only what is needed by the Vault application. The Vault application is minimally designed to provide

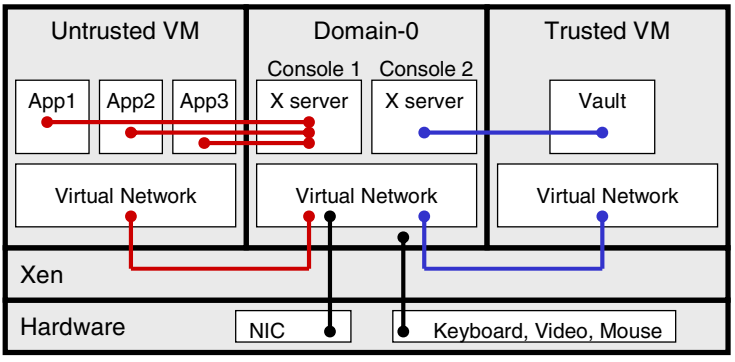


Fig. 3. The setup of the virtual machines. Applications in each VM display in a VM-specific virtual console in Domain-0. Each VM also have an isolated virtual network connection to Domain-0.

only functionalities required to handle user's long-term secrets. By limiting functionality and connectivity, we argue that even our research prototype is sufficient to neutralize a large class of potential attacks on the trusted VM.

Multiplexing of Trusted I/O. One of the most important requirement for the Vault is the trusted I/O path to the user. We leverage on the fact that the trusted Xen Domain-0 controls Linux's virtual consoles (accessible by the `Ctrl-Alt-Fn` sequence) to provide an attention key sequence for switching between the VMs. We run X servers on virtual console 1 and 2 of Domain-0. These X servers receive commands from applications in the VMs via the X Display Manager Control Protocol (XDMCP) and the X11 protocol [29]. Figure 3 illustrates this organization. Because the `Ctrl-Alt-Fn` sequences are serviced by Domain-0 without passing to the VMs, the transition between VMs is non-bypassable. Note again that X11 has not been verified to be bug-free, and we use it for the purpose of illustrating the configuration of the trusted I/O path. Utlimately, security depends on the safe I/O multiplexing feature provided by the underlying VMM. While Xen does not provide the highest assurance, commerically available VMMs do, as illustrated by their uses in NSA's NetTop architecture [19].

Virtual Network Configuration. The communications between the trusted VM and untrusted VM are handled by a virtual network exposed by the Xen VMM. The virtual network topology must ensure that the untrusted VM cannot eavesdrop on the traffic from the trusted VM. Therefore, the two VMs must never be connected to the same bridge. In our implementation, the trusted Domain-0 runs as a router and a network address translator (NAT). Two virtual interfaces are defined in Domain-0 for isolated connections to the two VMs (Figure 3).

5.2 Prototype 1: Submission of Long-Term Secrets to a Web Merchant

We consider the everyday scenario of entering passwords and credit card numbers for online shopping. The system we implemented is generic enough for any e-commerce web site to take advantage of Vault. We first describe the new user experience:

User Experience. The user starts by conducting normal online shopping activities in the untrusted VM. Once the user navigates to a page requesting a long-term secret, such as a credit card number at the check-out page, she presses an attention sequence to switch into the trusted VM. In our implementation, the attention sequence is `Ctrl-Alt-F10`. This switches the display to virtual console 2 – that of the trusted VM. In the trusted VM, the Vault application displays the name of the web merchant and its requests for the secret. The user checks the name of the merchant. If it is correct, she inputs the secret in the trusted VM and authorize it to be sent to the merchant. Next, she is automatically switched back to virtual console 1 of the untrusted VM. The browser loads a new page, which may indicate that the credit card information has been received. The user continues with the rest of the session in the untrusted VM, making full use of the rich functionality provided by the commodity OS in the untrusted VM.

As the user go through these steps, she participates in the protocol described in Section 4.2. Note that the only additional step required of user is the explicit switch to the trusted VM before she enters her credit card number. Next, we describe the modifications required in the untrusted VM and the server.

Communications between the Trusted and Untrusted VMs. As part of the protocol, the Vault application needs to get information from the web browser running in the untrusted VM. We augmented the Firefox browser with a browser extension that listens for connections from the Vault. Upon a connection from the Vault application, the browser extension sends it `sessionID` and `servername` (step 3). The values of `sessionID` and `servername` are specified by the web merchant as hidden input fields in an HTML form, for example:

```
<input type=hidden name=vault_sessionID value=165996028513308>
<input type=hidden name=vault_servername
       value='https://www.amazon.com/vault'>
```

Thus, the browser extension simply reads these values from the HTML page and sends them to the Vault application. Note that these values may be tampered with in the untrusted VM. Therefore, they are verified by the server at step 5, and by the user at step 6 of the protocol in Section 4.2. After sending these to the Vault application, the browser extension waits for an instruction that the Vault received from the merchant (Step 9). In this prototype, the instruction is simply a URL link to a web page acknowledging the receipt of the credit card number. The browser extension completes the protocol by loading this page in the browser.

Server Modifications. On the server side, hidden input fields for `sessionID` and `servername` are inserted in the check-out web page. The secure communications between the Vault application and the merchant is handled by a new set of handlers (steps 4, 5 and 8). We envision the Vault-server communications be based on a standard in XML format, possibly established by industry consortia. In addition, the server updates an internal session database, in order to keep track of the interactions with the Vault application and the session originally started by the user in the untrusted VM. Such session tracking feature is commonly employed in actual e-commerce sites. It is therefore straightforward to augment it with data for Vault.

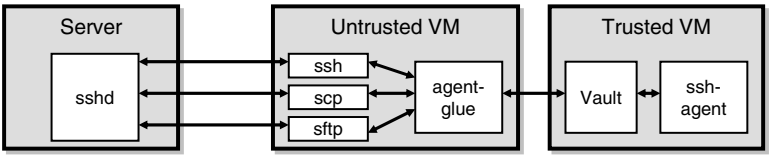


Fig. 4. The glue component in the prototype for SSH user authentication

The Vault Application. When user switches to the trusted VM, the Vault application detects it and establishes a connection to the Firefox extension in the untrusted VM to retrieve `sessionID` and `servername`. Using `servername`, it establishes a separate HTTPS connection to the web merchant. The Vault rejects the connection if the certificate is not signed by a known certificate authority. In this connection, the Vault sends the `sessionID`, and receives an XML forms requesting for long-term secrets. Next, the Vault prompts the user for the requested long-term secrets. Importantly, it also displays the name of the merchant in its GUI. This is the Common Name in the certificate presented by the merchant for the HTTPS connection. This allows the user to confirm that she is communicating with the right merchant. After user's input, the Vault sends the long-term secrets to the merchant via the HTTPS channel. As a last step, it receives a new URL from the merchant. This URL is the next page to be loaded in the browser running in untrusted VM. Finally, the Vault application triggers a switch back to the untrusted VM by making a request to a helper program resident in Domain-0. This helper program writes to `/dev/console` to trigger a virtual console switch back to the untrusted VM.

5.3 Prototype 2: Public-Key User Authentication in SSH

Overview of SSH Using `ssh-agent`. The SSH protocol supports many user authentication options. One of them uses public key cryptography [30]. The authentication private key is stored locally on the client computer, and is typically encrypted using a user-selected passphrase. At *every* SSH login, the `ssh` client prompts the user for the passphrase to unlock the private key, which is then used to generate a response (a signature) to a challenge posed by the remote SSH server. The `ssh-agent` application streamlines this process by enabling a single sign-on feature. At launch, `ssh-agent` prompts user for the passphrase *once*. It then runs as a background process, listening on a Unix socket. Client applications (such as `ssh`, `scp`, etc) engage `ssh-agent` via the socket to authenticate the user to remote SSH servers. Essentially, `ssh-agent` signs, using the unlocked private key, any binary data that is passed to it from any client. This "signature service" is unguarded once the passphrase has been entered at the launch of `ssh-agent`.

In our threat model, there are two attacks on an `ssh-agent` running in an untrusted VM: (1) an attacker who has compromised the OS can read and write the private key files stored locally, and (2) short of tampering with the private keys, a malicious program simply makes use of the unguarded signature service to login to a remote server where the user has an account, impersonating the user.

Moving `ssh-agent` *to the Trusted VM.* To defend against the first attack, `ssh-agent` and its public-private key database are relocated to the trusted VM. This step requires no changes to `ssh` and `sshd`. We first relocate `ssh-agent` to the trusted VM, and with it all the private keys of the user. We then implemented a new glue process, `agent-glue`, in the untrusted VM, as shown in Figure 4. `agent-glue` disguises as `ssh-agent` by opening a Unix socket and setting up the appropriate environment variables. It relays any connections to the trusted VM. In the trusted VM, the Vault application waits for connections from `agent-glue`, and in turn relays the connections to the genuine `ssh-agent` running there. With this arrangement, the private keys and the passphrase are protected from the untrusted VM. However, the *use* of them is still unguarded.

Securing against Misuse of Private Key. To defend against the second attack, we need to add a confirmation stage in the Vault application. Each time the user logs in to a remote server using SSH, she needs to explicitly switch to the trusted VM to confirm the use of her long-term secrets, using the protocol framework described in Section 4.2. However, with this protocol, `sshd` needs to be modified significantly in order to support a separate secure tunnel between the Vault application and `sshd`. A simpler approach can be devised by observing that the secure tunnel is a requirement only if the secrets need to be transmitted to the server in their actual forms, such as credit card numbers. For SSH, there is no such requirement because the secret – the private key – is never actually transmitted to the `sshd` server. The user only needs to prove the possession of the private key by furnishing a valid signature. This property is generally true for challenge-response authentication protocols. In Appendix A, we consider a subtle attack on this class of authentication under our threat model, and define a simplified delegation protocol framework that eliminates the Vault-server secure tunnel. The main requirement of the simplified protocol is that the authentication scheme must bind *both* `sessionID` and `servername` in the signature. However, SSH protocol only binds the `sessionID` in the signature. But we found that it is fairly easy to argument the SSH programs to fulfill the requirement without modification to the SSH protocol.

Modifications to `ssh` *and* `sshd`. During user authentication, the `ssh` client proves the possession of the private key by signing a pre-defined data structure that includes the `sessionID`, among other data. However, `servername` is absent from the structure. We need to augment this data structure with the `servername`. Conveniently, the `sessionID` field is variable-length. The `ssh` client can thus prefix the original `sessionID` field with `servername` and a delimiting character. This augmented structure is sent to `ssh-agent` in the same manner as the current challenge – as one binary data block. The Vault application intercepts this message to extract `servername` in order to display it for user confirmation, and only forwards this whole message to `ssh-agent` for signing if the user confirms that this is the intended use of her private key.

On the server side `sshd` checks *both* `sessionID` and `servername` to ensure that the signature is the correct authenticator intended for it. We estimate that this change requires only modest modifications to the code of `ssh` and `sshd`.

Limiting the Danger of Session Hijacking. Despite providing strong protection for long-term secrets, our design does not protect short-term secrets used in the untrusted VM,

most notably the session keys used in `ssh`. This implies that sessions are still vulnerable to *session hijacking*, whereby an attacker takes over a session (possibly without the user's awareness).

Because our threat model implies that no application in the untrusted VM is safe, there is nothing that can be done to prevent session hijacking. But we would like to design our system to minimize the damage caused by session hijacking, and in particular, prevent situations in which hijacking can be used to modify the server's notions of the user's long-term secrets.

The session opened up by `ssh` is a shell running on the server with the user's full privileges, allowing, for example, the modification of the `~/.ssh/authorized_keys` file. This could allow an attacker to insert or remove entries in the list of public keys authorized for user login. Therefore, in order to take advantage of our construction for the full protection of long-term secrets, the session interface must be limited in what the user can operate on long-term secrets. All operations related to the integrity of the long-term secrets must be arranged so that they are carried out via the Vault. This implies that the list of public keys accepted for user login should only be updateable using a protocol that uses the Vault for user confirmation.

6 Discussion

As discussed in the introduction, widespread adoption of our design rests on several working assumptions, which we discuss in this section.

Application Adaptation. Application designers must make careful decisions about what data needs to be protected in their systems, and make changes to follow the protocol framework we propose. Although modifications are clearly needed, the advantage of being able to provide a more secured environment to the customers can be a major incentive. In addition, our implementation shows that such modifications are simple. For example, the applications using the SSH protocol discussed in Section 5.3 and refined in Appendix A require only minor changes on the code of `ssh`. For applications related to online commerce (Section 5.2), a bit more work needs to be done to add a mechanism to delegate handling of long-term secrets to the Vault application. Fortunately, the extensible nature of Firefox and other modern web browsers makes it straightforward to augment the browser with a Vault-aware extension. Our prototype extension has less than 60 lines of Javascript code, and our server side support totals only 200 lines of Python code.

Changes in User Behavior. Users must be willing to minimally change their behavior. We also do not expect this to be a significant hurdle to deployment, since the adjustments in behavior are small: the user must learn to switch to the Vault application for the entry of sensitive data. With appropriately designed web pages, the user can be explicitly prompted to perform this action, so the user does not need to explicitly commit this new interaction to memory.

Widespread Adoption of Virtualization. The Vault system fundamentally relies on a virtual machine monitor running on the hardware. Although we can only conjecture

about the future, we feel the prospects of widespread adoption of VMM technology are very good. Hardware support for virtualization is improving [11]. There are other compelling applications, such as mobility [20], intrusion detection [8], and software maintenance [22, 28] that will help drive the demand for virtualization technology.

7 Conclusion

In this paper, we present a novel design for using virtual machine technology to protect user sensitive information, such as passwords, credit card data, and cryptographic keys. Our approach makes use of the strong isolation guarantees of a virtual machine monitor (VMM) to separate an untrusted, commodity operating system from a trusted "Vault" handling long-term user secrets. We define a protocol framework that can be employed by any application to use the Vault for the safe handling of user long-term secrets.

We achieve several key design goals: we allow the users to continue to use commodity operating systems with arbitrary software configuration; we ensure that software in the untrusted environment cannot observe or tamper with the secrets stored in the Vault; and we ensure that the untrusted domain cannot make use of the Vault's long-term secrets without user confirmation. In addition, the changes to user experience are minimal.

Our implementation shows that this design is practical. We use the Xen virtual machine monitor to provide strong isolation guarantees and a trusted GUI. We also found that adapting existing applications to utilize the Vault requires only small changes.

Acknowledgments

The authors would like to thank Dirk Balfanz, Diana Smetters, and Hao Chi Wong for their valuable comments on earlier drafts of the paper. We would also like to thank Professor Ruby Lee of Princeton University for her encouragement and support.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[2] D. M. Chess and S. R. White. An undetectable computer virus. In *Virus Bulletin Conference*, Sept. 2002.

[3] F. Cohen. Computer viruses: theory and experiments. *Comput. Secur.*, 6(1):22–35, 1987.

[4] R. Cox, E. Grosse, R. Pike, D. L. Presotto, and S. Quinlan. Security in Plan 9. In *Proceedings of the 11th USENIX Security Symposium*, pages 3–16, Berkeley, CA, USA, 2002. USENIX Association.

[5] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, pages 77–88, New York, NY, USA, 2005. ACM Press.

[6] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, Jan. 1999.

[7] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 193–206, New York, NY, USA, 2003. ACM Press.

[8] T. Garfinkel and M. Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *Internet Society's 2003 Symposium on Network and Distributed System Security (NDSS)*, pages 191–206, 2003.

[9] R. P. Goldberg. Architectural principles for virtual computer systems. PhD thesis, Harvard University, 1972.

[10] P. Gutmann. An open-source cryptographic coprocessor. In *Proc. 9th USENIX Security Symposium*, Denver, CO, Aug 2000.

[11] Intel. Intel Virtualization Technology Specification for the IA-32 Intel Architecture, 2005.

[12] Intel. Lagrande technology. http://www.intel.com/technology/security, 2005.

[13] S. Jiang, S. Smith, and K. Minami. Securing web servers against insider attack. In *17th Annual Computer Security Applications Conference*, Dec. 2001.

[14] R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, and Z. Wang. Architecture for protecting critical secrets in microprocessors. In *ISCA '05: Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 2–13, June 2005.

[15] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz. Architectural support for copy and tamper resistant software. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 168–177, 2000.

[16] J. Marchesini, S. W. Smith, O. Wild, J. Stabiner, and A. Barsamian. Open-source applications of TCPA hardware. In *20th Annual Computer Security Applications Conference*, Dec. 2004.

[17] B. McCarty. Automated identity theft. *IEEE Security & Privacy Magazine*, 1(5), 2003.

[18] R. Meushaw, M. Schneider, D. Simard, and G. Wagner. Device for and method of secure computing using virtual machines. U.S. Patent no. 6,922,774, July 2005.

[19] R. Meushaw and D. Simard. NetTop: Commercial Technology in High Assurance Applications. NSA Tech Trend Notes, 9(4), http://www.vmware.com/pdf/TechTrendNotes.pdf, 2000.

[20] M. T. Raghunath, C. Narayanaswami, C. Carter, and R. Caceres. Reincarnating PCs with Portable SoulPads. Technical Report RC 23418, IBM Research, June 2005.

[21] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Proc. 14th USENIX Security Symposium*, Aug. 2005.

[22] C. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, M. S. Lam, and M. Rosenblum. Virtual appliances for deploying and maintaining software. In *Proceedings of the Seventeenth Large Installation Systems Administration Conference (LISA 2003)*, October 2003.

[23] S. W. Smith and S. Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31(9):831–860, Apr. 1999.

[24] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas. Design and implementation of the aegis single-chip secure processor using physical random functions. In *ISCA '05: Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 25–36, 2005.

[25] R. Ta-Min, L. Litty, and D. Lie. Splitting interfaces: Making trust between applications and operating systems configurable. In *Proc. 7th Symposium on Operating Systems Design and Implementation*, Nov 2006.

[26] Trusted Computing Group. TCG TPM Specification Version 1.2 Revision 85. http://www.trustedcomputinggroup.org, Feb. 2005.

[27] J. D. Tygar and B. Yee. Dyad: a system for using phyiscally secure coprocessors. In *Joint Harvard-MIT Workshop on Technological Strategies for the Protection of Intellectual Property in the Network Multimedia Environment*, Apr. 1993.
[28] VMware. VMware ACE. http://www.vmware.com/products/desktop/ace_features.html.
[29] X.org Foundation. Documentation. http://www.x.org/, 2005.
[30] T. Ylonen and C. Lonvick. SSH authentication protocol. IETF Internet Draft, Mar. 2005.

A Simplified Delegation Protocol Framework

The protocol framework described in Section 4.2 is applicable to all types of long-term secrets. In particular, the Vault-server secure tunnel allows the Vault application to transmit secrets to the server in their actual form. However, if the exchange does not involve the secrets in their actual form, the secure tunnel can be eliminated. This is usually the case in authentication systems that use public-private or shared secret keys. We define a streamlined protocol framework that eliminates the Vault-server connection. We use SSH authentication as an example as we describe the design of the protocol.

In challenge-response authentication schemes such as the SSH authentication protocol [30], the secret – the authentication private key – is never sent to the server in its actual form. Only an authenticator, which usually is a signature of a challenge, is sent to the server.

We begin by exploring a subtle attack: Consider a scenario where the user has accounts with servers A and B. With both servers, she uses the same public-private key pair for authentication. We also consider an attacker who attempts to impersonate the user by logging on one of these servers. With user confirmation in the Vault, the attacker cannot do so readily. However, the attacker can mislead the user into approving a bogus login attempt by carrying out a man-in-the-middle attack. Consider a compromised `ssh` client in the untrusted VM. The user uses it the login to A. Instead of requesting a challenge from A, the malicious `ssh` client requests one from B, and pass it to the Vault for the generation of a response. The attack succeeds because the user is misled to believing that she is approving for a login to A.

To overcome this attack, the user intention must be bound to the response. In this case, the user intends to login to server A. Recall, from Section 4.2, that both `sessionID` and `servername` are supplied by the untrusted VM and so they must be verified by the trusted entities. `sessionID` is verified by the server at Step 5, whilst `servername` is verified by the user using the server certificate of the secure tunnel between Vault and the server, at Step 6.

Since the Vault-Server tunnel is to be eliminated, there is no server certificate. Therefore, `servername` must be verified by some other means. We observed that `servername` can indeed be verified by the trusted `sshd` itself. It can be carried out as follows: (1) the challenge supplied by `ssh` must contain *both* `sessionID` and `servername`; (2) Vault displays `servername` for user confirmation; (3) Vault signs the challenge, binding the user confirmation of `servername` to the signature; (4) finally, the trusted `sshd` verified that the signature is intended for it by verifying the correct `sessionID` and `servername` are both embedded in the signature.

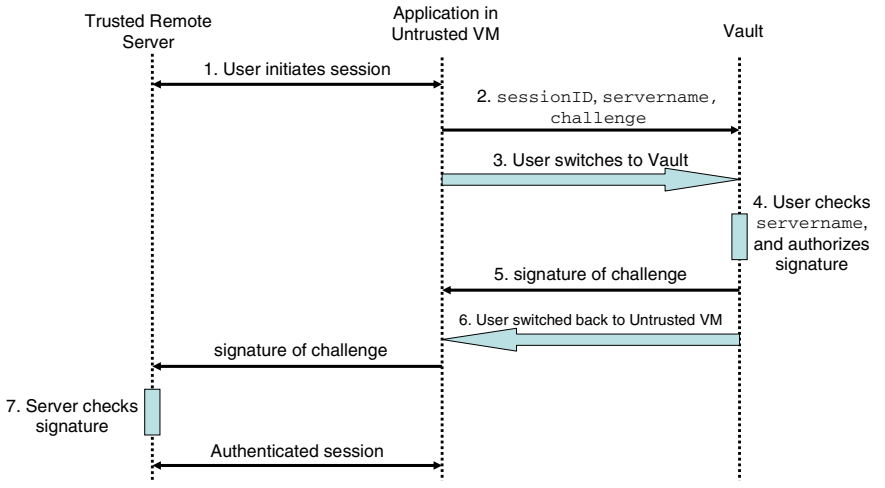


Fig. 5. The simplified protocol framework for challenge-response-based authentication protocols

The signature generated at Step 3 above can be sent to `sshd` via the untrusted VM, thus removing the need for a Vault-Server tunnel. The full simplified protocol is described below. Figure 5 illustrates it.

1. The user begins a session of interaction with the server via an application running in the untrusted VM. She proceeds to a point where she needs to generate a response to a cryptographic challenge.
2. The application sends the Vault `sessionID`, `servername`, and the challenge.
3. The user explicitly initiates the transition to the trusted VM. In its GUI, the Vault application displays `servername` for user confirmation.
4. Once the information is confirmed to be correct, the user authorizes the generation of a response using some long-term secrets, and incorporates in the response both `sessionID` and `servername`.
5. This response is passed back to the application in the untrusted VM for relaying to the remote server.
6. The Vault application signals the VMM to transition back to the untrusted VM.
7. The server verifies the response, `servername` and `sessionID`. If all are correct, it continues the session.

The above procedure is applicable to authentication protocols using public-private keys or any other shared cryptographic secrets. It is not applicable to long-term secrets that must be transmitted in their actual form, such as credit card numbers. For those the original protocol framework described in Section 4.2 should be used.

Formalization of RBAC Policy with Object Class Hierarchy

Jung Hwa Chae* and Nematollaah Shiri**

Concordia University
Dept. of Computer Science & Software Engineering
Montreal, Quebec, Canada
{chae,shiri}@cse.concordia.ca

Abstract. Formal methods and reasoning techniques can be useful tools for the representation and analysis of security policies and access control procedures. This paper presents a logical approach to representing and evaluating role-based access control (RBAC) policies, using description logics and a proof method, called tableaux. We propose a new variation of the RBAC model with a classification mechanism for objects. The key feature supported is the ability to model object classes, and class hierarchies used to restrict the validity and to control the propagation of authorization rules. We also demonstrate how access control decisions are made by tableaux, considering role and class hierarchies.

1 Introduction

Role-based access control (RBAC) models have been proposed to satisfy security requirements in complex systems with many users and many resources [1,2,3,4]. The central notion of RBAC is that users do not have direct access to objects, instead, permissions are assigned to a user according to his/her role in an organization. The explicit representation of roles makes it possible to simplify the system design and security management tasks in large systems.

This work aims to present a method to achieve further simplification in the security management tasks by the classification of objects. It uses the RBAC model as a basis and extends it by a classification mechanism for objects accessed in information system applications. Objects are classified into groups called object classes, and classes are organized into hierarchical structures. Once objects are categorized into groups, authorization tasks can be executed based on the classes instead of the individual objects. Object class hierarchy is a way to control the propagation of authorizations and to define boundaries for the validity of authorization rules. This modification of the RBAC model provides greater control and flexibility for the security administrative tasks. It also makes it easier to grant and revoke authorizations to entire groups of objects at a time.

* This work was supported by Institute for Information Technology Advancement (IITA) & Ministry of Information and Communication (MIC), Republic of Korea.
** This work was supported in part by Natural Science and Engineering Council (NSERC) of Canada.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 162–176, 2007.

We formally define the properties and relationships that should hold in the access control specifications, using description logics (DLs). A formal description of access control policies is necessary in order to check if security requirements are satisfied or not. Knowledge representation systems based on DLs have proven useful for representing the terminological knowledge of an application domain in a structured and formally well understood way [5,6]. These systems provide facilities to set up knowledge bases, to reason about their contents, and to manipulate them. A DL system not only stores terminologies and assertions, but also offers services that provide reasoning about them. In addition, a DL system is useful for validating the correctness and consistency of a knowledge base (to avoid redundant and conflicting policies). We use the DL language $\mathcal{ALCQI}$ [6] to define and reason about authorizations & privileges. In practice, we present an example of reasoning on access control via a decision method, called tableaux [7,8].

There has been much research on logical frameworks for the reasoning of access control models. Woo and Lam in [9] proposed a language to model authorization and control rules. A major issue in their approach was the tradeoff between expressiveness and efficiency. For the logical formalism approach, Jajodia et al. [10] proposed a logic-based language for specifying authorization rules. Massacci [11] introduced a logic for reasoning about RBAC, by extending the access control calculus in [12] to express role hierarchies. Their logic was mainly used to model concepts such as users, roles, and delegation. In [13], Rabitti et al. presented a model of authorization for next-generation database systems using the notion of implicit authorization. They developed an authorization model by including the properties of a class, class hierarchy, and composite objects. Bertino et al. [14] proposed a formal framework for reasoning about access control models. They introduced the concepts that subjects, objects, and privileges can be composed together in hierarchical structures and authorization can be derived along the hierarchies. A detailed description of implementing DL to reason about access control in a typical RBAC model is described by C. Zhao et al. in [15]. However, they formalized permissions as operations tied with objects. Moreover, their approach does not include the notions of classification of objects and class hierarchy.

While there has been a number of interesting research results describing roles and permissions, only a few of them consider the role-object relationships. There has been little work that study hierarchy for object classes in RBAC models. The idea of object classification for role-based policies was first introduced in [16]. Our approach is more pragmatic—complemented by a running prototype. It modifies the conventional role-based model, which leads to additional flexibility in access control administration and more reasoning power.

The rest of this paper is organized as follows: Section 2 gives a short overview of the RBAC model. In Section 3, we introduce the syntax and semantics of the DL language $\mathcal{ALCQI}$, and describes its associated reasoning tasks. We also illustrate the tableaux-based decision method. In Section 4, we describe how to build a DL knowledge base for the presented model in $\mathcal{ALCQI}$. In Section 5, we illustrate how user access control could be represented and how the reasoning

process proceeds upon user requests. We conclude the paper with a summary of the contributions and some suggestions for future research.

2 Role-Based Access Control

RBAC replaces direct user-permission associations in traditional access control policies with a combination of user-role and role-permission associations [17,18]. It defines a set of user assignments (UA) that relates each user to a set of roles and a set of permission assignments (PA), which connects each role to a set of privileges (see Figure 1).

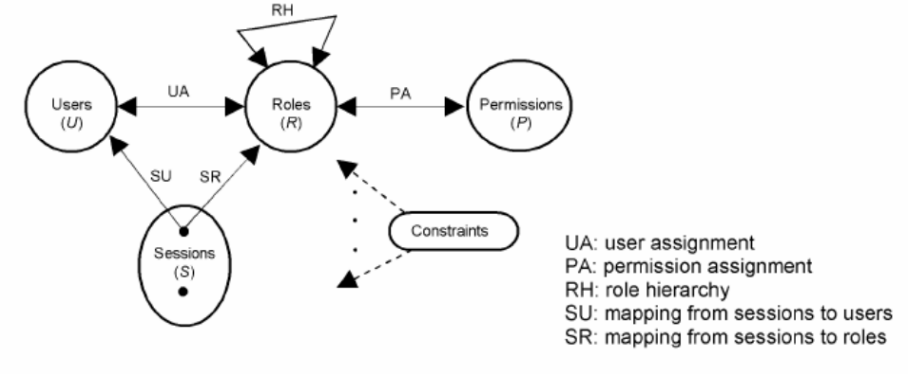


Fig. 1. RBAC model

Role-based policies provide a classification of users according to the activities they may execute. This approach simplifies security management by breaking user authorizations into two parts: one which assigns users to roles and one which associates access rights to objects for those roles. Analogously, one might expect to achieve further simplification in the security management if some classification is provided for objects. Objects could be classified according to their type or to their application area. Grouping objects into classes closely resembles the role concept. Figure 2 shows the proposed model, which consists of five entities including a set of objects and a set of classes. We also added a set of object assignments (OA) that relates each object to a set of classes.

Access authorizations of roles should then be defined based on the object classes. A role can be given the authorization to access all objects in a class, instead of giving explicit authorization for each individual object. Objects that are in the same class can be accessible for users with roles that have access right to that class. Ultimately, users exercise permissions on objects via roles to which they are assigned and classes to which the roles have access. We consider roles and object classes as mediators that let users exercise permission.

This modification of the RBAC model provides greater control and flexibility for the security administrative tasks. Furthermore, this approach makes the authorization management more simplified and easier; e.g., in order to add a new

object to the system, only the corresponding object assignment assertion should be included, whereas in the RBAC model, permission assignment should be explicitly given for each single role that has access privilege on the new object. Compared to roles, object classes have a greater potential for simplifying security administration since the number of objects in many systems is, in general, much larger than the number of subjects.

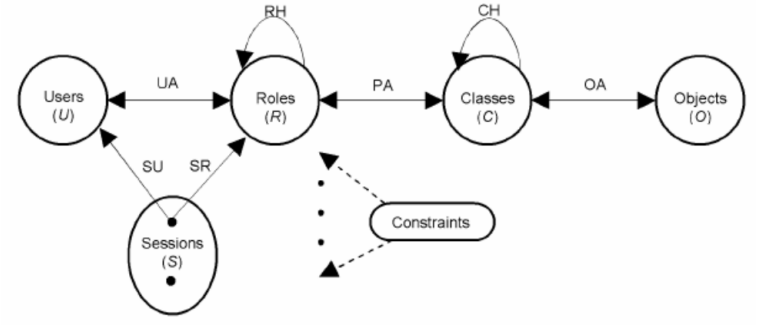


Fig. 2. Proposed modified RBAC model

2.1 Role Inheritance

In RBAC, roles are hierarchically organized into a role-subrole relationship that is called *role inheritance*. The hierarchy is interpreted using a graph where each node represents a role and a directed edge between two roles defines the implication of the authorization. Authorizations are implied along the edges of the role hierarchy. When role R_1 inherits from role R_2, denoted by $R_1 \geq R_2$, every user U explicitly assigned to R_1 is also implicitly associated with R_2; likewise, every permission p explicitly associated with role R_2 is implicitly associated with role R_1.

The role hierarchy is a partial order relation, which is reflexive, transitive, and antisymmetric. Inheritance is reflexive because a role inherits its own permissions; transitivity is a natural requirement in this context, and antisymmetry rules out cycles in the role hierarchy; i.e., roles that inherit from one another are disallowed.

2.2 Class Inheritance

In the proposed model, a set of objects are grouped together for security purposes. Each group is, in general, a set of individual objects, and is referred to as a class. Objects are associated with certain properties that can be used to construct groups for the authorization process. Examples of object properties are security levels, ownerships, classes (as in the object-oriented terminology), memberships, etc. Once the objects are categorized into finite sets of groups, authorization tasks can be executed based on the classes instead of individual objects.

Object classes are also organized into a hierarchical structure, called *class inheritance* (Note that the word class here is not used in the sense of object-oriented programming but represents any named group of objects). The hierarchy can be based on different criteria such as security levels, generalization and specialization associations, as in object oriented systems, and so on.

In the role inheritance, the concept of implied authorization is applied. The idea is to propagate the validity of authorization rule at some level in a hierarchy to its descendants [13]. Similarly, the same idea can be applied to object classes through a hierarchy. Class hierarchies coupled with role hierarchies are implemented in the reasoning process. The definition of object classes and its hierarchical structure provides more reasoning power compared to the conventional RBAC approach.

We propose the following authorization policies:

- Access to a class implies access to the objects explicitly assigned to that class;
- The class hierarchy is defined as follows: the relation $C_1 \geq_p C_2$ means that all roles given an access privilege p on class C_1 have the same access privilege on class C_2. Therefore, a user U who has a certain access to class C_1 is allowed to exercise the same access on class C_2. In general, the direction of the above inequality relation depends on the type of the operation; e.g., there may exist another operation denoted by p' for which the class inheritance relation between C_1 and C_2 would change to $C_2 \geq_{p'} C_1$; for example, read and write operations in mandatory policies where classes are formed based on the security level (access classes).

3 Description Logics and Reasoning

3.1 Description Logics

DLs are the family of logics that are well-suited to represent and provide reasoning about the knowledge of an application domain. The most expressive DL that we refer to in this paper is called $\mathcal{ALCQI}$. The basic elements of DLs are individuals, concepts, and roles, which respectively denote objects in the domain, sets of objects and binary relations. The set of constructors for concept expressions and role expressions considered in this work are listed in Table 1 and Table 2, respectively.

The interpretation function $\cdot^{\mathcal{I}}$ gives the semantics for individuals, concepts and roles in the application domain. The application domain is interpreted as $\Delta^{\mathcal{I}}$. The interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of the nonempty set of the interpreted application domain $\Delta^{\mathcal{I}}$, and the interpretation function $\cdot^{\mathcal{I}}$. Using this function, concepts are considered as subsets of $\Delta^{\mathcal{I}}$, roles as binary relations over $\Delta^{\mathcal{I}}$ and individuals as elements of $\Delta^{\mathcal{I}}$.

A knowledge base built using DLs is formed by two components: A TBox, which expresses intentional knowledge about classes and relations, and an ABox, which expresses extensional knowledge about individual objects. Formally, an

Table 1. Syntax and semantics of concept-forming constructors

Constructor Name	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjuction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
universal quantification	$\forall R.C$	$\{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
existential quantification	$\exists R.C$	$\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
collection of individuals	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

Table 2. Syntax and semantics of role-forming constructors

Constructor Name	Syntax	Semantics
atomic role	P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
role conjuction	$Q \sqcap R$	$Q^{\mathcal{I}} \cap R^{\mathcal{I}}$
inverse role	R^{-1}	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$

$\mathcal{ALCQI}$ knowledge base contains a finite set of inclusion assertions that are of the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 are arbitrary concept expressions. The notion of satisfaction of assertions determines the semantics of a knowledge base. The assertion $C_1 \sqsubseteq C_2$ is satisfied if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. An interpretation is said to be a model of a knowledge base if all of its assertions are satisfied. A knowledge base that admits a model is satisfiable. The following basic reasoning tasks are performed with respect to a given knowledge base:

- Knowledge base satisfiability; where we decide whether a knowledge base $\mathcal{K}$ admits at least one model (whether it is satisfiable);
- Concept consistency denoted by $\mathcal{K} \not\models C \equiv \bot$; where we decide whether a concept C is satisfiable in a given knowledge base $\mathcal{K}$, i.e., if $\mathcal{K}$ and C admit a common model;
- Concept subsumption or logical implication denoted by $\mathcal{K} \models C_1 \sqsubseteq C_2$; where we decide whether $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ holds for all models $\mathcal{M}$ of knowledge base $\mathcal{K}$.

All basic reasoning tasks are mutually reducible to each other [5,6]. For example, in order to prove the concept subsumption $\mathcal{K} \models C_1 \sqsubseteq C_2$, we can show that its negation $C_1 \sqcap \neg C_2$ is not satisfiable in any model $\mathcal{M}$ of knowledge base $\mathcal{K}$.

3.2 A Decision Method Based on Tableaux

The proof method of tableaux algorithms is based on satisfiability [7,8]. It can be used to test the satisfiability of concepts. The tableaux method builds a tree-like

model $\mathcal{M}$ of input concept C. In the tree $\mathcal{T}$, each node represents elements of $\Delta^{\mathcal{I}}$ labeled with the subconcepts of C, and each edge represents role-successorships between elements of $\Delta^{\mathcal{I}}$. The single root node x_0 in the tree $\mathcal{T}$ is initialized with C. Tableau rules are then repeatedly applied to node labels in an arbitrary order for as long as possible. In this process, labels can be extended, or the $\mathcal{T}$ structure can be extended or modified using the rules shown in Table 3.

Table 3. Completion rules for the logic of ALC

Rules	Formulas	Completion rules
Conjuction	$x \cdot \{C_1 \sqcap C_2, \ldots\}$	$x \cdot \{C_1 \sqcap C_2, C_1, C_2, \ldots\}$
Disjunction	$x \cdot \{C_1 \sqcup C_2, \ldots\}$	$x \cdot \{C_1 \sqcup C_2, C, \ldots\}$for $C \in \{C_1, C_2\}$
Existential restriction	$x \cdot \{\exists R.C, \ldots\}$	$x \cdot \{\exists R.C, \ldots\} \xrightarrow{R} y \cdot \{C\}$
Universal restriction	$x \cdot \{\forall R.C, \ldots\} \xrightarrow{R} y \cdot \{\ldots\}$	$x \cdot \{\forall R.C, \ldots\} \xrightarrow{R} y \cdot \{C, \ldots\}$

If a predecessor in the tree has a superset label then the rules can be blocked. It is said that tree $\mathcal{T}$ contains *clash* if there is a contradiction in some node label. If there is no rule applicable, then tree $\mathcal{T}$ is defined as *fully expanded*. The concept C is satisfiable iff $\mathcal{T}$ is a fully expanded, clash-free tree. In order to guarantee the termination of the reasoning process, we check if there is any cycle in the tree using blocking.

4 A Logic for Reasoning About Access Control

The TBox of a DL knowledge base $\mathcal{K}$ includes role inclusion axioms, class inclusion axioms, permission assignment axioms, and authorization axioms. The ABox of $\mathcal{K}$ includes the following assertions: role concept assertions, user concept assertions, class concept assertions, session concept assertions, role activation assertions, user role assignment assertions, object class classification assertions, and session creation assertions. Note that the term "role" has different meanings in RBAC and in DL. In RBAC, a role denotes a named job function within an organization. In DL, a role denotes a binary relationship between individuals.

4.1 Syntax

We introduced a collection of atomic concepts and atomic roles capturing the characters of RBAC. Let *User*, *Role*, *Class*, *Object*, and *Session* be atomic concepts that represent the users, roles, object classes, objects, and sessions, respectively. Let R be an atomic concept for each role r, where $r \in$ *Roles*, and let C be an atomic concept for each class c, where $c \in$ *Classes*. Here, the concept R is a subconcept of *Role*. Similarly, the concept C is a subconcept of *Class*. The concept expression $\exists \mathsf{assign}.R$ is adopted to represent the concept of "users that

Table 4. Atomic concepts and atomic roles

Atomic concepts and roles	Meaning
User, *Role*, *Class*, *Object*, and *Session*	atomic concept of users, roles, classes, objects, and sessions, respectively
R	atomic concept for each role r, where $r \in Roles$
C	atomic concept for each class c, where $c \in Classes$
assign	atomic role to connect users to roles
classify	atomic role to connect objects to classes
activate	atomic role to connect the session to the roles activated in it
canRead, canWrite, canExecute	atomic roles to associate roles to object classes in terms of read, write, and execute operations, respectively
authorizeRead, authorizeWrite, authorizeExecute	atomic roles to connect users to the authorized objects for read, write, and execute operations, respectively, based on the user's assigned roles

are assigned to the role R". Similarly, the concept expression $\mathsf{classify}.C$ represents the concept of "objects that are classified to the class C". We introduce the inverse relation $\mathsf{classify}^{-1}$. The expression $\exists\mathsf{classify}^{-1}.O$ is interpreted as the set of classes, where object O is categorized into that set of classes. The concept expression $\exists\mathsf{activate}.R$ denotes the concept of a set of sessions in which the role R is activated. Other concept expressions will be explained in subsequent sections. The atomic concepts and atomic roles that are considered in this paper are listed in Table 4.

4.2 Role Inclusion

Role hierarchies are represented by using inclusion axioms, which are of the form $C \sqsubseteq D$, where C and D are atomic concept for each role. Role C is interpreted as the set of users that are assigned to this role. As described in Section 2, the logical implication $C \sqsubseteq D$ is satisfied if its interpretation given, by $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, holds in each model $\mathcal{I}$ of the knowledge base $\mathcal{K}$. The relationship $R_1 \geq R_2$ is translated in DL to the role inclusion relation $R_1 \sqsubseteq R_2$. It indicates that R_1 subsumes the authorization for R_2.

4.3 Class Inclusion

The class hierarchy could also be represented by inclusion axioms. A class C is interpreted as a set of roles that have access to this class. This interpretation is consistent with the definition of subsumption or logical implication in DL, where $C_1 \sqsubseteq C_2$ has the same meaning as $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. Similarly, the relationship $C_1 \geq C_2$ is translated in DL to class inclusion relation $C_1 \sqsubseteq C_2$. The class hierarchy relation, similar to the role hierarchy relation, exhibits reflexive, transitive and antisymmetric properties.

4.4 Permission Assignment

In this subsection, binary relations canRead, canWrite, and canExecute are defined in order to associate RBAC roles with object classes. These binary relations represent operations such as Read, Write, and Execute, respectively. Therefore, the concept $\exists \mathsf{canRead}.C$ represents the concept of "roles associated with the Read operation on certain class C". It "connects" a role to a class on which the role has a Read permission. Similarly, $\exists \mathsf{canWrite}.C$ and $\exists \mathsf{canExecute}.C$ list all roles which are allowed to exercise Write and Execute operations, respectively, on class C. Moreover, the formula $\exists \mathsf{assign}.(\exists \mathsf{canRead}.(\exists \mathsf{classify}^{-1}.O))$ can be defined to represent the set of users assigned to at least one of the roles holding Read permissions on class C. Here, class C should include the object O.

4.5 Authorization Axioms

We now define authorization axioms that represent how users acquire the authorization to access objects according to their assigned roles and the object classifications. The concept expression $\exists \mathsf{authorizeRead}.O$ is interpreted as the set of users that are authorized to read a certain object O. It indirectly indicates users with a Read permission on object O via the user's assigned roles. According to the previous subsections, permissions are not directly connected either to users or to objects. The role's position as an intermediary lets a user exercise permission. In fact, it is the role that is assigned to permissions. The object class also plays a role as a mediator through which users can obtain authorization over objects. Roles have permissions over object classes. Objects will be involved in the permission assignment according to the classes that include them. The authorization axioms have the following form:

$$\exists \mathsf{assign}.(\exists \mathsf{canRead}.(\exists \mathsf{classify}^{-1}.O)) \sqsubseteq \exists \mathsf{authorizeRead}.O$$

This axiom indicates that all users assigned to at least one of the roles holding a Read permission on class C which includes object O are the users authorized to read object O.

Generally, a user that is already assigned to several roles can activate one or some of them, simultaneously. The instance of a user simultaneously exercising different roles is reflected by the concept of the session. Although a user can be assigned to several roles, he/she may be restricted to exercise some of them (in case of mutually exclusive roles) at the same time. The session is a useful concept when we consider dynamic separation of duty. The concept expression $\exists \mathsf{activate}.R$ indicates a set of sessions in which the role R is activated. The concept expression $\exists \mathsf{grantRead}.O$ denotes a set of sessions in which there is at least one activated role that has a Read permission on at least one class C that includes object O. The authorization within a session can be represented as follows:

$$\exists \mathsf{activate}.(\exists \mathsf{canRead}.(\exists \mathsf{classify}^{-1}.O)) \sqsubseteq \exists \mathsf{grantRead}.O$$

The above axiom asserts that sessions in which there is at least one activated role with a Read permission on class C that includes object O, are the sessions that have Read permission on object O.

Here, we have defined one atomic role for each operation, e.g., canRead, canWrite, canExecute. However, in [15] that describes a DL representation of RBAC, a single atomic role "cando" is defined and used to associate roles with permissions that are combinations of operations over individual objects; e.g., ReadDocs, WriteSrc, etc. It can be argued that if one needs to allow more operations in information systems, then using our approach, it is necessary to define more atomic roles corresponding to each operation in order to relate roles to classes. To address this issue, we first consider the structure of a TBox or the signature of the knowledge base. As already mentioned, a TBox includes role inclusion axioms, permission assignment axioms, and authorization axioms. We also include class inclusion axioms in the TBox. In order to add an operation in our model, a role (binary relation) of the form "canOperate" should be defined and the corresponding permission assignment axioms should be added to the TBox. The number of new permission assignment axioms is in the order of the number of roles multiplied by the number of object classes, $O(|Role| \times |Class|)$. In the approach described in [15], addition of a new operation increases the number of permissions in the system. Since permissions are of the form "OperationObject", e.g. ReadDocs , the number of permissions added to the system would be $O(|Object|)$. Each new permission concept requires the definition of the new permission assignment axioms. For each permission, the number of axioms is $O(|Role|)$. Therefore, the total number of permission assignment axioms added to the TBox in [15] would be $O(|Role| \times |Object|)$. However, the number of objects in a typical information system is much larger than the number of object classes. Hence, the number of permission assignments assertions added to the TBox in our model is much less than that in the RBAC model in [15].

We already discussed the advantages of our approach in Section 2, in terms of the reasoning power and authorization management when a new object is added to the information system. In conclusion, grouping objects into classes and giving privileges to classes using atomic roles for each operation is justified for its clear advantage in the management of the authorization tasks.

5 Example: RBAC Policies in DL

In this section, we illustrate the representation of RBAC policies using DL in our framework through an example. The tableaux method is used to evaluate the indicated request of a user.

We consider an information system that consists of five different roles: System-admin, Manager, OS-developer, Local-client, and Remote-client. Figure 3 shows the hierarchical relationships between various roles in the example. We use the following abbreviation to represent the concept of roles depicted in

Figure 3: SysAdmin, Mag, OSDev, LocCli, RemCli. The role hierarchy is modelled using the following inclusion axioms that are placed in the TBox of DL:

$$\text{SysAdmin} \sqsubseteq \text{Mag},\ \text{SysAdmin} \sqsubseteq \text{OSDev},\ \text{Mag} \sqsubseteq \text{LocCli},$$
$$\text{OSDev} \sqsubseteq \text{LocCli},\ \text{LocCli} \sqsubseteq \text{RemCli}$$

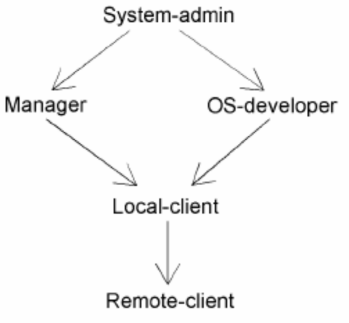


Fig. 3. Role hierarchy

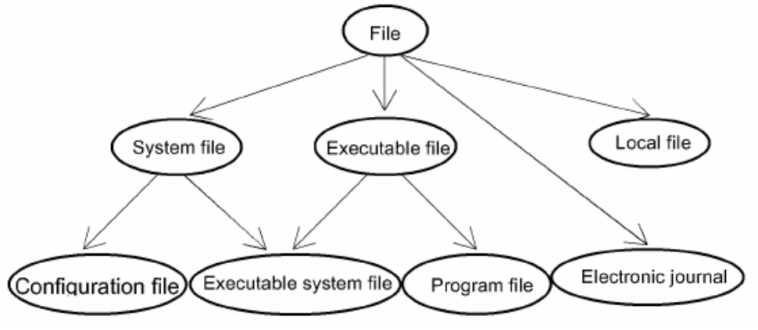


Fig. 4. Class hierarchy

Objects are classified into eight categories: Program-file, Electronic-journal, Configuration-file, Executable-system-file, Local-file, Executable-file, System-file, File. For each class shown in Figure 4, one distinct class concept is defined. Class concepts are represented by: ProFile, ElcJ, ConFile, ExeSysFile, LocFile, ExeFile, SysFile, and File. Read, Write, and Execute permissions are assigned to the roles according to Table 5. The inheritance relations among classes are given by the following class inclusion axioms that are part of the TBox of the DL model of the system:

$$\text{File} \sqsubseteq \text{SysFile},\ \text{File} \sqsubseteq \text{ExeSysFile},\ \text{File} \sqsubseteq \text{LocFile},$$
$$\text{File} \sqsubseteq \text{ElcJ},\ \text{SysFile} \sqsubseteq \text{ConFile},\ \text{SysFile} \sqsubseteq \text{ExeSysFile},$$
$$\text{ExeFile} \sqsubseteq \text{ExeSysFile},\ \text{ExeFile} \sqsubseteq \text{ProFile}$$

Table 5. Permission assignments for roles

	ElcJ	LocFile	ConFile	SysFile	ExeSysFile	ProFile	ExeFile	File
SysAdmin	r	r,w	r,w	r,w	r,w,x	r,w,x	r,w,x	r,w,x
Mag	r	r,w	r,w		x	x	x	
OSDev	r	r,w	r,w	r,w	r,w,x	r,w,x	r,w,x	
LocCli	r	r,w			x	x	x	
RemCli		r,w			x	x	x	

Permission assignment axioms for all roles that exist in the model are shown below. These axioms are also placed in the TBox.

$$\text{SysAdmin} \sqsubseteq \exists \mathsf{canRead}.\text{File},\ \text{SysAdmin} \sqsubseteq \exists \mathsf{canWrite}.\text{File},$$
$$\text{SysAdmin} \sqsubseteq \exists \mathsf{canExecute}.\text{File},\ \text{Mag} \sqsubseteq \exists \mathsf{canRead}.\text{ConFile},$$
$$\text{Mag} \sqsubseteq \exists \mathsf{canWrite}.\text{ConFile},\ \text{Mag} \sqsubseteq \exists \mathsf{canExecute}.\text{ExeFile},$$
$$\text{LocCli} \sqsubseteq \exists \mathsf{canRead}.\text{ElcJ},\ \text{RemCli} \sqsubseteq \exists \mathsf{canRead}.\text{LocFile},$$
$$\text{RemCli} \sqsubseteq \exists \mathsf{canWrite}.\text{LocFile},\ \text{RemCli} \sqsubseteq \exists \mathsf{canExecute}.\text{ExeFile}.$$

As already mentioned, the class hierarchy reduces the number of permission assignment axioms in the TBox; e.g., for System-admin, it is sufficient to specify permissions only over the class File. All permissions over other classes for System-admin can be implied using the class hierarchy. A similar inference capability based on the role hierarchy already exists in the RBAC model, e.g., the specification of permissions for Remote-client implicitly gives the same permissions to Local-client. However, the class hierarchy provides additional axioms that can be used together with the role hierarchy to enhance the reasoning power. The authorization axiom for Read operation is defined as follows:

$$\exists \mathsf{assign}.(\exists \mathsf{canRead}.(\exists \mathsf{classify}^{-1}.\mathit{object})) \sqsubseteq \exists \mathsf{authorizeRead}.\mathit{object}$$

Session is a very important concept when we consider the constraints such as the dynamic separation of duty and role cardinality. However, since the focus of our research is on how to represent object classes in role-based policies and how to incorporate class hierarchies for authorizations, we only address static aspects in the specification. We do not deal with the concept of session in this example.

In our example, we assume five different users: Alice, Bob, Charles, David, Edward. They are assigned to roles System-admin, Manager, OS-developer, Local-client, and Remote-client, respectively. The set of objects consists of five different files: $f1$, $f2$, $f3$, $f4$, and $f5$ that are classified into Configuration-file, Executable-system-file, Program-file, Electronic- journal, and Local-file, respectively. The class System-file includes $f1$ and $f2$, the class Executable-file includes $f2$ and $f3$, and the class File contains all of the objects. User assignment and object classification assertions are placed in the ABox. Suppose that Charles is explicitly assigned to an OS-developer role and he wishes to read the file $f1$. This request is equivalent to the following relation:

$$\text{Charles} \sqsubseteq \exists \mathsf{assign}.(\exists \mathsf{canRead}.(\exists \mathsf{classify}^{-1}.f1))$$

TBox: $SysFile \sqsubseteq ConFile$; $OSDev \sqsubseteq \exists \mathsf{canRead}.SysFile$.
ABox: ConFile $\sqsubseteq \exists \mathsf{classify}^{-1}.f1$; Charles $\sqsubseteq \exists \mathsf{assign}.\mathrm{OSDev}$.

Charles ; $\forall \mathsf{assign}.\forall \mathsf{canRead}.\forall \mathsf{classify}^{-1}.\neg f1$; $\neg \mathrm{ConFile} \sqcup \exists \mathsf{classify}^{-1}.f1$;
$\neg \mathrm{Charles} \sqcup \exists \mathsf{assign}.\mathrm{OSDev}$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcap \exists \mathsf{canRead}.SysFile$.
$\downarrow$
(**Model1**)
Charles; $\forall \mathsf{assign}.\forall \mathsf{canRead}.\forall \mathsf{classify}^{-1}.\neg f1$; $\neg \mathrm{ConFile} \sqcup \exists \mathsf{classify}^{-1}.f1$;
$\neg \mathrm{Charles} \sqcup \exists \mathsf{assign}.\mathrm{OSDev}$; $\exists \mathsf{assign}.\mathrm{OSDev}$; $\exists \mathsf{classify}^{-1}.f1$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcup \exists \mathsf{canRead}.SysFile$.
$\downarrow$ assign
Charles; $\forall \mathsf{canRead}.\forall \mathsf{classify}^{-1}.\neg f1$; $\neg \mathrm{ConFile} \sqcap \exists \mathsf{classify}^{-1}.f1$;
OSDev; $\exists \mathsf{classify}^{-1}.f1$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcup \exists \mathsf{canRead}.SysFile$; $\exists \mathsf{canRead}.SysFile$.
$\downarrow$ canRead
Charles; $\forall \mathsf{classify}^{-1}.\neg f1$; $\neg \mathrm{ConFile} \sqcup \exists \mathsf{classify}^{-1}.f1$;
$\neg \mathrm{Charles} \sqcup \exists \mathsf{assign}.\mathrm{OSDev}$; OSDev; $\exists \mathsf{classify}^{-1}.f1$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcup \exists \mathsf{canRead}.SysFile$; $SysFile$; $ConFile$.
$\downarrow$ $\mathsf{classify}^{-1}$
Charles; $\neg f1$; $\neg \mathrm{ConFile} \sqcup \exists \mathsf{classify}^{-1}.f1$; $\neg \mathrm{Charles} \sqcup \exists \mathsf{assign}.\mathrm{OSDev}$;
OSDev; $f1$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcup \exists \mathsf{canRead}.SysFile$; $SysFile$; $ConFile$.
$\bot$
(**Model2**)
Charles; $\forall \mathsf{assign}.\forall \mathsf{canRead}.\forall \mathsf{classify}^{-1}.\neg f1$; $\neg \mathrm{ConFile} \sqcup \exists \mathsf{classify}^{-1}.f1$;
$\neg \mathrm{Charles} \sqcup \exists \mathsf{assign}.\mathrm{OSDev}$; $\exists \mathsf{assign}.\mathrm{OSDev}$; $\neg \mathrm{ConFile}$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcup \exists \mathsf{canRead}.SysFile$; $\neg SysFile$.
$\downarrow$ assign
Charles; $\forall \mathsf{canRead}.\forall \mathsf{classify}^{-1}.\neg f1$; $\neg \mathrm{ConFile} \sqcup \exists \mathsf{classify}^{-1}.f1$;
$\neg Charles \sqcup \exists \mathsf{assign}.\mathrm{OSDev}$; OSDev; $\neg \mathrm{ConFile}$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcup \exists \mathsf{canRead}.SysFile$; $\exists \mathsf{canRead}.SysFile$;
$\neg SysFile$; $\exists \mathsf{canRead}.SysFile$.
$\downarrow$ canRead
Charles; $\forall \mathsf{classify}^{-1}.\neg f1$; $\neg \mathrm{ConFile} \sqcup \exists \mathsf{classify}^{-1}.f1$;
$\neg \mathrm{Charles} \sqcup \exists \mathsf{assign}.\mathrm{OSDev}$; OSDev; $\neg \mathrm{ConFile}$.
$\neg SysFile \sqcup ConFile$; $\neg OSDev \sqcup \exists \mathsf{canRead}.SysFile$; $SysFile$; $\neg SysFile$.
$\bot$

Fig. 5. Proving steps using the tableaux algorithm

Instead of proving the validity of above logical implication, we can prove that its negation is not satisfiable in any model that is built based on the axioms in the TBox and ABox. The negation of the above equation is given by:

$$\mathrm{Charles} \sqcap \forall \mathsf{assign}.(\forall \mathsf{canRead}.(\forall \mathsf{classify}^{-1}.\neg f1))$$

The reasoning process(shown in Figure 5) starts with the above relation. All axioms in the TBox are written with an italicized font. Models 1 & 2 correspond to the two longest branches of the tree built based on the given axioms. All other branches that are not shown here lead to the clashes in less number of steps than those that are shown for models 1 & 2. Since the negation is not satisfiable, the relation itself is valid and Charles's request should be granted. When a model includes both the role and its inverse, dynamic blocking method is used to verify whether a branch is blocked [19]. In Figure 5, object classification axioms are presented using $\mathsf{classify}^{-1}$ role instead of $\mathsf{classify}$. This simplifies the proof process by avoiding cycling that is required for dynamic blocking.

6 Conclusion and Future Work

In this paper, we demonstrated a formalization of an access-control policy model using a logical framework called DL. Such formalization allows one to make use of inference capabilities offered by DL to answer queries over the specification. In particular we have introduced a classification mechanism for objects and a notion of class hierarchies. Object class hierarchy is a way to control the propagation of authorizations and to define boundaries for the validity of authorization rules. This modification of the RBAC model provides greater control and flexibility for the security administrative tasks. A proof of concept specification has been written and tested with the tableaux method, as well. The object class hierarchy can be exploited in order to control the flow of information in RBAC. Mandatory policies, which are designed to restrict the information flow, are shown to be very strict in most of the information systems. Combining mandatory policies with RBAC via the object class hierarchies will result in more flexibility as well as control of information flow. This aspect will be investigated in the future work.

References

1. R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *IEEE Computer*, 29(2): 38–47, 1996.
2. D.F. Ferraiolo, J.F. Barkely, and D.R. Kuhn. A role based access control model and reference implementation within a corporate Intranet. *ACM Trans. Inf. Syst. Secur. (USA)*, 1(2): 34–64, 1999.
3. J.F. Barkely, V. Cincotta, D.F. Ferraiolo, S. Garrvrilla, and D.R. Kuhn. Role based access control for the world wide web. *NIST 20th National Computer Security Conference*, pages 331–340, 1997.
4. M. Koch, L.V. Mancini, and F. Parisi-Presicce. A graph-based formalism for RBAC. *ACM Trans. Inf. Syst. Secur. (USA)*, 5(3): 332–365, 2002.
5. F. Baader, D.L. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge university Press, Cambridge, United Kingdom, 2003.
6. D. Calvanese, G. De Giacomo, and M. Lenzerini. Description logics: foundations for class-based knowledge representation. *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 359–370, 2002.

7. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Stud. Log. (Netherlands)*, 69(1): 5–40, 2001.
8. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. *Principles of Knowledge Representation, Studies in Logic, Language and Information.* CSLI Publications, 1996.
9. T.Y.C. Woo and S.S. Lam. Authorization in distributed systems: a new approach. *J. Comput. Secur. (Netherlands)*, 2(2-3): 107–136, 1993.
10. S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst. (USA)*, 26(2): 214–260, 2001.
11. F. Massacci. Reasoning about security: A logic and a decision method for role-based access control. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, 1244: 421–435, 1997.
12. M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst. (USA)*, 15(4): 706–734, Sept. 1993.
13. F. Rabitti, E. Bertino, Won Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM Trans. Database Syst. (USA)*, 16(1): 88–131, 1991.
14. E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur. (USA)*, 6(1): 71–127, 2003.
15. C. Zhao, N. Heilili, S. Liu, and Z. Lin. Representation and reasoning on RBAC: a description logic approach. *Theoretical Aspects of Computing ICTAC 2005. Second International Colloquium. Proceedings (Lecture Notes in Computer Science Vol.3722)*, pages 381–393, 2005.
16. R.S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9): 40–48, 1994.
17. D.E. Ferraiolo, J.A. Cugini, and D.R. Kuhn. Role-based access control (RBAC): features and motivations. *Proceedings. 11th Annual Computer Security Applications Conference*, pages 241–248, 1995.
18. D.F. Ferraiolo, R. Sandhu, S. Gavrila, R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur. (USA)*, 4(3): 224–274, 2001.
19. I. Horrocks and U. Sattler. Description logics, basics, applications, and more. *Retrieved from http://www.cs.man.ac.uk/~horrocks/Slides/ecai-handout.pdf*, 2002.

Privacy-Preserving Credentials Upon Trusted Computing Augmented Servers

Yanjiang Yang, Robert H. Deng, and Feng Bao

School of Information Systems
Singapore Management University, Singapore 178902
Institute for Infocomm Research, Singapore 119613
{yjyang,robertdeng}@smu.edu.sg,baofeng@i2r.a-star.edu.sg

Abstract. Credentials are an indispensable means for service access control in electronic commerce. However, regular credentials such as X.509 certificates and SPKI/SDSI certificates do not address user privacy at all, while anonymous credentials that protect user privacy are complex and have compatibility problems with existing PKIs. In this paper we propose privacy-preserving credentials, a concept between regular credentials and anonymous credentials. The privacy-preserving credentials enjoy the advantageous features of both regular credentials and anonymous credentials, and strike a balance between user anonymity and system complexity. We achieve this by employing computer servers equipped with TPMs (Trusted Platform Modules). We present a detailed construction for ElGamal encryption credentials. We also present XML-based specification for the privacy-preserving credentials.

1 Introduction

It is well accepted that user privacy is an important issue in online services such as electronic commerce [1]. User privacy concerns actually result from the fact that online systems routinely enforce access control over the services they provide in order to distinguish qualified and illegitimate users, and current standard technologies implement access control/authorization through user identification. For example, a user provides her credential (e.g., a X.509 certificate) to a service provider in order to attest her qualification for the service in question. As a result, the service provider is enabled to log transactions and derive accurate dossiers of user activities.

Currently, there are mainly three kinds of credentials in PKI: X.509 certificates [17], SPKI/SDSI authorization certificates [12,23], and attribute certificates [18]. A X.509 certificate binds a public key to a globally unique user identity so as to enable the use of public keys at the discretion of user identities. X.509 certificates are thus known as *identity certificates*. In access control, however, it has been noted that a user's identity is almost never a factor in an authorization decision [11], and what really counts is whether the user has the required permissions. This gives rise to the concept of SPKI/SDSI *authorization certificate* and *attribute certificate*: an authorization certificate binds a set of user attributes that

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 177–192, 2007.

convey access permissions to a public key, while an attribute certificate binds user attributes to an identity. These standard credentials do not deal with user privacy[1].

Anonymous credentials (e.g., [7,6,9,8]) can be viewed as a special class of certificates for authorization. Instead of directly passing a credential to the verifier as with the regular credentials, use of anonymous credentials is through zero-knowledge proof protocols [14], where the verifier ends up learning whether or not the credential satisfy its access control policies but nothing beyond this fact. *Unlinkability* is a core feature of anonymous credentials, i.e., transactions using the same credential cannot be linked. While anonymous credentials offer strong user privacy protection, they have not been widely used in real world applications. A main reason was believed to be that they are not compatible with the existing PKIs [5]. Other reasons may attribute to the use of zero-knowledge proof techniques, which results in: (1) limited expressiveness. Zero-knowledge proof techniques are not effective in conveying complex relations between user attributes and access control policies; (2) low efficiency. Zero-knowledge proof techniques are in general expensive in terms of both computation and communication, and this makes it particularly difficult for resource-constraint users (e.g., wireless users) to use anonymous credentials.

Our Contributions. In this paper, we propose privacy-preserving credentials, which represent a concept between regular credentials and anonymous credentials. Specifically, the privacy-preserving credentials are built upon and thus compatible with regular credentials, but endeavor to achieve unlinkability as of anonymous credentials. For efficiency reasons, we avoid any zero-knowledge proof technique; rather, we manage to achieve *relaxed unlinkability* (see Section 3 for details). As a result, the privacy-preserving credentials enjoy the advantages of both regular credentials and anonymous credentials: compatibility with existing PKI and rich expressiveness, of regular credentials, and privacy-enhancing feature of anonymous credentials. Moreover, we implement partial disclosure of sensitive attributes based on "need to know".

A key challenge in constructing the privacy-preserving credentials lies in the public keys embedded in credentials, which are globally unique quantities. The public keys cannot be disclosed to the service providers, but they must still be usable for data encryption or data authentication. We solve this problem by running a specialized software program, PEM (Privacy Enhancing Module), at the sever side, which composes "chameleon public keys" by *blinding* the original public keys without compromising the usages of the keys. Trustworthiness of PEM is maintained through a TPM under the auspice of Trusted Computing Group (TCG) specifications [28] (more details on TCG/TPM are provided in Appendix). We design our protocol using TPM commands Version 1.2 [29].

Organization. We review related work in Section 2, followed by discussions on the concept, general construction, and security features of our privacy-preserving

[1] While a SPKI/SDSI authorization certificate does not necessarily contain a user identity, the public key associated with the certificate uniquely indicates a user, which *links* all the transactions under the same certificate.

credentials in Section 3. An instantiation of the credentials for ElGamal public encryption keys is presented in Section 4. We implement credential specification using XML in Section 5, and Section 6 concludes the paper.

2 Related Work

Our work is clearly closely related to anonymous credentials (e.g., [7,6,9,8]). Anonymous credentials achieve *unlinkability*, which to our belief is an essential factor for any privacy-enhancing technique. As such, the privacy-preserving credentials we propose are endeavored to provide unlinkability. However, in order not to compromise efficiency, we shall avoid zero-knowledge proof techniques in our construction, so the privacy-preserving credentials attain *relaxed unlinkability*, striking a balance between user anonymity and system complexity.

Trust negotiation (e.g.,[24,31]) is a procedure whereby a user and a server establish trust through a gradual exchange of the user's credential attributes and the server's access control policies. The technique is on the one hand to protect sensitive attributes of users from unqualified server, while on the other to protect the server's access control policies against illegitimate users. After a successful negotiation, the server obtains the user credential. In other words, trust negotiation is not meant to protect user privacy from qualified server. In contrast, our privacy-preserving credentials are designed to protect user privacy from the server, be it qualified or unqualified.

The Oblivious Attribute Certificates proposed in [20] work in such a way that a user gets a service iff the attributes stored in her certificate satisfy the policies of the server, yet the server learns nothing about these attribute values. While the Oblivious Attribute Certificates do not rely on zero-knowledge proof techniques, they still have the limitations of anonymous credentials such as restricted expressiveness and low efficiency. The objective of our privacy-preserving credentials is not concealing attribute values from the server, but disclosing only those satisfying "need to know". Other certificate-based access control relates to ours include Secret Handshakes [3], Hidden Credentials [15], and Oblivious Signature Based Envelope [19]. They however work in different ways: the server sends an encrypted message to a user, and the user can decrypt iff she has a certificate having attribute values specified by the server's access control policies; but the server does not learn whether or not the user has such a certificate. [5] suggested a novel method to extend standard attribute certificates so as to achieve user privacy, but the resultant certificates are essentially linkable.

3 Privacy-Preserving Credentials

3.1 Concept

Conceptually, a credential contains a subject together with a set of subject attributes specified by name/value pairs (e.g., Issuer/ca, Age/28, Role/professor), attached with a digital signature over the credential content; the signature is

generated by a Credential Issuer using his private key, and the authenticity of the credential can be verified by using the Issuer's public key. In the context of authorization, the objective of a credential is to attest that the subject indicated (directly or indirectly) by the public key possesses the specified attributes, and authorization decisions must be made to the public key based upon the attributes. We point out that the subject of a SPKI/SDSI authorization certificate is directly the public key contained in the certificate; while the subject of a X.509 certificate refers to the user identity in the certificate, but what is really effective in authorization is the public key, and the access control decisions are eventually granted to the public key. It is important to observe that in a credential, the public key, i.e., the credential subject, must be a globally unique quantity, whereas other attributes are not necessarily unique. For example, a SPKI/SDSI certificate includes attributes such as Issuer, Delegation, Authorization and Validity, but none of them would have values that are unique. This is logical since in virtually any application in practice, there must be a group of users share an attribute value or a combination of attribute values, and thus have the same permission. As a result, even a user discloses the attribute values in her credential to a server while without revealing her public key (and possibly other unique quantities), the server is still not able to accurately link the current transaction to the user's previous transactions.

Based upon this observation, we propose the concept of *privacy-preserving credentials* outlined in Figure 1. In particular, the privacy-preserving credentials represent a kind of authorization tokens between regular credentials and

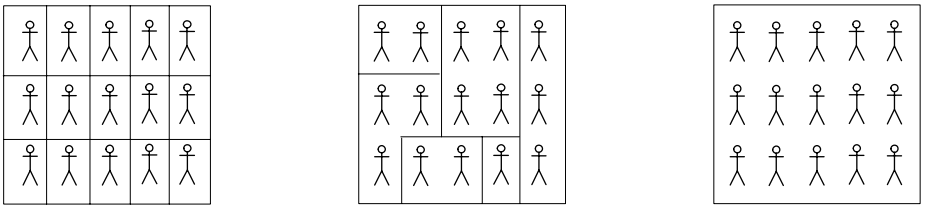


Fig. 1. Concept of Privacy-Preserving Credentials

anonymous credentials; regular credentials distinguish individual users, thereby not protecting user privacy at all (shown in Figure 1(a)), and anonymous credentials achieve unlinkability and recognize users as the whole user population, thereby fully protecting user privacy (shown in Figure 1(c)). The basic principle of the privacy-preserving credentials works as follows: the credential user does not reveal the credential subject (e.g., the public key and the user identity, and other unique quantities) to the verifier, and only discloses a minimal subset of attributes that satisfy "need to know" requirement of the verifier. As a result, the verifier distinguishes user cohorts among the whole user population (shown in

Figure 1(b)), where a cohort comprises users who have the same attribute values. In other words, the privacy-preserving credentials achieve *relaxed unlinkability* in the sense that the verifier can link an individual user to a cohort of users. The size of the cohorts relates only to the attribute values exposed to the verifier, and there exists the possibility that some particular users could be recognized as long as the size of the cohorts they belong to is one, but the majority of users cannot be differentiated (further discussion is given in subsection 4.3).

3.2 General Construction

The way we achieve privacy-preserving credentials is to use the credentials upon servers that are equipped with TCG-conformant TPMs. TPM at the server together with the Privacy Enhancing Module (PEM) constitutes a trusted computing platform that cannot be tampered with regardless of software or physical attacks (see Figure 2). PEM is a specialized software taking charge of enhancing user privacy, and users trust it to execute certain functions and not reveal information to the server. PEM is a protected application under the auspices of TPM. According to the TCG specifications, TPM takes integrity measurement of PEM and reports the integrity metrics to remote users through attestation. As a result, unless TPM is tampered with, the server cannot compromise PEM. It should be noted that while PEM colocates with the server, it essentially act as an extension of the user side.

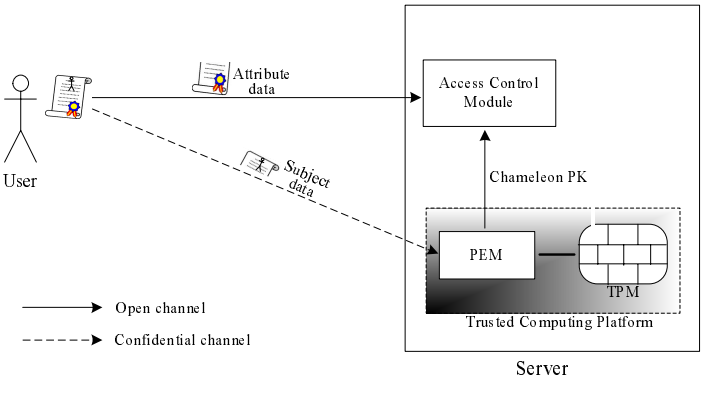


Fig. 2. General Construction of Privacy-Preserving Credentials

Our general construction works as follows. We partition the content of a credential into two parts: one is *subject data*, and the other is *attribute data*. The subject data, denoted as SubjDTA, include the data that uniquely indicate a subject, e.g., the public key, the unique user name if any, the digital signature over the credential (denoted as credSIG), and possibly some other data (e.g., the credential serial number if any, and some auxiliary data that are necessary for the verification of credSIG). The attribute data, denoted as AttrbDTA, include

all the subject attributes that affect access control decisions. As we made it clear the attribute data almost never uniquely identify a user, since it is unlikely that the attribute values in a credential are unique to a single user. As such, our way to achieve privacy-preserving credentials is that the attribute data are submitted to the server (precisely to the access control module that manages access control), but the subject data are given to PEM, shown in Figure 2.

The main task of PEM is to examine the validity of the credential, and derive a *chameleon public key* from the public key contained in SubjDTA and pass it to the access control module if the credential is valid. The access control module is an integral part of the server, responsible for evaluating the attribute values against its access control policies and making the final authorization decision. In our system, the server actually entrusts *validation of credentials* to PEM, but still takes the full responsibility in enforcing access control; and PEM does not in any way involve into the enforcement of access control. It is important to note that while PEM takes root in TPM, it still uses the resources (computation and storage) of the server platform for execution, so there is no efficiency penalty upon PEM. TPM does not perform any application-specific function, only taking charge of keeping the trusted state of PEM. Therefore, although TPM is strongly limited by its computation and storage capability, the overall system does not subject to the hardware constraint of the coprocessor.

3.3 Security Features

We desire the following security features upon the privacy-preserving credentials in the above construction.

- *Unforgeability*: Unforgeability is a fundamental feature of any credential system, which requires that nobody other than the Credential Issuer can issue valid credentials.
- *Partial disclosure of attributes*: A credential normally includes some sensitive attributes, and the credential user may be reluctant to reveal them to the verifier beyond "need to know". A user thus should be enabled to choose to disclose only the attributes that are absolutely necessary for the fulfilment of the server's access control requirements. The server should not be able to learn the hidden attribute values.
- *Relaxed Unlinkability*: The server should not be able to link transactions by inspecting the subject data that could obviously lead to linkability, and the extent of linkability is only dependent on the attribute values disclosed by the user. This suggests that the channel from the user to PEM (dote line in Figure 2) must be confidential against the server.
- *Usability of public keys*: In certificate-based access control, an authorization decision is often made to the public key contained in a credential, which is either for the purpose of data encryption or data authentication. We know that for any online service, access control is the first step whereby the server determines whether the user who uses a credential has the permission to the service in question; and what after access control is the *service provision*

procedure, where the public key of the credential must be used for either data encryption or verification of data signed by the user. The privacy-preserving credentials thus must not disable the usage of the public keys.

Goals of Adversary: Adversary behaviors towards the privacy-preserving credentials include: users may wish to break the unforgeability feature to forge credentials, and to defeat non-repudiation of digital signatures; the server may attempt to compromise partial disclosure of attributes by inferring the attribute values of hidden attributes, as well as to compromise relaxed unlinkability by linking individual users.

4 Concrete Instantiation

In this section, we give a concrete instantiation of the privacy-preserving credentials upon TPM-augmented servers, according to the above general construction. We know that the public key contained in a credential may correspond to either public key encryption or digital signature, but for limit of space we only instantiate ElGamal type digital signature credentials. Our instantiation can also be extended to Elgamal public key encryption credentials and even RSA credentials.

4.1 Preliminaries

We shall use the following notations in the sequel. p, q, g are parameters of ElGamal public key encryption scheme, where p, q are two large primes such that $q|p-1$, and $g \in Z_p^*$ is of order q. $h(.)$ is a collision resistant hash function such as SHA-1. $\{.\}_k$ denote secret key encryption by a secret key k. $\mathcal{E}_{PK}(.)$ denotes public key encryption by a public key PK, and $\mathcal{S}_{SK}(.)$ denotes signature signing by a private key SK.

ElGamal Public Key Encryption. A user has a key pair $(PK = y, SK = x)$, where $y = g^x \pmod p$ is the public key and x is the private key. The encryption of a message m generates a ciphertext (c_1, c_2), where $c_1 = g^t \pmod p$, $c_2 = my^t = mg^{xt} \pmod p$, with $t \in_R Z_q$. The decryption by the user using x works as $c_2/c_1^x = mg^{xt}/g^{tx} = m \pmod p$.

Merkle Hash Tree. The Merkle hash tree [22] is an efficient method to authenticate a set of data in such a way that given a signature over the whole data set along with some *auxiliary authenticating data*, a subset of data can be verified while in the absence of the remaining data. We illustrate the Merkle hash tree by a simple example shown in Figure 3, which is to authenticate a set of data $\{d_1, d_2, d_3, d_4\}$.

The construction of the Merkle hash tree is as follows: each leaf node of the tree is assigned a hash value of a datum, so the values represented by the leaf nodes are $h_1 = h(d_1), h_2 = h(d_2), h_3 = h(d_3), h_4 = h(d_4)$, respectively. The value of each internal node including the root node is derived from its child nodes. For example,

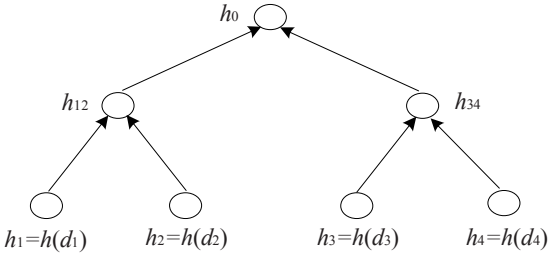


Fig. 3. Construction of Merkle Hash Tree

the value of node h_{12} is $h_{12} = h(h_1||h_2)$, where $h(.)$ is a collision-resistant one-way hash function and $||$ denotes concatenation. Similarly, $h_{34} = h(h_3||h_4)$ and $h_0 = h(h_{12}||h_{34})$. With a signature issued upon the root value h_0, any subset of the data set can be authenticated with the help of some auxiliary authenticating data while without disclosing the remaining data. For example, d_1 can be authenticated by given the authenticating data h_2 derived from d_2 and h_{34} derived from d_3 and d_4, while in the absence of d_2, d_3 and d_4; d_1 and d_2 can be authenticated if the authenticating data h_{34} is given, while without knowing d_3 and d_4. The efficiency of the Merkle hash tree rests with the fact that the number of the authenticating data is linear to $\log_2 N$, where N is the size of the whole data set. It is clear that given a root value of a data set, it is computationally infeasible to find a different data set that has the same root value, which amounts to the security of the Merkle hash tree method.

4.2 Protocol

Security Assumptions

- The hardware layer of TPM is tamper resistant regardless of hardware attacks and software attacks by any party.
- PEM is running in a protected execution environment, within which different applications run in isolation, free from observed or compromised by other processes running in the same protected partition, or by processes in any insecure partition that may exit in parallel. TPM defined by the TCG specifications itself does not suffice to afford this kind of protected execution environments, but a TPM with slightly extended mechanisms, such as the Intel's LaGrande Technology (LT) [16], can achieve this objective.

Overview. Let us first give some insights on our instantiation. First, the main challenge in constructing the privacy-preserving credentials is to simultaneously achieve relaxed unlinkability and usability of public keys. The feature of relaxed unlinkability requires the public key in a credential to be hidden from the server, while the feature of usability of public keys suggests the server must use the public key in the subsequent service provision procedure for either data encryption or

data authentication. Our solution is that PEM composes and gives a "chameleon public key" by "blinding" the actual public key to the server such that the usability of the public key is enabled, yet the server is not able to compute the actual public key.

Second, to enable a user to selectively disclose credential attributes (i.e., partial disclosure of attributes), we organize the content of a credential into a Merkle hash tree, and the credential signature credSIG by the Credential Issuer is issued upon the root value of the Merkle hash tree. Note that credSIG is a unique quantity, so it must be hidden from the server. In fact, it is included in SubjDTA, and never revealed to the server.

Third, TPM is responsible for integrity measurement and reporting of the platform including the protected software, PEM. In particular, PCR values of TPM record the integrity metrics of the platform from booting, to loading of operation system, to loading of PEM. Before sending a credential to the server, a user must first make sure that the protected computing platform is running in the expected status. TPM reports the platform configuration and status through *platform attestation* (it will be clear shortly how our protocols implement platform attestation).

Finally, recall the general construction that to achieve relaxed unlinkability, the subject data sent to PEM must be through a confidential channel against the server. We thus suppose PEM has a certified key pair (PK_{PEM}, SK_{PEM}) that corresponds to a standard public key encryption scheme, so that one can encrypt and send messages to it using the public key, and the server cannot decrypt and learn the messages. To achieve better security, we protect the secret key SK_{PEM} by sealed storage of TPM (invoking TPM_Seal to seal SK_{PEM}), and the integrity metrics of the platform is bound with the seal.

Based on these ideas, we next give a protocol to construct the privacy-preserving credentials that contain ElGamal public keys for encryption.

ElGamal Public Key Encryption Credentials. We suppose a user Alice has an ElGamal key pair $(PK_A = y = g^x \pmod p, SK_A = x)$, and the public key contained in her credential is thus PK_A. Moreover, without loss of generality, we suppose Alice needs to submit a subset of her credential attributes to a server in order to access a service. In such a case, the attribute data AttrbDTA comprises this subset of attribute values, while the auxiliary authenticating data that are derived from the hidden attribute values should be included in the subject data SubjDTA (see the general construction). We have to hide the auxiliary authenticating data of the hidden attributes from the server, since otherwise the server could infer the hidden attribute values in case the domains of the hidden attributes are small. To see this, suppose the server is given the auxiliary authenticating data d_{34} in order to authenticate d_1 and d_2 in Figure 3. While the server is not able to directly get d_3 and d_4 from d_{34}, it can enumerate every value in the domains of d_3 and d_4 to find out the actual values that amount to d_{34}, as long as the domains are small.

For a public key encryption credential, the public key PK_A will be used by the server to send sensitive data to Alice in the subsequent service provision

procedure. In our context, the server should not directly get PK_A, and PEM composes a chameleon public key in order to conceal PK_A. The protocol for platform attestation and credential processing is described as follows ($A \rightarrow B : m$ denotes that entity A sends m to B).

1. Alice → PEM: Attestation Request, R_A. R_A is a nonce generated by Alice.
2. PEM → TPM: TPM_Quote($h(R_A||ID_S)||indx(I)$). The TPM_Quote command instructs TPM to attest the platform status. The parameters given to this command include the indices of the PCRs that record the platform integrity metrics, I. TPM_Quote may also be given 160 bits of externally supplied data which, in our case, is the hash value of R_A and the server ID ID_S.
3. TPM → PEM: $\mathcal{S}_{AIK}(h(R_A||ID_S)||I)$. TPM returns a signature upon $h(R_A||ID_S)||I$ issued using its AIK.
4. PEM → Alice: I, $\mathcal{S}_{AIK}(h(R_A||ID_S)||I)$, AIK certificate. PEM sends platform integrity metrics I, the signature, and AIK certificate to Alice.
5. Alice: first checks the validity of $\mathcal{S}_{AIK}(h(R_A||ID_S)||I)$; then checks $h(R_A||ID_S)$ to ensure the message is fresh; finally decides whether the integrity metrics I represents a trustworthy state of the platform. From I, Alice can know whether PEM has been compromised or not, and whether it is running as expected.
6. Alice → PEM: AttrbDTA, c. If all checks pass, Alice encrypts SubjDTA using PK_{PEM}, the public key of PEM, to generate $c = (\mathcal{E}_{PK_{PEM}}(k), \{\text{SubjDTA}\}_k)$, where k is a random secret key for a standard secret key encryption scheme. Then Alice sends AttrbDTA and c to PEM.
7. PEM → TPM: TPM_Unseal(SK_{PEM}). PEM instructs TPM to unseal SK_{PEM}. SK_{PEM} is unsealed only if the platform is in the agreed state, i.e., the integrity metrics I matches the PCR values stored together with the protected SK_{PEM} at the time TPM_Seal was invoked.
8. TPM → PEM: SK_{PEM}. TPM returns SK_{PEM} to PEM.
9. PEM: decrypts c using SK_{PEM} to get k, and decrypts $\{\text{SubjDTA}\}_k$ using k to get SubjDTA; then verifies the authenticity of the credential by checking the validity of credSIG included in SubjDTA:
   (a) credSIG is valid: PEM picks a random blinding element $\alpha \in_R Z_q$ and composes a chameleon public key $PK'_A = g^{\alpha} PK_A = g^{x+\alpha} \pmod{p}$; it then encrypts α using PK_A to generate $c' = \mathcal{E}_{PK_A}(\alpha)$, and sends AttrbDTA, PK'_A, c', and VALID (a symbol indicating the credential is valid), to the server (precisely to the access control module of the server).
   (b) credSIG is invalid: PEM sends INVALID (a symbol indicating the credential is invalid) to the server.
10. Server: If receiving INVALID, it simply rejects Alice and aborts the transaction; otherwise, it evaluates AttrbDTA against its access control policies to determine whether the permission is granted to the user. If the permission is granted, the server sends c' to the user, and will use PK'_A for data encryption in the subsequent service provision.

11. Alice: if granted access permission and receiving c', she decrypts c' to get α, and computes and uses $SK'_A = SK_A + \alpha = x + \alpha \pmod{q}$ as her private key in the service provision procedure. (PK'_A, SK'_A) clearly constitutes a valid ElGamal key pair.

4.3 Security Analysis

We next discuss how the above instantiation achieve the security features in Section 3.

Unforgeability. Unforgeability of credentials is trivial due to the security of the Merkle hash tree and the digital signature of the Credential Issuer.

Partial disclosure of attributes. The feature of partial disclosure of attributes is also clear, since a credential is signed by the Credential Issuer upon the root value of the Merkle hash tree organized by the content of the credential, so the user is enabled to only reveal a subset of attributes that satisfy the server's access control policies. Furthermore, the server cannot see the auxiliary authenticating data derived from the hidden attributes, thereby learning nothing on the hidden attributes even their domains are small.

Relaxed Unlinkability. Clearly, relaxed unlinkability is achieved by the approach that the server is not allowed to inspect the subject data SubjDTA that obviously leads to linkability (all unique quantities including credSIG are included in SubjDTA). From the instantiation, the extent of linkability depends totally on the attribute values contained in AttrbDTA, since the chameleon public key and the ciphertext of the blinding element are random quantities. "Relaxability" (relaxed unlinkability) comes from the fact that users would be linked to cohorts, each consists of users having the same attribute values. It is important to note that we do not rule out the possibility that some particular users can be linked, in which case the size of the cohort a user belongs to is 1. For example, a particular user may have a set of attribute values distinct from anyone else. We next give an analysis on the conditions under which no individual user is linked.

Suppose a type of privacy-preserving credentials has κ attributes (attribute 1 to attribute κ), and attribute i takes v_i values, $i = 1..\kappa$. Note that these v_i values do not necessarily constitute the domain of attribute i, and they are the values that are actually assigned to users. There are thus $\prod_{i=1}^{\kappa} v_i$ combinations of attribute values in total, and clearly each combination determines a cohort. In order that no individual user is linked, there must be at least two users to take every combination of the attribute values, i.e., the size of every cohort must be at least 2. As such, there are at least $2\prod_{i=1}^{\kappa} v_i$ users. Further, consider each particular attribute: for each of the v_i values of attribute i, it must occur at least $2v_1...v_{i-1}v_{i+1}...v_\kappa = 2\prod_{j=1,j\neq i}^{\kappa} v_j$ times when in combination with the remaining attributes, which suggests that there must be at least $2\prod_{j=1,j\neq i}^{\kappa} v_j$ users to take the value. As a result, we have the following claim.

CLAIM. *There would be no individual user be linked as long as the following conditions are satisfied:*

1. *There are at least* $2\prod_{i=1}^{\kappa} v_i$ *users registered to the Credential Issuer; and*
2. *For each value of the* v_i *values of attribute* i, $i = 1..\kappa$, *there are at least* $2\prod_{j=1,j\neq i}^{\kappa} v_j$ *users taking the value.*

Of course, the second condition already implies the first one. These conditions can be used as a criteria to evaluate the relaxed unlinkability of the privacy-preserving credentials.

Usability of public keys. The usability of public keys is determined by the chameleon public keys and the corresponding private keys. It can be easily seen that what we construct in the above instantiation is a valid ElGamal encryption key pair.

4.4 Discussions

We discuss the advantages and limits of the privacy-preserving credentials. Our constructions do not use any zero-knowledge proof technique, hence the privacy-preserving credentials have efficiency advantage over anonymous credentials. We do not give the exact comparison result, as this depends on the specific anonymous credential schemes to be compared, but a casual estimate on the overhead of the privacy-preserving credentials is simply several operations of signature generation/verification and encryption/decryption (note that platform attestation involves essentially no more than one signature signing operation and one signature verification operation.). Moreover, there is no major architectural change at the user side, so we believe resource-constraint users such as mobile devices will not be affected in our system. A more appealing advantage is that since the privacy-preserving credentials are totally compatible with regular credentials, they have no expressiveness problem; more importantly, this makes them implementable directly upon existing standard PKIs. In contrast, a major limit of anonymous credentials is their incompatibility with PKIs.

The main limit of the privacy-preserving credentials we can imagine is that they have to use TPM at the server side. It should be noted that TPM can only be trusted up to the level of its hardware tamper resistance, and should be assumed to deter only the least resourceful attackers [13]. On the bright side however, numerous techniques to take hardware tamper resistance and the threat from the local host users into account in the design of trusted systems have been studied extensively, and much progress has been made in recent years (e.g., [21,25,26,27]). It is thus reasonable to expect that as tamper resistant hardware becomes more widely adopted, high quality tamper resistant hardware will become affordable due to economy of scale.

5 Credential Specification

We implement XML-based credential specification for the privacy-preserving credentials, compatible with the structure of regular credentials such as X.509 certificates and SPKI/SDSI authorization certificates. Observe that while the

```
<DOCTYPE CashBank_Customer[
   <!ELEMENT CashBank_Customer(issuer, validity, profession,
         city, dateBirth, credLevel, cusID, publicKey, credSIG, extension)>
   <!ELEMENT issuer ANY>
   <!ELEMENT validity (#PCDATA|NULL)>
   <!ELEMENT profession (#PCDATA|NULL)>
   <!ELEMENT city (#PCDATA|NULL)>
   <!ELEMENT dateBirth (#PCDATA|NULL)>
   <!ELEMENT credLevel (HIGH|MEDIUM|LOW|NULL)>
   <!ELEMENT extension (extAttr*)>
   <!ELEMENT extAttr ANY>
   <!ATTLIST extAttr attrSeq CDATA #REQUIRED
      attrV CDATA #REQUIRED>
   <!ATTLIST issuer XML:LINK CDATA #FIXED "SIMPLE"
      HREF CDATA #REQUIRED
      signKey CDATA #REQUIRED>
      signAlg CDATA #REQUIRED>
   <!ATTLIST CashBank_Customer cusID ID NULL>
   <!ATTLIST CashBank_Customer publicKey CDATA NULL)>
   <!ATTLIST CashBank_Customer credSIG CDATA NULL)>
]>
```

```
<CashBank_Customer cusID='NULL' publicKey='NULL' credSIG='NLL'>
<issuer HREF='http://www.cashbank.com'
   signKey='2ABG64897HJ' signAlg='RSA'>
<validity> 01-10-1006 </validity>
<profession> software_engineer </profesion>
<city> NULL </city>
<dateBirth> NULL </dateBirth>
<credLevel> MEDIUM </credLevel>
<extension>
   <extAttr attrSeq= '4' attrV='♯◇†‖⌜‡⌟↯♭ℓ'/>
   <extAttr attrSeq= '5' attrV='↯®‖ℑØ§∃℘§⌜£ℑ'/>
   <extAttr attrSeq= '7' attrV='⌜Ø©ℜ√⊤§‡¶✠'/>
   <extAttr attrSeq= '8' attrV='¶ℓ℘ℜℑ⊤♣♯♮†'/>
   <extAttr attrSeq= '9' attrV='∞□△ℵℏ∂♡℧'/>
</extension>
</CashBank_Customer>
```

Fig. 4. Example of Privacy-Preserving Template and Credential

exact fields contained in regular credentials may be different, they have similar structure, e.g., they usually include fields such as Serial Number, Issuer, Validity Period, Subject Name, Public key, etc., and an Extension field. Our basic idea for constructing the privacy-preserving credentials is utilizing the Extension field to encode the data to be submitted to PEM. While we can directly extend the X.509 certificates or the SPKI/SDSI certificates by placing all the application-specific attributes and the data intended for PEM in the Extension field, the examples we give below do not follow this method, simply for illustration purposes. XML [30] has extensive support in practice, so the implementation of XML-based specification entitles the privacy-preserving credentials wider applicability. For instance, we can use the privacy-preserving credentials in a trust negotiation system that enforces P3P privacy policies.

To simplify the management of credentials, we define *credential templates*. A credential template specifies a type of credentials specific to a particular application. We model a credential template as a XML DTD [30]. The upper part of Figure 4 shows an example of a privacy-preserving credential template CashBank_Customer. To facilitate partial disclosure of attributes, template fields are

assigned a default value NULL, which is a special symbol indicating the field may be concealed from the server.

A credential is an instance of the credential template, specifying the attribute values that characterize a user. A privacy-preserving credential is thus a valid XML document conforming to the corresponding DTD credential template. The lower part of Figure 4 gives an instance of the CashBank_Constomer template.

The attributes to be concealed from the server are assigned the special symbol NULL, and the auxiliary authenticating data derived from them according to the Merkle hash tree are encoded in the "extension" field. To simplify credential parsing, each piece of auxiliary authenticating data is encrypted as a separate extended attribute "extAttr". In particular, the example credential in Figure 4 (lower part) is as follows: the cleartexts for cusID, publicKey, and credSIG are removed, and the ciphertexts of them are encoded in the extension filed as "attrSeq= '7' attrV='⌜∅©ℜ√⊤§‡¶✠"', "attrSeq= '8' attrV='¶ℓ℘ℜℑ⊤♣♯♮†"', and "attrSeq= '9' attrV='∞□△ℵℏ∂♡℧"', respectively; the ciphertext of the hidden attribute "city" is represented by "attrSeq= '4' attrV='♯◇†‖⌜‡⊥♮♭ℓ"' and the hidden attribute "dataBirth" is encoded as "attrSeq= '5' attrV='♮®‖ℑ∅§∃℘§⌜£ℑ"'.

6 Conclusions

The new initiatives of trusted computing by placing TPM at the server machine are a promising paradigm in addressing user privacy in online services. Upon such servers, we proposed privacy-preserving credentials that represent a concept between regular credentials and anonymous credentials, in the sense that the privacy-preserving credentials are compatible with regular credentials while incorporating the privacy-enhancing features of anonymous credentials. We gave concrete construction of the privacy-preserving credentials containing ElGamal encryption keys for data encryption. We also implemented XML-based credential specification.

References

1. A. Acquisti. Privacy in Electronic Commerce and the Economics of Immediate Gratification. *Proc. ACM. Electronic Commerce (EC 04)*, pp. 21-29, 2004.
2. E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation, *Proc. ACM Conference on Computer and Communications Security, CCS'04*, pp. 132-145, 2004.
3. D. Balfanz, et al. Secret HandShakes from Pairing-based Key Agreement. *Proc. IEEE Symposium on Security and Privacy*, pp. 180-196, 2003.
4. R. Bradshaw, J. Hlot, K. Seamons. Concealing Complex Policies with Hidden Credentials. *Proc. ACM Conference on Computer and Communication Security*, pp. 146-157, 2004.
5. V. Benjumea, J. Lopez, J. A. Montenegro, and J. M. Troya, A First Approach to Provide Anonymity in Attribute Certificate. *Proc. Public Key Cryptography, PKC'04*, LNCS 2947, pp. 402-415, 2004.

6. S. Brands. Rethinking Public Key Infrastructures and Digital Certificates: Building in Privay. MIT Press, 2000.
7. D. Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete, *Communications of the ACM*, Vol 28, No. 10, pp. 1030-1044, 1985.
8. J. Camenisch and E. V. Herreweghen, Design and Implementation of the Idemix Anonymous Credential System. *Proc. ACM Conference on Computer and Communication Security, CCS'02*, 2002.
9. J. Camenisch and A. Lysyanskaya, An Efficient Non-Transferable Anonymous Multi-Show Credential System with Optional Anonymity Revocation. *Proc. Advances in Cryptology, Eurocrypt'01*, LNCS 2656, pp. 93-118, 2001.
10. W. Diffie, M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644-654, 1976.
11. C. M. Cllison. The Nature of A Usable PKI. *Computer Networks*, Vol. 31, No. 8, Elsevier, pp. 823-830, 1999.
12. C. M. Ellison, B. Frantz, B. Lampson, R.Rivest, B. Thomas, T. Ylnen. SPKI Certificate Theory, RFC 2693.
13. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. A Virtual Machine-Based PLatform for Trusted Computing. *Proc. 9th ACM Symposium on Operating Systems Principles*, pp. 193-206, 2003.
14. S. Goldwasser, S. Micali, C. Rackoff. The Knowledge Complexity of Interactive Proof-systems. 17^{th} *ACM Symposium on Theory of Computing*, pp. 291-304, 1985.
15. J. Holt, R. Bradshaw, K. Seamons, H. Orman. Hidden Credentials. *Proc. ACM Workshop in the Electronic Society*, pp. 1-8, 2003.
16. LaGrande technology architecrure. Intel Developer Forum, 2003.
17. ISO/IEC 9594-8, Information Technology - Open Systems Interconnections - The Directory: Authentication Framework.
18. ITU-T X.509 Recommandation, 2000, available at http://www.itu.int/rec/T-REC-X.509-200003-I/
19. N. Li, W. Du, D. Boneh. Oblivious Signature-based Envelope. *Proc. ACM Symposium on Principles of Distributed Computing*, pp. 182-189, 2003.
20. J. Li, N. L. Oacerts: Oblivious Attribute Certificates. *Proc. Applied Cryptography and Network Security*, LNCS 3531, pp. 301-307, 2005.
21. C. Mitchell. Trusted Computing, pp. 115-141, The Instituition of Electronic Engineers, London, 2005.
22. R. Merkle. A Certified Digital Signature. *Proc. Advances in Cryptology, Crypto'89*, LNCS 0435, pp. 218-238, 1989.
23. R. Rivest, B. Lampson. SDSI - A Simple Distributed Security Infrastructure. http://theory.lcs.mit.edu/ rivest/sdsi10.html, 1996.
24. K. E. Seamons, M. Winslett, T. Yu. Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation. *Proc. Network and Distributed System Security Symposium*, 2001.
25. S. Smith. *Trusted Computing Platforms - Design and Applications*. Springer 2005.
26. R. Sandhu, X. Zhang. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. *Proc. ACM symposium on Access control models and technologies*, pp. 147-158, 2005.
27. R. Sailer, X. Zhang, T. Jaeger, L, van Doorn. *Design and Implementation of a TCG-based Integrity Measurement Architecture*, USENIX Security Symposium, pp. 223-238, USENIX, 2004.
28. Trusted Computing Group. www.trustedcomputinggroup.org

29. TCG. TPM Main, Part 3 Commands, TCG Specification Ver. 1.2, Revision 62, www.trustedcomputinggroup.org, 2003.
30. World Wide Consortium. http://www.w3.org
31. T. Yu, M. Winslett. A Unified Scheme for Resource Protection in Automated Trust Negotiation. *IEEE Symposium on Security and Privacy*, pp. 110-122, 2003.

Appendix: TCG/TPM

The Trusted Computing Group (TCG) [28] defines a set of specifications aiming to provide hardware-based root of trust and a set of mechanisms to propagate trust to applications as well as across platforms. The root of trust in TCG is TPM (Trusted Platform Module), a tamper resistant secure coprocessor. TPM provides cryptographic functions, such as random number generation and RSA algorithms. Security mechanisms offered by TPM include integrity measurement and reporting, and sealed storage for secret data such as cryptographic keys. We next give a brief introduction on them. A TPM contains a set of Platform Configuration Registers (PCRs). PCR values record the integrity and state of a running platform from booting to loading of operation system to of loading applications [26]. With the integrity measurement and storage, TPM (attestator) can attest to a remote challenging platform the integrity of the platform under its protection through *platform attestation*. In particular, the challenging platform sends a challenge message to the attestator platform, who in turn returns the related PCR values signed by its Attestation Identity Key (AIK); the challenging platform verifies this attestation by comparing the signed values with expected values. The TPM command that instructs TPM to report the signed PCR values is TPM_Quote, whose input parameters specify the indices of the PCRs to be reported. Attestation can also be anonymous through Direct Anonymous Attestation [2].

TPM provides *sealed storage* that protect sensitive data with integrity values. Besides applying an encryption key (public key encryption) to encrypt the data, one or more PCR values are stored together with protected data during encryption. Consequently, TPM releases a protected data only if the current PCR values match those stored during encryption. The encryption key is protected either by a storage root key (SRK) that resides within TPM or by a key protected by the SRK. This actually forms a key hierarchy with the root being the SRK. The TPM commands that relate to sealed storage include TPM_Seal and TPM_Unseal. The TPM_Seal operation allows the invoking entity to explicitly state the future "trusted" configuration that the platform must be in for the secret to be revealed. The TPM_Seal operation also implicitly includes the relevant platform configuration (PCR values) when the TPM_Seal operation was performed. The TPM_Unseal operation will reveal TPM_Seal'ed data only if it was encrypted on this platform and the current configuration is the one named as qualified to decrypt it.

Two-Party Privacy-Preserving Agglomerative Document Clustering

Chunhua Su[1], Jianying Zhou[2], Feng Bao[2], Tsuyoshi Takagi[3], and Kouichi Sakurai[1]

[1] Department of Computer Science and Communication Engineering, Kyushu University, Japan
{su,sakurai}@itslab.csce.kyushu-u.ac.jp
[2] Systems and Security Department (SSD), Institute for Infocomm Research, Singapore
{jyzhou,baofeng}@i2r.a-star.edu.sg
[3] School of Systems Information Science, Future University-Hakodate, Japan
takagi@fun.ac.jp

Abstract. Document clustering is a powerful data mining technique to analyze the large amount of documents and structure large sets of text or hypertext documents. Many organizations or companies want to share their documents in a similar theme to get the joint benefits. However, it also brings the problem of sensitive information leakage without consideration of privacy. In this paper, we propose a cryptography-based framework to do the privacy-preserving document clustering among the users under the distributed environment: two parties, each having his private documents, want to collaboratively execute agglomerative document clustering without disclosing their private contents.

Keywords: documents clustering, privacy-preserving, cryptographic protocol.

1 Introduction

In today's information age, data collection is ubiquitous. Effective knowledge management is a major competitive advantage in today's information society. Data mining is becoming increasingly common in many areas such as banking, insurance, medicine, scientific research, and even in homeland security area to detect the terrorism.

Document clustering. Document clustering is a kind of textual data mining techniques. Different from regular data mining, in textual mining the patterns are extracted from natural language text rather than from structured databases of facts. Document clustering is the act of collecting similar documents into bins, where similarity is computed by some functions on the documents. This technique can enable the cross-enterprise document sharing over a similar topic. There are many different ways to show how a set of documents are related to one another. One solution is to show the terms in a query are related to the words in

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 193–208, 2007.

the document. It is very helpful when a party who holds some documents which belong to a discovered cluster with the description "security, privacy, database" wants to do a co-research with other parties who hold the similar documents. In this work, we concentrate more on agglomerative document clustering, since this method provides us more illustrations about the partitions of the clusters. The main idea is to find which documents have many words in common, and place the documents with the most words in common into the same groups and then build the hierarchy bottom-up by iteratively computing the similarity between all pairs of clusters and then merging the most similar pair.

Privacy issues in document clustering. The content of documents may include some sensitive information. The content could be misconstrued or cause great embarrassment if seen by unintended viewers. Even worse, the information can reveal a map to your corporate network and be used for outside-in malicious attacks. So the direct use of the document clustering by a complete scan over the content will cause security risks. Our purpose is to automatically group related documents based on their content with respect to the document privacy. We consider the scenario where there are two parties, says Alice and Bob, in a distributed network, each having a private document database denoted by D_A and D_B respectively. They want to collaboratively build an agglomerative document clustering on the concatenation of their databases, and get the common benefit for doing clustering analysis in the joint databases $D = D_A \cup D_B$ associated with understandable descriptions of the clusters. Moreover, the agglomerative clustering can enforce this function by building a hierarchical tree to show the relationship between two similar clusters. As each party concerns about his data privacy, neither party is willing to disclose his raw document data set to others. That is, the only information learned by Alice about D_B is that which can be learned from the output of the data mining algorithm, and vice versa. We do not assume any "trusted" third party who computes the joint output.

1.1 Related Works

Our paper is focus on both document clustering and its privacy. We will give out a brief review on the relative works of the both aspects. In the past few years, a lot of different document clustering algorithms have been proposed in the literature, including Scatter/Gather [3], SuffixTree Clustering [18]. Bisecting k-means is proposed by M. Steinbach et al. [14] based on an analysis of the specifics of the clustering algorithms and the nature of document data. The above methods of text clustering algorithms do not really address the special challenges of text clustering. They do not provide an understandable description of the clusters. This has motivated the development of new special text clustering methods which are not based on the vector space model and some frequent-term based methods have been proposed such as Beil et al. [2].

The first paper to take the classic cryptographic approach to privacy-preserving data mining was presented by Lindell and Pinkas [11]. They presented an efficient protocol for the problem of distributed decision tree learning. Some privacy-preserving $k-$means clustering schemes are also proposed. The work by Vaidya

and Clifton [16] introduces a solution based on secure multi-party computation techniques. Specifically, the authors proposed a method for $k-$means clustering when different sites contain different attributes for a common set of entities. In the solution, each site learns the cluster of each entity, but learns nothing about the attributes at other sites. Jagannathan and Wright proposed the privacy-preserving k-means clustering scheme and introduced the concept of arbitrarily partitioned data which is generalization of both horizontally and vertically partitioned data and provides a privacy-preserving protocol for k-means clustering in the setting of arbitrarily partitioned data [9].

1.2 Our Contributions

Based on the related works, we made an extension to preserve the privacy document clustering over two parties. There are important differences compared with the previous works. The traditional document clustering techniques do not consider privacy. Also, the existing privacy-preserving data mining researches do not deal with the unstructured textual data. Our work is going to solve the two problems.

- **Distributed Documents Clustering.** We assume that there are two parties who hold their private documents dataset respectively want their documents clustered. And we propose distributed document clustering with divide phase and merge phase the to preserve the privacy document clustering over two-party's databases. Finally, the protocols output the cluster description and the agglomerative clusters, and the parties can check to which clusters their documents belong to.
- **Privacy-Preserving Framework.** We propose a framework to preserve the each privacy in two-party document clustering. In our proposal, every party does the documents clustering computation via the interactive protocol and outputs the clusters for each document without revealing the document's content. During the execution of the protocol, all the privacy of the parties is preserved.
- **Dealing with Unstructured Data.** The existing privacy-preserving data mining techniques [16] [9] focus on databases which are clearly defined and structured, it is easy to run queries and formulas which extract meaningful information. We deal with document data which are unstructured. We use privacy keyword searching techniques to extract frequent terms from large unstructured document data sets, discover relationships and summarize the information.

The rest of the paper is organized as follows: We outline the frequent-term based agglomerative document clustering preliminary in next section and give out privacy definition and brief introduction of cryptographic primitives used in this paper in Section 3. Then we propose a privacy-preserving protocol in Section 4. The implementation of the privacy-preserving protocol is described in Section 5. The performance and security is discussed in Section 6. We give our conclusions in Section 7.

2 Frequent-Term Based Agglomerative Document Clustering

In this paper, we apply frequent-term based method to execute the agglomerative document clustering. The intuition of this criterion is that many frequent items should be shared within a cluster while different clusters should have more or less different frequent terms. A term is any preprocessed word within a document, and a document can be considered as a set of terms occurring in that document at least once. The key idea is to greedily select the next frequent term, which represents the next cluster, minimizing the overlap of clusters in terms of shared documents. It clusters the high-dimensional vector space, but to consider only the low-dimensional frequent term sets as cluster description candidates. The set of all frequent item sets can be efficiently determined even for large databases. It is promising because it provides a natural way of reducing the large dimensionality of the document vector space.

Problem Definition: Our protocol is a two-party frequent term-based clustering protocol with divide-and-merge process. At first, we get the minimum-overlap clusters and their descriptions and then build a agglomerative tree model for these clusters. Unlike the partitional algorithms that build the hierarchical solution for top to bottom, agglomerative algorithms build the solution by initially assigning each document to its own cluster and then repeatedly selecting and merging pairs of clusters, to obtain a single all-inclusive cluster. We have defined a flat clustering as a subset of the set of all subsets of the database D, described by a subset of the set of all frequent term sets, that covers the whole database. To discover a clustering with a minimum overlap of the clusters, we follow a greedy approach. After that, we employ agglomerative algorithms to build the tree from bottom (i.e., its leaves) toward the top (i.e., root). The root is the whole database $D_A \cup D_B$ while the leavers is the minimum-overlapping cluster.

Preliminaries: At first, we use frequent term-based clustering determines a flat clustering without overlap. An unstructured set of clusters covering the whole databases of the two parties. Let $D_A = \left\{Doc_A^1, ..., Doc_A^m\right\}$ be a database of m text documents held by the Alice (D_B is held by Bob, respectively). Each document Doc_A^j is represented by the set of terms occurring in D_A. Let $minsupp$ be a real number, $0 \leq minsupp \leq 1$ which is agreed by both Alice and Bob. Let $w_i = \left\{w_i^1, ..., w_i^k\right\}$ be the set of all frequent term sets in D_i (i = A or B) with respect to $minsupp$, the set of all term sets contained in at least $minsupp$ of the D_i documents, let $cov(w_i)$ denote the *cover* of w_i, the set of all documents containing all terms of w_i, more precisely, $cov(w_i) = \left\{Doc_i^j \in D_i \mid w_i \subseteq Doc_i^j\right\}$ The cover $cov(w_i)$ of each element w_i is a cluster candidate. A cluster can be any subset of the set of all subsets of two parties' database $D = D_A \cup D_B$ such that each document of D is contained in at least one of the sets (clusters). We define a clustering description CD as a subset of $w_1, ..., w_n$. We determine a cluster with a minimum overlap of the cluster candidates. The overlap can be measured by the mutual entropy.

3 Privacy Definition and Cryptographic Primitives

3.1 Privacy Definition of Secure Multi-party Computation

Our protocol construction is based on secure multi-party computation which was introduced by Yao [17] and extended by Goldreich, Micali, and Wigderson [8]. It allows a set of n players to securely compute any agreed function on their private inputs and the corrupted players do not learn any information about the other players' inputs. In secure multi-party computation, we always assume that all parties are *semi-honest*. A semi-honest party follows the rules of the protocol giving its correct input, but it is very curious and it only tries to deduce information on the inputs of the honest parties by inspecting all the information available to the corrupted parties. This is somewhat realistic in the real world because parties who want to mine data for their mutual benefit will follow the protocol to get correct results. In the secure computation setting, there are two models. In ideal model, every party sends inputs to a trusted party, who computes the document clustering and sends the outputs. In real model, every party runs a real private document clustering protocol with no trusted help. We say that a real protocol that is run by the parties (in a world where no trusted party exists) is secure, if no adversary can do more harm in a real execution than in an execution that takes place in the ideal world.

Let $f : 0,1^* \times 0,1^* \longrightarrow 0,1^* \times 0,1^*$ be a function. A two-party protocol is defined by a pair of probabilistic polynomial-time interactive algorithms $\pi = (\pi_A, \pi_B)$. The protocol π is executed as follows. Initially, Alice, who operates according to π_A, receives an input a and a random input r_A, and Bob, who operates according to π_B, receives an input b and a random input r_B. The execution then proceeds by synchronous rounds, where, at each round, each party may send to the other party a message as specified by π, based on her input, her random input, and messages received in previous rounds. At each round, each party may decide to terminate and output some value based on her entire view consisting of her input, random input, and received messages. For defining the privacy of π with respect to a functionality f, it is convenient to use the following notation. Consider the probability space induced by the execution of π on input $x = (a, b)$ (induced by the independent choices of the random inputs r_A, r_B).

(Privacy in The Semi-honest Model): Consider the probability space induced by the execution of π on input $x = (a, b)$ (induced by the independent choices of the random inputs r_A, r_B). Let $view_A^\pi(x)$ (resp., $view_B^\pi(x)$) denote the entire view of Alice (resp., Bob) in this execution, including her input, random input, and all messages she has received. Let $output_A^\pi(x)$ (resp., $output_B^\pi(x)$) denote Alice's (resp., Bob's) output. Note that the above four random variables are defined over the same probability space. We say that π privately computes a function f if there exist probabilistic, polynomial-time algorithms S_A and S_B such that:

$$\{(S_A(a, f_A(x)), f_B(x))\}_{x=(a,b)\in X} \equiv \{(VIEW_A^\pi(x), OUPUT_B^\pi(x))\}_{x\in X},$$

$$\{(f_A(x), S_B(b, f_B(x)))\}_{x=(a,b)\in X} \equiv \{(OUPUT_A^\pi(x), VIEW_B^\pi(x))\}_{x\in X}.$$

where $\equiv$ denotes computational indistinguishability, which means that there is no probabilistic polynomial algorithm A can distinguish the probability distribution over two random string. This paper only considers the semi-honest adversaries. In this model, every party is assumed to act according to their prescribed actions in the protocol.

Any two-party functionality can be securely computable in the malicious model by introducing some cryptographic premitivies which force each party to either behave in a semi-honest manner or be detected (which is called forcing semi-honest behavior), it was proved in [7]. Thus, although totally-honest behavior may be difficult to enforce, semi-honest model may be assumed in many settings.

3.2 Cryptographic Primitives

For constructing a secure documents clustering protocol, we apply the following cryptographic primitives.

Homomorphic Encryption: In our protocol, we require a homomorphic encryption scheme satisfying: $E(a) * E(b) = E(a+b)$, where E is a cryptosystem, $*$ and $+$ denote modular multiplication and addition, respectively. It also follows that $E(a)^c = E(a * c)$ for $c \in N$. The Paillier cryptosystem [13] is a proper scheme which has this property and is the cryptosystem of our choice to construct a secure protocol.

Oblivious Transfer (OT): The 1-out-of-N oblivious transfer protocol is used to do the circuit evaluation privately. In the case of semi-honest adversaries, there exist simple and efficient protocols for oblivious transfer. An 1-out-of-N Oblivious Transfer protocol refers to a protocol where at the beginning of the protocol one party, party B has N inputs $x_1, ..., x_N$ and at the end of the protocol the other party A learns one of the inputs x_i for some $1 \leq i \leq N$ of her choice, without learning anything about the other inputs and without allowing B to learn anything about x_i.

Oblivious Polynomial Evaluation (OPE): Oblivious Polynomial Evaluation(OPE) is one of fundamental cryptographic techniques. It involves a sender and a receiver. The sender's input is a polynomial $Q(x)$ of degree k over some field F and the receiver's input is an element $z \in F$. The receiver learns $Q(z)$. It is quite useful to construct some protocols which enable keyword queries while providing privacy for both parties: namely, (1) hiding the queries from the database (client privacy) and (2) preventing the clients from learning anything but the results of the queries (server privacy).

4 Privacy-Preserving Document Clustering Protocol

There are two phases in our protocol, divide phase and merge phase. It works by breaking down the distributed documents clustering problem into two subproblems and the solutions to the sub-problems are then combined to give a solution to the original problem. In our protocol, divide-and-merge algorithms are implemented in a non-recursive way and the computation is interactive,

so both two parties plays roles of both client and server, called client party and server party, respectively. At first, every local party makes a keyword list, and then he make an private intersection computation with the other parties' keyword list. After that two parties will make local clustering according his keyword list to get the local minimum-overlapping clusters. Finally both Alice and Bob can merge the two clusters into one according to similarity of every two clusters. The output will be a tree construction with a set of all documents as the root, and cluster description will be the intersection of two keyword sets.

Privacy-Preserving Document Clustering Protocol

Input: Alice's document database D_A and Bob's document database D_B
Output: The clusters and their descriptions based on $D_A \cup D_B$

1. The two parties execute the document clustering in the *Divide Phase* and get the minimum-overlapping clusters respectively.
2. The two parties execute the interactive agglomerative clustering computation in the *Merging Phase*.
3. Both parties get agglomerative clusters and their descriptions.

In our protocol, all the computation of agglomerative document clustering is based on each document's the frequent-term without reveal any unnecessary content of document. What a party learns during the execution of the protocol is the common frequent-term with the other party and the final output.

4.1 Local Pre-computation for Textual Data

All methods of text clustering require several steps as preprocessing of the data. First, all non-textual information, punctuation as well as stopwords such as "I", "am", "and" are removed from the documents . Note that a term may either be a single word or consist of several words. After that, every party should do the pre-computation on their own text data in every individual document. Every party form his database which contains N frequent terms as $X = (x_i, num_i)$ with $1 \leq i \leq N$. In every document which is held by the participant party, x_i is a keyword and num_i is the document number that the keyword x_i occurs in one document whose frequency is lager than *minsupp*. two parties number all his own documents for 1 to m, where m is the number of total documents which are held by a party. For example, the Alice can arrange number in order to his documents as $Doc_A^1, ..., Doc_A^m$.

4.2 Divide Phase of Document Clustering

At first, Alice and Bob have to pre-determine the common threshold minimum support *minsupp* and the function for calculating the mutual overlap of frequent

sets. After that, two parties execute the privacy-preserving frequent term query scheme interactively and gets the frequent term sets for document clustering.

In this phase, we apply the algorithm proposed by Beil et al. [2] which works in a bottom-up fashion. Starting with an empty set, it continues selecting one more element (one cluster description) from the set of remaining frequent term sets until the entire database is contained in the cover of the set of all chosen frequent term sets (the clustering). Setting the database formed by selected documents. In each step, party selects the remaining frequent term set with a cover having the minimum overlap with the other cluster candidates local. Note that the documents covered by the selected frequent term set are removed from the database D (Let D denote the union set of a party's document database and the other party's documents which contain the common frequent keywords.) and in the next iteration, the overlap for all remaining cluster candidates is recalculated with respect to the reduced database.

Clustering Algorithm of Divide Phase

Input: Party P(can be Alice or Bob)'s frequent keyword w_i of databases D_i, where $1 \leq i \leq n$ and the threshold minimum support $minsup$
Output: The clusters and their descriptions based on frequent terms in w_i.

1. Party P locally find out all frequent keyword whose frequency is larger than $minsupp$, denoted by w.
2. Do the private keyword queries with w and gets the IDs of the other party's documents which have the common keywords.
3. Calculate entropy overlap for w and let $Candidate_w$:= element of w with minimum entropy overlap;
4. Removes all documents in $cov(Candidate_w)$ from D and from the coverage of all of the remaining documents.
5. Let $Selected_w := Selected_w \cup \{Candidate_w\}$ and $Remain_w := w - \{Candidate_w\}$;
6. Remove all documents in $cov(Candidate_w)$ from D and from the coverage of all of the $Remain_w$;
7. Party P updates the clusters until all the clusters are minimum-overlap;
8. Return the keyword sets of $Selected_w$ as cluster descriptions and the cover of the sets $Selected_w$ as clusters.

Every party executes the clustering algorithm of divide phase, and get his local clusters. The algorithm returns clustering description and clusters which is non-overlapping. After this local computation, every party can continue the agglomerative clustering: merging the cluster to build a agglomerative clustering tree. We will show how to do the queries privately in next section.

4.3 Merge Phase of Document Clustering

In the divide phase, every party gets the local non-overlap clusters based on the frequent terms of their own documents and other parties' document with common frequent keywords. The agglomeration algorithm creates hierarchical clusters. At each level in the hierarchy, clusters are formed from the union of two clusters at the next level down. In the merge step, every party starts with his own cluster and gradually merge clusters until all clusters have been gathered together in one big cluster.

There are two steps for clusters merging computation: **1. Cluster Inclusion Merging Step:** A smaller cluster which is included by the larger one will be merged into the larger one. **2. Agglomerative Step:** The two similar clusters will be merged as a new cluster according to the similarity computation. At the same time, the description of new cluster is will be an intersection of the two clusters' descriptions.

Algorithm of Merge Phase

1. Initially, every party uses his clusters to do a private inclusion test with other party's clusters. Merge the two cluster if one cluster is included in the other cluster. Stop until every included subset of clusters is merged.
2. Among all remaining clusters, pick the two clusters to do the private similarity computation.
3. Replace these two clusters with a new cluster, formed by merging the two original ones with the most similarity.
4. Repeat the above step 2 and step 3 until there is only one remaining cluster which covers all parties' databases.

Note that in the algorithm, we can preserve the privacy by only outputting the cluster description. The merged cluster description CD is an intersection of two original cluster descriptions, not the union of the two. Because the coverage $Cov(CD)$ can cover all the documents whose frequent terms are included in CD, with the output of the protocol, the clients match their documents with the cluster descriptions CD and assign them to the proper cluster with a subset relationship between each cluster and its predecessors in the hierarchy privately. This produces a binary tree or dendrogram, of which final agglomerative cluster is the root and each cluster is a leaf, the height of the bars indicates how close the clusters and their descriptions are.

5 Implementing the Privacy-Preserving Protocol

Here in this section, we show how to implement the privacy-preserving protocol using the cryptographic techniques which we have mentioned in Section 3.

Our constructions use a semantically-secure public-key encryption scheme that preserves the group homomorphism of addition and allows multiplication by a constant. It is a triple $C = (G, E, D)$, where G is the key generation algorithm that returns (sk, pk) consisting of a secret key sk and a public key pk, E is the encryption algorithm and D is the decryption algorithm.

5.1 Private Document Selection

When a party gets the local frequent keywords, he has to construct some queries to select the documents which contain the same frequent term with respect to the privacy. In this section, we construct a protocol using oblivious polynomial evaluation (OPE) scheme from [6] and apply the zero knowledge proof to avoid the malicious inputs of the client party. The basic idea of the construction is to encode the database $D's$ entries in $\{X = (x_1, num_1), ..., (x_n, num_n)\}$ as values of a polynomial, i.e., to define a polynomial Q such that $Q(x_i) = (num_i)$, where x_i denotes the keyword and num_i denotes the document number for clustering. Note that this design is different from previous applications of OPE, where a polynomial (of degree k) was used only as a source for $(k+1)$-wise independent values.

Document Selection with Private Keyword Search

Input: Client party inputs his local frequent keyword w; Server party input $\{x_i, num_i\}_{i\in[n]}$, all x_i's are distinct
Output: Client party gets document number $number_i$ if $w = x_i$, nothing otherwise; Server party: nothing

1. The server party defines L bins and maps the n items into the L bins using a random, publicly-known hash function H with a range of size L. H is applied to the database's frequent keywords, frequent keyword x_i is mapped to bin $H(x_i)$. Let m be a bound such that, with high probability, at most m items are mapped to any single bin.
2. For every bin j, the server party defines two polynomials P_j and Q_j of degree $(m-1)$. The polynomials are defined such that for every pair (x_i, num_i) mapped to bin j, it holds that $P_j(x_i) = 0$ and $Q_j(x_i) = (num_i|0^l)$, where l is a statistical security parameter.
3. For each bin j, the server party picks a new random value r_j and defines the polynomial $Z_j(w) = r_j \cdot P_j(w) + Q_j(w)$.
4. The two parties run an OPE protocol in which the server evaluates all L polynomials at the searchword w.
5. The client party learns the result of $Z_{H(w)}(w)$, i.e., of the polynomial associated with the bin $H(w)$. If this value is of the form $number_i|0^l$ the client party gets the $number_i$.

Our construction uses an OPE method based on homomorphic encryption such as Paillier's system [13] in the following way.

- The server party's input is a polynomial of degree m, where the polynomial $P(x) = \sum_{i=0}^{m} a_i x^i$, The client party's input is a keyword value w.
- The client party sends to the server party homomorphic encryptions of the powers of w up to the m-th power, i.e., $Enc(w), Enc(w^2), ..., Enc(w^m)$.
- The server party uses the homomorphic properties to compute the following: $\prod_{i=0}^{m} Enc(a_i w^i) = \sum_{i=0}^{m} Enc(a_i w^i) = Enc(P(w))$. The client party sends this result back to the server party.

For preventing client from cheating in OPE, the server party can ask client party do zero knowledge proof of $Enc(w_i)$ before the construction in terms of a single database bin. We can use the Damgård and Jurik's scheme proposed in [4] to prove that the input is the encryption of w_i without disclosing the keyword w_i.

The document selecting protocol preserves client party's privacy because the server cannot distinguish between any two of client party' inputs w, w', the protocol also protects the server party's privacy if a polynomial Z with fresh randomness is prepared for every query on every bin, then the result of the client party's query w is random if w is not a root of P, and the malicious input of clients party can be prevented by using the zero knowledge proof of the frequent keyword w.

Lemma 1 (Client party's privacy is preserved). If the encryption scheme is semantically secure, then the views of client for any two inputs are computationally indistinguishable. (The proof uses the fact mentioned above that the only information that server party receives consists of semantically secure encryptions.)

Lemma 2 (Server party's privacy is preserved). For C' that operates in the real model, there is a client C operating in the ideal model, such that for every input X of Bob, the views of Bob in the ideal model is indistinguishable from the views in the real model. (The proof is that a polynomial Z with fresh randomness is prepared for every query on every bin, then the result of the client's query w is random if w is not a root of P.)

5.2 Private Cluster Inclusion Test

After the local computation of document clustering, there may be overlaps among each party's local result. So we have to combine such overlap and make a cluster to be unique in the global result. Every cluster can be represented as a binary string according to documents' order from party A to party B, such as $Doc_A^1, Doc_A^2, ..., Doc_B^m$. Each bit of the string corresponds to a document, there is 1 in the entry i if and only if the cluster contains the party 1's document Doc_1^i, if the document doesn't exist, there is 0. Client party i has a set $C_i \subseteq D$, Server party j has a set $C_j \subseteq D$, and the two parties must establish whether $C_i \subseteq C_j$ or with neither of the parties obtaining any additional information. More precisely,

the protocols must satisfy client-privacy and server-privacy. We assume that the client has n words in his database. Our basic idea is based on the fact that if for two clusters C_i and $C_j$satisfying $C_i \in C_j$, then we have $|C_i \cap C_j| = |C_j|$.

Here, we modify the matrix-based private inclusion scheme [10] into a new scheme which can deal with binary string to construct our private cluster merging protocol. We implement this with the homomorphic cryptosystem which is proved to be secure in the sense of IND-CPA under reasonable complexity assumptions.

Private Cluster Inclusion Test Protocol

PRIVATE INPUT: Client party: cluster C_i, Server party: cluster C_j.
PRIVATE OUTPUT: Client party knows whether $C_i \subseteq C_j$, if yes, outputs $CD_i \cap CD_j$.

1. Client party generates a new key pair $(sk, pk) \longleftarrow G$. Send pk to server party. For any $i \in [n]$, generate a new nonce $r_i \xleftarrow{r} R$. Send $e_i \longleftarrow E_{pk}(C_i; r_i)$ to server party.
2. Server party draws $s \xleftarrow{r} P$, $r \xleftarrow{r} R$ uniformly at random. Set $e \longleftarrow (\prod_{t=1}^{l} C_i[t]/C_j[t])^s \cdot E_{pk}(0; r)$, where l is the last l_{th} bit of 1. Send e to client party.
3. Client party sets $d \longleftarrow D_{sk}(e)$. Accept that $C_i \subseteq C_j$ iff $d = 0$ and send the result to server party.
4. server party returns cluster C_j as a merged cluster and outputs the $CD_j = CD_i \cap CD_j$.

After this process, the flat clusters for the agglomerative document clustering are generated and only the cluster descriptions are output. And all the parties can use those clusters descriptions to group their documents. By using zero-knowledge proofs, client party can prove the correctness of (a) pk is a valid public key and that (b) every bit of C_i encrypts either 0 or 1.

Lemma 3: Private cluster inclusion testing protocol is a privacy-preserving protocol. Computational client-privacy follows directly from the IND-CPA security. So an adversary can learn nothing about the plaintext corresponding to a given ciphertext, even when the adversary is allowed to obtain the plaintext corresponding to ciphertexts of its choice. As server party sees only ciphertexts of encrypted clusters, his privacy is guaranteed as the second step depends only on whether $C_i \in C_j$ or not.

5.3 Private Measurement of Similarity

To measure the similarity of the cluster, we consider that two clusters which have the most overlap of documents have the most similarity. Such two clusters

which contain most documents in common should be merged into a cluster in the agglomerative clustering process. We use Hamming distance to measure that similarity of two clusters. The Hamming distance is the number of positions in two strings of equal length for which the corresponding elements are different. Every cluster can be represented by a binary string as the same as in the private inclusion cluster merging protocol. To compute the Hamming distance privately, we use the Private-Sample-XOR Protocol proposed by J. Feigenbaum [5] as following:

Notions: In this protocol, We let $d_h(a, b)$ denote the Hamming distance between (a, b), for any $x \in \{0,1\}^n$ $r \in [n]$ and $m \in \{0,1\}^n$, we denote by $x << r$ a cyclic shift of x by r bits to the left, and by $x \bigoplus m$ the string whose i-th bit is $x_i \bigoplus m_i$

Private Approximation of Hamming Distance

1. Party A generates a random mask $m_A \overset{R}{\leftarrow} \{0,1\}^n$ and a random shift amount $r_A \overset{R}{\leftarrow} [n]$. And he computes the n-bit string $a' \overset{def}{=} (a << r_A) \bigoplus m_A$
   Symmetrically, Party B generates $m_B \overset{R}{\leftarrow} \{0,1\}^n$ and $r_B \overset{R}{\leftarrow} [n]$, and computes $b' \overset{def}{=} (b << r_B) \bigoplus m_B$
2. A and B invoke in parallel two $\binom{n}{1}$-OT protocols:
   - A retrieves $z_A \overset{def}{=} b'_{r_A}$ from B;
   - B retrieves $z_B \overset{def}{=} a'_{r_B}$ from A;
3. A sends $z'_A \overset{def}{=} z_A \bigoplus m_A$ to B. B sends $z'_B \overset{def}{=} z_B \bigoplus m_B$ to A. Both parties locally output $z'_A \bigoplus z'_B$.

After executing the protocol we can get the approximate result of similarity of the two clusters. The smaller the hamming distance, the more similar of two clusters, and the most similar two clusters' cluster descriptions will be joined into an intersection, i.e, $CD_A \cap CD_B$.

Lemma 4: Both parties' privacy is preserved. Proof : The privacy can be formally argued by describing a simulator for each party. Alice's random inputs m_A, r_A in the real protocol are independent of the inputs (a, b) and the output z, and are thus distributed in the simulated view as they should. And the output z_A received from $\binom{n}{1}$-OT protocol in the real model is independent of a, b, m_A, r_A, z, as in the simulated view. As in ideal model, a simulator for Alice's view based on the input a' and output $z'_A \bigoplus z'_B$ is computationally indistinguishable with the view in real model. Simulator for Bob's view may be obtained similarly.

5.4 Performance Evaluation

During the private keyword search, we assume that client party assign the n items to L bins arbitrarily and evenly, ensuring that L items are assigned to every bin; thus, $L = \sqrt{n}$. The server party's message during the OPE consists of $L = O(\sqrt{n})$ homomorphic encryptions, he evaluates L polynomials by performing n homomorphic multiplications, and replies with the $L = \sqrt{n}$ results. This protocol has a communication overhead of $O(\sqrt{n})$, $O(n)$ computation overhead at the client party's side, and $O(\sqrt{n})$ computation overhead at the server party's side. In private cluster inclusion test protocol, server party does not perform any pre-computation, when server party gets client party's query as an encrypted binary string, the communication of this protocol is $len(|d|)$ bits. For computation of similarity of clusters, we use a $\binom{n}{1}$-OT protocol (in the semi-honest model) as a sub-protocol. Then, the protocol for approximating $d_h(a, b)$, and whose round complexity is $OT + 1$. Hamming distance function can be privately ϵ-approximated with communication complexity $O(n^{1/2}/\epsilon)$ and three rounds of interaction.

6 Security Analysis of the Whole Protocol

Except for the three interactive sub protocols above, other computation process in our protocol are done locally by the two parties, so under the semi-honest model, one party only gets the information based on his own frequent keywords, and any probabilistic polynomial time adversary can not distinguish the responding output in real model from the one in ideal model with any party's private input. By using the zero knowledge proof, our protocol also can be secure against the malicious party, but the computational and communication complexity will increase.

Theorem 1 (Security of approximation protocol). *The document clustering protocol is the privacy-preserving against the semi-honest adversary.*

Proof: We also can see our protocol is privacy-preserving as a whole. Intuitively, the privacy of the protocol follows from the fact that in all processes of obtaining the output no party learns any additional information which is not published by the other party. According to the privacy definition in Section 3.1, we give out the privacy proof as following.

From the Lemma 1 and lemma 2, we know that in private documents selection, the security of the sub protocol is based on the assumptions used for proving the security of the homomorphic encryption system. Since the server receives semantically-secure homomorphic encryptions and the sub protocol protects the privacy of the client, the subprotocol ensures the client party's privacy because server cannot distinguish between any two of client party's inputs w, w'. For server party, if w is not a frequent keyword, the output is just a random number. It means that the adversary's views of both parties in both real model and ideal model is computationally indistinguishable. Every party only learn that w is a common frequent keyword.

During the private cluster inclusion test, computational client-privacy also follows directly security of the homomorphic encryption system which assures that e is a random encryption of zero if $C_i \subset C_j$, or a random encryption of a random plain text if $C_i \subsetneq C_j$. According to Lemma 3, the server party sees only ciphertexts, so any adversary that can distinguish two vectors of ciphertexts can be used for distinguishing only two ciphertexts. Every party only learn that whether $C_i \subset C_j$ or not.

When computing the private approximation of hamming distance between the inputs a and b, the view of each party in these invocations can be simulated from its input and $d_h(a, b)$. Summarizing, we have it has a simulator S such that $S(d_h(a, b))$ and the output $d_h^{'}(a, b)$ are identically distributed according to Lemma 4's security proof, so that no probabilistic polynomial time adversary can distinguish $S(d_h(a, b))$ and $d_h^{'}(a, b)$.

So the whole protocol is privacy-preserving against the probabilistic polynomial time adversary under semi-honest model.

7 Conclusions and Future Works

In this paper, we proposed a divide-and-merge method for two-party agglomerative document clustering and gave a framework to preserve the privacy of the two parties. We used the cryptographic techniques to construct a secure two-party computation model to get final agglomerative clusters without revealing the content of the documents. All the similar documents will be grouped into corresponding clusters without revealing the document content. With the agglomerative tree and the clustering descriptions as the output, the two parties can enable the privacy-preserving document sharing.

As our future work, we want to extend the scheme to a multi-party case which involves n mistrusful parties. and design the secure multi-party protocol is how to deal with adaptive adversaries who may choose the corrupted parties during the document clustering computation. Developing the efficient privacy-preserving data mining algorithms is also a challenging task. Developing some new techniques with low communication and computation complexity to do other privacy-preserving data mining tasks is also one of our future goals.

Acknowledgement

The first author is partly supported by NEC Foundation for Computers and Communications (C&C) Promotion.

References

1. D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano. *Public Key Encryption with Keyword Search.* In EUROCRYPT, Interlaken, Switzerland, 2004.
2. F. Beil, M. Ester, X. Xu. *Frequent Term-Based Text Clustering*, Proceedings of the 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD), 2002.

3. D. R. Cutting, D. R. Karger, J. O. Pedersen, J. W. Tukey. *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*. pp.318-329, Proc. ACM SIGIR 92, 1992.
4. I. Damgård and M. Jurik. *Client/server Tradeoffs for Online Elections.* volume 2274 of Lecture Notes in Computer Science, pp.125-140, PKC2002, 2002.
5. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. *Secure multiparty computation of approximations.* pp.435-472, ACM Transactions on Algorithms 2, 2006.
6. M. J. Freedman, Y. Ishai, B. Pinkas and O. Reingold. *Keyword Search and Oblivious Pseudorandom Functions*. Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, 2005.
7. O. Goldreich. *Foundations of Cryptography Volume 2, Chapt.7*, Cambridge Univ. Press, 2004.
8. O. Goldreich, S. Micali and A. Wigderson. *How To Play Any Mental Game*. In Proceedings of the 19th annual ACM symposium on Theory of computing, 1987.
9. G. Jagannathan and R. Wright. *Privacy-Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data.* Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2005.
10. S. Laur, H. Lipmaa and T. Mielikainen. *Private Itemset Support Counting*. volume 3783 of Lecture Notes in Computer Science, pp. 97-111, Beijing, China, 2005.
11. Y. Lindell and B. Pinkas. *Privacy preserving data mining.* In Advances in Cryptology CRYPTO '00, volume 1880 of Lecture Notes in Computer Science, pp. 36-54. Springer-Verlag, 2000.
12. W. Ogata and K. Kurosawa. *Oblivious Keyword Search*. Journal of Complexity, 20(2.3)pp.356-371, 2004.
13. P. Paillier. *Public-key Cryptosystems Based on Composite Degree Residuosity Classes.* In EUROCRYPT, Prague, Czech Republic,1999.
14. M. Steinbach, G. Karypis and V. Kumar. *A Comparison of Document Clustering Techniques*. In KDD Workshop on Text Mining, 2000.
15. D. Song, D. Wagner and A. Perrig. *Practical Techniques for Searches on Encrypted Data*. In Proc. of the 2000 IEEE Security and Privacy Symposium, May 2000.
16. J. Vaidya and C. Clifton. *Privacy-Preserving K-Means Clustering Over Vertically Partitioned Data*. In Proc. of the 9th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, Washington, 2003.
17. A.C Yao, *Protocols for Secure Computation,* In 23rd FOCS, 1982.
18. O. Zamir, O. Etzioni, *Web Document Clustering: A Feasibility Demonstration*. pp.46-54, Proc. of 21st ACM SIGIR on Research and Development in Information Retrieval, Melbourne, Australia, 1998.

Efficient Bid Validity Check in ElGamal-Based Sealed-Bid E-Auction

Kun Peng and Ed Dawson

Information Security Institute, Queensland University of Technology

Abstract. Bid opening in sealed-bid e-auction is efficient when a homomorphic encryption algorithm is employed to seal the bids and homomorphic bid opening is employed to open the bids. Such e-auction schemes are called homomorphic auctions. However, high efficiency of homomorphic auctions is based on an assumption: the bids are valid (e.g. within a special range). An undetected invalid bid can compromise correctness and fairness of the auction. Unfortunately, in most existing homomorphic auction schemes, proof and verification of validity of the bids is either ignored or too inefficient. Recently, a technique called batched bid validity check [25] is proposed to improve efficiency of proof and verification of bid validity in a special kind of homomorphic auction schemes: secret-sharing-based homomorphic auctions. However, secret-sharing-based homomorphic auction schemes [13, 15, 26, 24] are not a main stream in homomorphic auction schemes as they employ threshold secret sharing techniques to seal the bids. Main stream homomorphic auction schemes employ a homomorphic encryption algorithm with threshold distributed decryption to seal the bids as it is simpler and more efficient than secret sharing. In this paper, an ElGamal-encryption-based homomorphic encryption scheme is proposed. It employs a batched proof and verification of bid validity to achieve high efficiency in bid validity check. Its batch proof and verification technique is more advanced than that in [25], so it is simpler and more efficient than the homomorphic auction scheme in [25].

1 Introduction

In a sealed-bid auction scheme, each bidder chooses his evaluation from a number of biddable prices and submits it to some auctioneers before a deadline. After the deadline, the auctioneers open the bids and determine the winning price and winner(s) according to a pre-defined auction rule. The commonly applied auction rules include first bid auction (the bidder with the highest bid wins and pays the highest bid), Vickrey auction (the bidder with the highest bid wins and pays the second highest bid) and the ρ^{th} bid auction (the bidders with the $\rho - 1$ highest bids win, pay the ρ^{th} highest bid and each get an identical item). The first-bid auction and Vickrey auction can be regarded as special cases of the ρ^{th} bid auction, which is a general solution. An auction must be correct, namely the auction result is strictly determined according to the auction rule. Fairness is

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 209–224, 2007.

necessary in any auction such that every bidder is equally treated and no bidder can take advantage over other bidders. Usually, bid privacy must be kept in a sealed-bid auction scheme, which requires that in the course of bid opening no bid is revealed. Robustness is also desired in auction schemes, which requires that even in abnormal situations, correct auction result can be obtained after running the auction protocol. In addition, a sealed-bid auction scheme should be flexible and support various auction rules.

When bid privacy must be kept in a non-interactive sealed-bid e-auction.To adopt this bid opening function, one-selection-per-price principle must be employed. Each bidder has to submit a bidding selection at every biddable price to indicate whether he is willing to pay that price ("YES" or "NO"). Every selection is sealed with a homomorphic sealing function, so that the auctioneers can exploit its homomorphism to test whether the number of bidders willing to pay a price is over ρ without revealing any bidding selection. When this homomorphic bid opening mechanism is applied together with binary search strategy, the winning bid can be determined very efficiently.

The sealed-bid e-auction schemes employing homomorphic bid opening [13, 15, 1, 19, 26, 24, 23] are called homomorphic auction schemes in this paper. There are two kinds of homomorphic auction schemes according to the bid sealing function. The first one [13, 15, 26, 24] employs homomorphic threshold secret sharing to seal the bids. Every bidder shares each of his bidding selections using a homomorphic threshold secret sharing algorithm (e.g. [22]) among the auctioneers, who can apply homomorphic threshold secret reconstruction to bid opening. The other [1, 19, 23] employs homomorphic encryption algorithms with threshold distributed decryption to seal the bids. Every bidder encrypts each of his bidding selections with a homomorphic encryption algorithm (e.g. ElGamal encryption or Paillier encryption [20]) while the corresponding private (decryption) key is shared among the auctioneers to enable a threshold decryption. The auctioneers apply threshold distributed decryption to bid opening. Obviously, the latter is simpler and more efficient in both computation and communication. Firstly, for each bidding selection in a bid, multiple instances of secret sharing (bid sharing) are needed in the former while only one instance of secret sharing (key sharing) is needed in the latter. Secondly, for each bidding selection in his bid, a bidder has to submit multiple encrypted and signed shares to the auctioneers in the former while he has to submit only one encrypted and signed value to the auctioneers in the latter.

In homomorphic auction schemes, each bidding selection is only valid when it is either "YES" or "NO". As shown in [25] invalid bidding selections (neither "YES" nor "NO") may compromise correctness and fairness of the auction. So validity of the bids must be guaranteed in all the homomorphic auction schemes [13, 15, 1, 19, 26, 24, 23]. In most existing homomorphic auction schemes [13, 15, 26, 1, 19], one integer (usually 0) is chosen to stands for "NO" in a bidding selection and another integer (usually 1) is chosen to stands for "YES" in a bidding selection. In these schemes proved validity of the bids must be proved by the bidders and verified by the auctioneers and independent observers. In a publicly verifiable auction

scheme, the proof must be publicly verifiable. However, proof and verification of bid validity in them is either ignored [13,15,26] or inefficient [1,19].

In [24,23], all the integers in the domain of the sealing function (either secret sharing or encryption) are divided into two subsets. Any integer in one subset stands for "YES" and any integer in the other subset stands for"NO". Thus, in these two schemes there is no invalid bidding selection and no bidder has to prove validity of his bid. However, these two schemes have their own drawbacks. They only support first bid auction, so are not flexible. Moreover, they need more complicated bid opening.

So, there is a problem in homomorphic auction: correctness, fairness, robustness, flexibility and high efficiency cannot be simultaneously guaranteed. To achieve correctness and fairness, either flexibility and robustness is sacrificed (in [24,23]) or costly bid validity check must be employed (in [13,15,1,19,26]). If this problem is not solved, the advantage of homomorphic auction over other sealed-bid auction solutions [18,12,11,8,4,16,30,31,32,29,28] is dubious.

A batched proof and verification technique is used in [25] to solve this problem. It permits a bidder to prove validity of all the bidding selections in his bid in a batch while a verifier can verify his proof in a batch as well. The batched proof and verification are much more efficient than the multiple instances of proof and verification. However, batched proof and verification of bid validity in [25] can only be applied to secret-sharing based homomorphic auction. It does not address simpler, more efficient and popular encryption-based homomorphic auction.

A new homomorphic auction scheme is proposed in this paper to fill the gap. More advanced batched proof and verification of bid validity than that in [25] is designed to batch prove and verify validity of bids in encryption-based homomorphic auction. Firstly, an ElGamal-based homomorphic sealed-bid auction scheme with normal inefficient bid validity check is proposed as a prototype. Then its bid validity check mechanism is optimised using the new batched proof and verification technique. It improves efficiency of encryption-based homomorphic auction compared to [1,19,26], but does not have the drawbacks of [24,23].

2 Symbols and Parameters

The following symbols and parameters will be used in this paper.

- Suppose there are n bidders $B_1, B_2, \ldots, B_n$ and w biddable prices $p_1, p_2, \ldots, p_w$ from the highest to the lowest.
- $E()$ denotes encryption; $D()$ denotes decryption.
- $\langle x \rangle$: the bit length of integer x.
- Two large primes p and q are chosen, such that $p = 2q+1$ and $w < q$. Integer g_0 is a generator of Z_p^*. Integers g and h are generators of G, the subgroup of Z_p^* with order q.
- Unless specified, all the multiplication calculations in this paper is modulo p.

– **Definition 1.** $|\ |$ *is the absolute-value function from* Z_p^* *to* G *defined as*

$$|\sigma| = \begin{cases} \sigma & \text{if } \sigma \in G \\ g_0^q \sigma & \text{if } \sigma \in Z_p^* \setminus G \end{cases}$$

– Let L be a security parameter such that $2^L < q$.

3 Background

Homomorphic auction and a cryptographic tool to be employed later, batch proof and verification, are introduced in this section.

3.1 Homomorphic Auction and Bid Validity Check

The idea of homomorphic auction is simple: to exploit homomorphism of a bid sealing function to implement efficient bid opening while no bid is revealed. We abstract, supplement and summarize the existing ρ^{th} bid homomorphic auction schemes [13,15,1,19,26] as follows where unlike in [25] both secret-sharing based schemes and encryption-based schemes are taken into account. Suppose $S()$ is a sealing function and a message m is sealed into $c = S(m)$. $S()$ must be additive homomorphic, namely $S(m_1)S(m_2) = S(m_1 + m_2)$ and $S(m)^t = S(tm)$ for any messages m, m_1, m_2 and any integer t. As stated in Section 1, one-selection-per-price principle is employed and every bidder has to make a choice at every biddable price. Suppose the bidding selections at a price are $m_1, m_2, \ldots, m_n$ where m_i is the selection of B_i. $m_i = 1$ implies that P_i is willing to pay this price; $m_i = 0$ implies that P_i is unwilling to pay this price. Bid opening at this price is described as follows.

1. The unsealing power (trap-door) is shared by multiple auctioneers such that bid opening is feasible if and only if the number of cooperating auctioneers is over a pre-defined threshold.
2. m_i is sealed into $c_i = S(m_i)$ for $i = 1, 2, \ldots, n$ when being submitted.
3. The auctioneers cooperate to unseal $(\prod_{i=1}^n c_i)^{t_0}$, $((\prod_{i=1}^n c_i)/S(1))^{t_1}$, $\ldots$ $((\prod_{i=1}^n c_i)/S(\rho-1))^{t_{\rho-1}}$ after randomly shuffling them where t_j for $j = 0, 1, \ldots, \rho-1$ are secret random integers corporately chosen and shared by the auctioneers. If any of the unsealed results is zero, the number of non-zero selections is smaller than ρ at this price. If all the unsealed results are non-zero, the number of non-zero selections is at least ρ at this price.

Binary search can be employed to search for the winning price among the biddable prices. If the number of non-zero selections is found to be smaller than ρ at the currently searched price, the search goes on to lower prices; if the number of non-zero selections is at least ρ at the currently searched price, the search goes on to higher prices. The winning price can be found after bid opening is performed at about $\langle w \rangle$ prices, while no bid is unsealed. Note that binary search

requires that the bidding selections in a bid must be consistent, namely non-zero selections must be at lower prices than zero selections in a bid. If the sealing function is a one-way trap-door function, no bid is revealed. This idea is generally adopted by most homomorphic auction schemes [13,15,1,19,26] although most of them simplify bid opening at Step 3 by directly unsealing $\prod_{i=1}^{n} c_i$, which sacrifices complete bid privacy for higher efficiency.

Peng *et al* [25] illustrate that in homomorphic auction bid validity check is necessary for correctness and fairness of the auction by presenting an attack using invalid bids to compromise correctness and fairness in homomorphic auction. Their viewpoint is correct in general as [23] and [24] are very special homomorphic auction schemes. Although homomorphic auction schemes in [23] and [24] do not need bid validity check, they have three drawbacks: 1)they only supports first bid auction and cannot be extended to other auction rules; 2)they employ more complicated bid opening function than the traditional homomorphic auction schemes [13,15,1,19,26]; 3)robustness may be compromised when binary search is employed as inconsistent bidding selections (non-zero selections are at higher prices than zero selections in a bid) cannot be detected. Although their efficiency trade-off (saving the cost of bid validity check at the cost of complicated bid opening) is successful and these two schemes are more efficient than the traditional homomorphic auction schemes [13,15,1,19,26], they lack flexibility and robustness.

In the new auction scheme in this paper, ElGamal encryption is employed while 0 and 1 are used to stand for "YES"' and "NO" in a bid selection. To achieve flexibility and robustness, bid validity check is needed to guarantee that the correct format is used in every bid selection. As binary search is employed in the new auction scheme in this paper, bid validity also requires that the bidding selections in any bid must be consistent. However, we need a new bid validity check mechanism more efficient than the existing one. The existing bid validity check mechanism in [1,19] employs *zero knowledge proof of partial knowledge* by Cramer *et al* [7] to prove and verify bid validity, which has a high cost: $O(w)$ full-length exponentiations for each bidder's proof and $O(nw)$ full-length exponentiations for any verifier (e.g. each auctioneer). Although the batched bid validity check in [25] is efficient, it cannot be applied to the new scheme. The batched bid validity check in [25] only supports secret-sharing-based homomorphic auction, while ElGamal encryption (bid sealing by which is simpler and more efficient than that by secret sharing) is employed in the new scheme to seal the bids.

3.2 1-out-of-w Oblivious Transfer

A 1-out-of-w oblivious transfer protocol is needed in this paper. In our application only one value is transferred using the 1-out-of-w oblivious transfer protocol, so the 1-out-of-w oblivious transfer protocol is performed only once and security requirement for multiple transfers need not be considered. However, the employed 1-out-of-w oblivious transfer protocol must be very efficient to achieve high efficiency. So the 1-out-of-w oblivious transfer in [25] is employed, which is

based on RSA encryption and the 1-out-of-2 oblivious transfer protocol in [12]. This 1-out-of-w oblivious transfer protocol only provides one-time security, but is very efficient.

The sender has w secrets s_1, s_2, $\ldots s_w$ and the chooser wants to know $s_\delta \in \{1, 2, \ldots, w\}$. They run the following protocol where the chooser's operation is denoted as $OT1(s_\delta)$ and the sender's operation is denoted as $OT2(s_\delta)$.

1. **Initialisation:** The sender sets up RSA encryption, keeps private key d and publishes public key e and N' where $ed = 1 \bmod \phi(N')$. He randomly selects $e_{i,j}$ from $Z_{N'}$ for $i = 1, 2, \ldots, \log_2 w$ and $j = 0, 1$ and publishes $s'_l = s_l - \prod_{i=1}^{\log_2 w} e_{i,b_{l,i}} \bmod N'$ for $l = 1, 2, \ldots w$ where $b_{l,i}$ denotes the i^{th} bit of l. He chooses randomly $c_i \in Z_{N'}$ for $i = 1, 2, \ldots \log_2 w$ and sends them to the chooser.
2. **OT1: choosing a secret**
   The chooser chooses secrets $\tau_i \in N'$ for $i = 1, 2, \ldots \log_2 w$. He calculates public keys $y_{i,b_{\delta,i}} = \tau_i^e \bmod N'$ for $i = 1, 2, \ldots, \log_2 w$. Then he calculates $y_{i,1\oplus b_{\delta,i}} = y_{i,b_{\delta,i}} c_i \bmod N'$ if $b_{\delta,i} = 0$ or $y_{i,1\oplus b_{\delta,i}} = y_{i,b_{\delta,i}}/c_i \bmod N'$ if $b_{\delta,i} = 1$ for $i = 1, 2, \ldots \log_2 w$ where $\oplus$ stands for XOR. He sends $y_{i,0}$ and $y_{i,1}$ for $i = 1, 2, \ldots \log_2 w$ to the sender in correct order.
3. **OT2: sending the secret**
   The sender verifies $y_{i,1} = c_i y_{i,0} \bmod N'$ for $i = 1, 2, \ldots \log_2 w$ and sends $E_{i,0} = y_{i,0}^d e_{i,0} \bmod N'$ and $E_{i,1} = y_{i,1}^d e_{i,1} \bmod N'$ for $i = 1, 2, \ldots \log_2 w$ to the chooser.
4. **Obtaining the secret:** The chooser can only get $s_\delta = s'_\delta + (\prod_{i=1}^{\log_2 w} E_{i,b_{\delta,i}})/\prod_{i=1}^{\log_2 w} \tau_i \bmod N'$.

The following properties of this 1-out-of-w oblivious transfer protocol are demonstrated in [25].

1. Correctness: If the sender and the chooser follow the protocol, the chooser can obtain s_δ as

$$s'_\delta + (\prod_{i=1}^{\log_2 w} E_{i,b_{\delta,i}})/\prod_{i=1}^{\log_2 w} \tau_i = s'_\delta + (\prod_{i=1}^{\log_2 w} y_{i,b_{\delta,i}}^d e_{i,b_{\delta,i}})/\prod_{i=1}^{\log_2 w} \tau_i$$
$$= s'_\delta + (\prod_{i=1}^{\log_2 w} y_{i,b_{\delta,i}}^d e_{i,b_{\delta,i}})/\prod_{i=1}^{\log_2 w} \tau_i = s'_\delta + (\prod_{i=1}^{\log_2 w} \tau_i^{ed} e_{i,b_{\delta,i}})/\prod_{i=1}^{\log_2 w} \tau_i$$
$$= s'_\delta + \prod_{i=1}^{\log_2 w} e_{i,b_{\delta,i}} = s_\delta \bmod N'$$

2. Privacy of the chooser: As $y_{i,0}$ and $y_{i,1}$ for $i = 1, 2, \ldots \log_2 w$ are distributed uniformly, the sender has no knowledge of δ. Namely, information-theoretic privacy of the chooser is achieved.
3. Privacy of the sender: It is widely believed that composite factorization is intractable and without the knowledge of the factorization of n it is intractable to find d given e and n. So the chooser can get only one of $e_{i,0}$ and $e_{i,1}$ for every i in $[1, \log_2 w]$. Therefore, the chooser does not know any other secret than s_δ.
4. High efficiency:

- the cost to the sender is $2\log_2 w$ exponentiations and $n(\log_2 w - 1) + 2\log_2 w$ multiplications;
- the cost to the chooser is $(\log_2 w)/2 + 1$ divisions and $1.5\log_2 w + 1$ multiplications on average if e is a small integer as suggested in [12].

In this paper, this 1-out-of-w oblivious transfer protocol will be applied to transfer an L-bit integer where $N' > 2^L$.

3.3 Batch Proof and Verification of Equality of Logarithms

Batch verification was first described formally and in detail by Bellare, who proposed three protocols: RS (random subset) test, SE (small exponent) test and Bucket test [2]. Bellare's batch verification techniques are used to improve efficiency of verification of multiple common base exponentiations. Bellare's tests are only sound when certain integers involved in the verification are in a special subgroup. It is illustrated in [3] that as membership test in the special subgroup is inefficient, Bellare's techniques cannot improve efficiency in the application he proposed. Batch verification is extended to batch zero knowledge proof and verification of secret knowledge statements in [27], which also solves the membership test problem. Theorem 1 is proposed in [27] to batch prove and verify equality of logarithms without need of membership test: $\log_g |y_i| = \log_h |z_i|$ where $y_i \in Z_p^*$ and $z_i \in Z_p^*$ for $i = 1, 2, \ldots, w$.

Theorem 1. *Suppose $y_i \in Z_p^*$ and $z_i \in Z_p^*$ for $i = 1, 2, \ldots, w$. Let t_i for $i = 1, 2, \ldots, w$ be random integers such that $t_i < 2^L$. If $\log_g \prod_{i=1}^{w} y_i^{t_i} = log_h \prod_{i=1}^{w} z_i^{t_i}$ with a probability larger than 2^{-L}, then $\log_g |y_i| = \log_h |z_i|$ for $i = 1, 2, \ldots, w$.*

According to Theorem 1, proof and verification of $\log_{g_1} |y_i| = \log_{g_2} |z_i|$ for $i = 1, 2, \ldots, w$ can be batched to proof and verification of $\log_g \prod_{i=1}^{w} y_i^{t_i} = log_h \prod_{i=1}^{w} z_i^{t_i}$ when $t_1, t_2, \ldots, t_n$ are randomly chosen. If $\log_{g_1} |y_i| \neq \log_{g_2} |z_i|$ for any i in $\{1, 2, \ldots, w\}$, $\log_g \prod_{i=1}^{w} y_i^{t_i} = log_h \prod_{i=1}^{w} z_i^{t_i}$ with only negligible probability.

4 Prototype of the New Scheme

As mentioned before, in homomorphic auction encryption-based bid sealing is simpler and more efficient than secret-sharing-based bid sealing. So the new homomorphic auction scheme employs encryption-based bid sealing. As sharing the private key of Paillier encryption among the auctioneers without revealing it to any single party (including trusted third party) is very complex and impractical, ElGamal encryption instead of Paillier encryption (often used in the existing encryption-based homomorphic auction schemes) is employed in the new auction scheme in this paper. As shown later in this section, sharing the private key of ElGamal encryption among the auctioneers without revealing it to any single party is quite simple and efficient. Although ElGamal encryption is not additive homomorphic as required in the existing encryption-based homomorphic auction

schemes, efficient homomorphic bid opening is implemented based on it in the new scheme. Suppose there are m auctioneers $A_1, A_2, \ldots, A_m$. The prototype auction protocol is as follows.

1. Each auctioneer A_j chooses his private key x_j from Z_q for $j = 1, 2, \ldots, m$. Each A_j publishes $y_j = g^{x_j}$ and shares his key among the auctioneers using a t-out-of-m secret sharing technique (e.g. [21]). Each auctioneer sums up his shares from all the auctioneers and obtain his share of the decryption key, which is $x = \sum_{j=1}^{m} x_j \bmod q$. The public key is $y = \prod_{j=1}^{m} y_j \bmod p$. A message s is encrypted into $(g^r \bmod p,\ sy^r \bmod p)$ where r is randomly chosen from Z_q and a ciphertext $c = (a, b)$ is decrypted into $b/\prod_{j=1}^{t} s_j \bmod p$ where $s_j = a^{x_j} \bmod p$ is provided by A_j and for simplicity $A_1, A_2, \ldots, A_t$ are supposed to be t honest auctioneers. A_j can demonstrate validity of s_j by proving $\log_g y_j = \log_a s_j$.
2. Each bidder B_i chooses his bid b_i from $\{1, 2, \ldots, w\}$ and builds his bidding vector $(m_{i,1}, m_{i,2}, \ldots, m_{i,w})$ where $m_{i,b_i} = 1$ and all the other components are zeros.
3. Each B_i uses ElGamal encryption to encrypt $(g^{m_{i,1}}, g^{m_{i,2}}, \ldots, g^{m_{i,w}})$ into $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ where $c_{i,l} = (a_{i,l},\ b_{i,l}) = (g^{s_{i,l}} \bmod p,\ g^{m_{i,l}} y^{s_{i,l}} \bmod p)$ and $s_{i,l}$ is randomly chosen from Z_q for $l = 1, 2, \ldots, w$.
4. Each B_i proves

$$\log_g \prod_{l=1}^{w} a_{i,l} = \log_y((\prod_{l=1}^{w} b_{i,l})/g) \tag{1}$$

   using ZK proof of equality of logarithms [5] and

$$\log_g a_{i,l} = \log_y b_{i,l} \ \vee\ \log_g a_{i,l} = \log_y(b_{i,l}/g) \text{ for } l = 1, 2, \ldots, w \tag{2}$$

   by repeating for $l = 1, 2, \ldots, w$ the proof protocol in Figure 1, which is a combination of ZK proof of equality of logarithms [5] and ZK proof of partial knowledge [7].
5. $c'_{i,l} = (a'_{i,l},\ b'_{i,l}) = (\prod_{k=1}^{l} a_{i,k} \bmod p,\ \prod_{k=1}^{l} b_{i,k} \bmod p)$ are calculated for $i = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, w$. This operation transforms the bids into the one-selection-per-price format, so that a binary search can be performed.
6. The auctioneers perform a binary search for the winning price. At each price p_l on the binary search route, t honest auctioneers among $A_1, A_2, \ldots, A_m$ cooperate to decrypt

$$\begin{gathered}((\prod_{i=1}^{n} a'_{i,l})^{r_{l,0}} \bmod p,\ (\prod_{i=1}^{n} b'_{i,l})^{r_{l,0}} \bmod p),\\ ((\prod_{i=1}^{n} a'_{i,l})^{r_{l,1}} \bmod p,\ ((\prod_{i=1}^{n} b'_{i,l})/g)^{r_{l,1}} \bmod p),\\ ((\prod_{i=1}^{n} a'_{i,l})^{r_{l,2}} \bmod p,\ ((\prod_{i=1}^{n} b'_{i,l})/g^2)^{r_{l,2}} \bmod p), \ldots\\ ((\prod_{i=1}^{n} a'_{i,l})^{r_{l,\rho-1}} \bmod p,\ ((\prod_{i=1}^{n} b'_{i,l})/g^{\rho-1})^{r_{l,\rho-1}} \bmod p)\end{gathered}$$

   after re-encrypting and shuffling them (e.g. using the publicly verifiable shuffling scheme in [10]). After the decryption, each A_j proves that his partial decryption is correct. If any of the decryption result is zero, the search goes

To prove $\log_g a_{i,l} = \log_y b_{i,l} \ \vee \ \log_g a_{i,l} = \log_y (b_{i,l}/g)$

where $\log_g a_{i,l} = \log_y (b_{i,l}/g^k) = \theta$ and $k = 0$ or 1

1. B_i publishes α_0, β_0, α_1 and β_1 where

$$\alpha_{1-k} = g^{v_{1-k}} a_{i,l}^{u_{1-k}} \bmod p$$
$$\beta_{1-k} = y^{v_{1-k}} (b_{i,l}/g^{1-k})^{u_{1-k}} \bmod p$$
$$\alpha_k = g^r \bmod p$$
$$\beta_k = y^r \bmod p$$

   and r, u_{1-k}, v_{1-k} are randomly chosen from Z_q.
2. A verifier (e.g. acted by the auctioneers corporately) publishes a random integer u.
3. B_i publishes u_1, u_2, v_1 and v_2 where

$$u_k = u - u_{1-k} \bmod q$$
$$v_k = r - u_k \theta \bmod q$$

   Public verification:

$$\alpha_0 = g^{v_0} a_{i,l}^{u_0} \bmod p$$
$$\beta_0 = y^{v_0} b_{i,l}^{u_0} \bmod p$$
$$\alpha_1 = g^{v_1} a_{i,l}^{u_1} \bmod p$$
$$\beta_1 = y^{v_1} (b_{i,l}/g)^{u_1} \bmod p$$
$$u = u_0 + u_1 \bmod q$$

Fig. 1. Normal method to prove (2)

on to lower prices; if all the decrypted results are non-zero, the search goes on to higher prices. After visiting $\langle w \rangle$ prices, the binary search stops at the winning price.

7. All the selections at the price just higher than the winning price (in the case of first bid auction, all the selections at the winning price) are decrypted by the auctioneers while each participating auctioneer proves that his partial decryption is correct. All the bidders with a non-zero selection at that price are winners.

It is easy to check that if a bidder's bid is valid, he can always successfully prove (1) and (2). Soundness of ZK proof of equality of logarithms [5] and ZK proof of partial knowledge [7] guarantee that when the number of biddable prices is smaller than q (which is always satisfied in practice) with an overwhelmingly large probability a bid is valid if the bidder's proof of (1) and (2) can pass public verification. Honest verifier zero knowledge property of ZK proof of equality of logarithms [5] and ZK proof of partial knowledge [7] guarantee that no bid is revealed in this bid validity check mechanism. In this prototype, bid opening

is efficient compared to bid validity check and only costs each auctioneer $O(n)$ full-length exponentiations. Each bidder costs $2w$ full-length exponentiations in bid encryption, which is acceptable. However, bid validity check is too inefficient. Although proof and verification of (1) is efficient, the proof and verification of (2) in Figure 1 costs a bidder much higher cost than bid encryption and a verifier at least $O(wn)$ full-length exponentiations. This is a very high cost and an efficiency bottleneck of the auctions scheme.

5 Efficiency Optimisation

As shown in last section, the prototype is too inefficient. Its cost for a bidder's proof and an auctioneer's verification in bid validity check is too high. An efficiency optimisation is proposed in this section. To apply batch proof and verification to improve efficiency, two modifications are made to the prototype.

- The format of the bidding vector is slightly changed. $m_{i,l}$ is encrypted into $(a_{i,l}, b_{i,l}) = (\mu g^{r_{i,l}} \bmod p,\ \nu g^{m_{i,l}} y^{r_{i,l}} \bmod p)$ where B_i randomly chooses $r_{i,l} \in Z_q$, $\mu = 1$ or $g_0^q \bmod p$ and $\nu = 1$ or $g_0^q \bmod p$.
- Proof and verification of (2) in the prototype is optimised using batch proof-verification techniques and 1-out-of-w oblivious transfer. An optimised proof and verification protocol for (2) is described in Figure 2 where $m_{i,\delta} = 1$ and $m_{i,l} = 0$ for $l = 1, 2, \ldots, \delta - 1, \delta + 1, \ldots, w$. According to Theorem 1 and the privacy property for the sender in the 1-out-of-w oblivious transfer in Section 3.2, the protocol in Figure 2 proves

$$\log_g |a_{i,l}| = \log_y |b_{i,l}| \text{ for } w - 1 \text{ instances of } l \text{ where } l \in \{1, 2, \ldots, w\}$$

  without revealing the bid.
- The decryption function is modified: a ciphertext $c = (a, b)$ is decrypted into $|b / \prod_{j=1}^{t} s_j \bmod p|$ where $s_j = a^{x_j} \bmod p$ is provided by A_j.

$B_i \rightarrow$ Auctioneers : $OT_1(t_\delta)$

Auctioneers $\rightarrow B_i$: $OT_2(t_\delta)$

$B_i \rightarrow$ Auctioneers : $\alpha = g^r \bmod p,\ \beta = y^r g^{t_\delta} \bmod p$ where r is random.

Auctioneers $\rightarrow B_i$: $t_1, t_2, \ldots, t_w$ where $t_l \in Z_{2^L}$ for $l = 1, 2, \ldots, w$.

$B_i \rightarrow$ Auctioneers : $z = s - r \bmod q$ where $s = \sum_{l=1}^{w} r_{i,l} t_l \bmod q$

Verification: $\prod_{l=1}^{w} a_{i,l}^{t_l} = \alpha g^z \bmod p$, $\prod_{l=1}^{w} b_{i,l}^{t_l} = \beta y^z \bmod p$

Fig. 2. Batched Proof and Verification of Bid Validity

The other operations in the prototype are not changed. This new bid valid check in addition to Proof of (1), guarantees that

- $|D(c_{i,l})| = 1$ for $w-1$ cases of l in $\{1, 2, \ldots, w\}$;
- $|D(c_{i,l})| = g$ for 1 case of l in $\{1, 2, \ldots, w\}$.

6 Analysis

Security of the proposed auction scheme, especially its bid validity check mechanism, is analysed in this section. Theorem 2 together with the privacy property of the 1-out-of-w oblivious transfer protocol proved in [25] guarantees that the proof protocol in Figure 2 is private.

Theorem 2. *The last three steps in the proof protocol in Figure 2 are honest-verifier zero knowledge.*

Proof: A party without any secret knowledge can simulate $(\alpha, \beta, t_1, t_2, \ldots, t_w, z)$, the proof transcript for the last three steps, as follows.

1. Randomly choose integers z from Z_q and $t_1, t_2, \ldots, t_w$ from $\{0, 1, \ldots, 2^L - 1\}$.
2. Calculates $\alpha = (\prod_{l=1}^{w} a_{i,l}^{t_l})/g^z \bmod p$ and $\beta = (\prod_{l=1}^{w} b_{i,l}^{t_l})/y^z \bmod p$.

If the challenges are randomly chosen in the proof protocol in Figure 2, then in both the simulated transcript and the proof transcript in Figure 2, α, β are uniformly distributed in Z_p^*, $t_1, t_2, \ldots, t_w$ are uniformly distributed in $\{0, 1, \ldots, 2^L - 1\}$ and z is uniformly distributed in Z_q. Therefore, the two transcripts have the same distribution and are indistinguishable. □

Theorem 3. *The proof protocol in Figure 2 is specially sound. More precisely, if in Figure 2 the challenges are randomly chosen and the proof passes the verification with a probability no smaller than $2^{-wL} + 2^{-L}$, the proof together with (1) guarantees that $(g, 1, 1 \ldots, 1)$ is encrypted in $c_{i,1}, c_{i,2}, \ldots, c_{i,w}$ after being permuted.*

To prove Theorem 3, the following two lemmas are proved first.

Lemma 1. *If a prover can pass the protocol in either Figure 3 or Figure 2, then he can pass the other as well.*

Proof: As the oblivious transfer protocol guarantees privacy of sender, in the protocol in Figure 2 the prover only gets t_δ in the first two steps and gets no information about $t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ until the fourth step. So in both the protocols in Figure 2 and Figure 3 the prover tries to prove the same statement while given the same knowledge. Therefore, he can pass the other as well if he can pass one of them. □

Lemma 2. *The protocol in Figure 3 is specially sound. More precisely, if in Figure 3 the challenges are randomly chosen and the prover can pass the verification with a probability no smaller than $2^{-wL} + 2^{-L}$, then $\log_g |a_{i,l}| = \log_y |b_{i,l}|$ for $l = 1, 2, \ldots, \delta - 1, \delta + 1, \ldots, w$.*

$B_i \rightarrow$ Auctioneers : δ

Auctioneers $\rightarrow B_i$: t_δ,

$B_i \rightarrow$ Auctioneers : $\alpha = g^r \bmod p,\ \beta = y^r g^{t_\delta} \bmod p$ where r is random.

Auctioneers $\rightarrow B_i$: $t_1, t_2, \ldots, t_w$ where $t_l \in Z_{2^L}$ for $l = 1, 2, \ldots, w$.

$B_i \rightarrow$ Auctioneers : $z = s - r \bmod q$ where $s = \sum_{l=1}^{w} r_{i,l} t_l \bmod q$

Verification: $\prod_{l=1}^{w} a_{i,l}^{t_l} = \alpha g^z \bmod p$, $\prod_{l=1}^{w} b_{i,l}^{t_l} = \beta y^z \bmod p$

Fig. 3. A protocol used in proof of Theorem 3

Proof: As the prover can pass the verification in the protocol in Figure 3 with a probability larger than $2^{-wL} + 2^{-L}$, the prover must be able to give two responses z and z' to two different challenges $t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ and $t'_1, t'_2, \ldots, t'_{\delta-1}, t'_{\delta+1}, \ldots, t'_w$ to the same commitment a and the same t_δ, such that

$$\prod_{l=1}^{w} a_{i,l}^{t_l} = \alpha g^z \bmod p \tag{3}$$

$$\prod_{l=1}^{w} b_{i,l}^{t_l} = \beta y^z \bmod p \tag{4}$$

$$(\prod_{l=1}^{\delta-1} a_{i,l}^{t'_l}) a_{i,\delta}^{t_\delta} \prod_{l=\delta+1}^{w} a_{i,l}^{t'_l} = \alpha g^{z'} \bmod p \tag{5}$$

$$(\prod_{l=1}^{\delta-1} b_{i,l}^{t'_l}) b_{i,\delta}^{t_\delta} \prod_{l=\delta+1}^{w} b_{i,l}^{t'_l} = \beta y^{z'} \bmod p \tag{6}$$

with a probability larger than 2^{-L}. Otherwise, the prover can give correct response to at most one challenge with a probability larger than 2^{-L}. This deduction implies when a random challenge is raised the probability that the prover can pass the verification is no more than

$$0 \times P(E_0) + p_1 P(E_1) + p_2 P(E_2)$$

where p_1 is a probability larger than 2^{-L}, p_2 is a probability no larger than 2^{-L}, E_0 denotes the event that the prover can give correct response to zero challenge with a probability larger than 2^{-L}, E_1 denotes the event that the prover can give correct response to one challenge with a probability larger than 2^{-L} and that challenge happens to be chosen, E_2 denotes the event that the prover can give correct response to one challenge with a probability larger than 2^{-L} and that challenge is not chosen. As

$$0 \times P(E_0) + p_1 P(E_1) + p_2 P(E_2) = p_1 2^{-wL} + p_2(1 - 2^{-wL}) < 2^{-wL} + 2^{-L}$$

there is a contradiction to the assumption that the prover can pass the verification in the protocol in Figure 3 with a probability larger than $2^{-wL} + 2^{-L}$.

Table 1. Efficiency comparison of non-interactive auction schemes with bid privacy

Auction	Flexi-	Robust-	Bidder		Auctioneer	
scheme	-bility	-ness	multiplication	example	multiplication	example
[18, 12, 11, 8, 4, 16, 28]	Yes	Yes	$\geq 3072 \log_2 w + 1536$	≥ 38400	$\geq 337920n \log_2 w$	≥ 4055040000
[29]	Yes	Yes	$\geq (1.5w + 1)1536 + n(0.5L + 1) + 4609$	≥ 11492329	$\geq (0.5w(n + 3) + 1)1536 + 2304n + n(0.5L + 1) + 1$	≥ 3158751721
[13, 15, 6, 14, 1, 19] with bid validity check	Yes	Yes	$\geq 12291.5w + 1536$	≥ 50346140	$\geq 12292nw + (10752 + 2n) \log_2 L + 1536(0.5n + 2) + 1537$	≥ 50348957633
[23]	No	No	$1.5w + 1536$	7680	average $((0.5n + 1)m(T_1 + T_2) + (1.5m + 0.125n - 1)T_1T_2 + 1.25nT_2 + 0.5T_1 + 1) \log_2 w + 1536(1 + (0.5T_1 + 1) \log_2 w) + 1$	2338021
[24]	No	No	$\geq (w(2m + t + 3) + 1)$ 1536	≥ 69207552	$\geq 1536(5n + 5 \log_2 w + 1)$	≥ 7773696
[25]	Yes	Yes	$t(0.5wL + 3w + L + 750 \log_2 w + 10730)$	218994	$n((3025 + 1047t) \log_2 w + 2L + wL + w \log_2 w + 14112) + 4502 \log_2 w$	219000024
New scheme	Yes	Yes	$0.5wL + 3w + L + 769.5 \log_2 w + 10755$	73257	$n(3074 \log_2 w + 2L + wL + w \log_2 w + 13828) + 4608 \log_2 w$	181683296

Dividing (3) with (5) yields

$$\prod_{1\le l\le w, l\neq\delta} a_{i,l}^{t_l-t'_l} = g^{z-z'} \bmod p$$

and dividing (4) with (6) yields

$$\prod_{1\le l\le w, l\neq\delta} b_{i,l}^{t_l-t'_l} = y^{z-z'} \bmod p$$

both of which are correct with a probability larger than 2^{-L}. So $\log_g \prod_{1\le l\le w, l\neq\delta} a_{i,l}^{t_l-t'_l} = \log_y \prod_{1\le l\le w, l\neq\delta} b_{i,l}^{t_l-t'_l}$ with a probability larger than 2^{-L}.

As the challenges $t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ and $t'_1, t'_2, \ldots, t'_{\delta-1}, t'_{\delta+1}, \ldots, t'_w$ are randomly chosen, $t_1 - t'_1, t_2 - t'_2, \ldots, t_{\delta-1} - t'_{\delta-1}, t_{\delta+1} - t'_{\delta+1}, \ldots, t_w - t'_w$ are random. So according to Theorem 1,

$$\log_g |a_{i,l}| = \log_y |b_{i,l}| \text{ for } l = 1, 2, \ldots, \delta-1, \delta+1, \ldots, w.$$

□

Proof of Theorem 3: According to Lemma 1, if the proof protocol in Figure 2 passes the verification with a probability no smaller than $2^{-wL} + 2^{-L}$ when the challenges are random, the prover can also give a proof and pass the verification in Figure 3 with the same probability when the challenges are random.

So according to Lemma 2, $c_{i,1}, c_{i,2}, \ldots, c_{i,\delta-1}, c_{i,\delta+1} \ldots, c_{i,w}$ encrypt 1. Proof (1) guarantees that $\prod_{l=1}^{w} D(c_{i,l}) = D(\prod_{i=1}^{w} c_{i,l}) = g$. So $c_{i,\delta}$ encrypts g. Therefore, $(g, 1, 1 \ldots, 1)$ is encrypted in $c_{i,1}, c_{i,2}, \ldots, c_{i,w}$ after being permuted. □

Comparison of computational cost between the existing non-interactive auction schemes with bid privacy and the new auction scheme is made in Table 1 where multiplications are counted. Auction schemes in [30,31] are not included as they require $O(w)$ rounds of interactive communication between the bidders and auctioneers, so are not practical. The auction schemes in [9,32,17] are not included as they do not (or at least do not strictly) protect bid privacy. Any full-length integer is assumed to be 1024 bits long. An exponentiation with an x-bit exponent is regarded as $1.5x$ multiplications and product of y exponentiations with x-bit exponents is regarded as $x + 0.5xy$ multiplications. In [23], T_1 and T_2 are length of challenges and cutting factor in cut-and-choose mechanism. In [24], t is the sharing threshold in the secret sharing mechanism used to share the bids among the auctioneers. An example of the efficiency comparison is also given in the table, where $n = 1000$, $w = 4096$, $m = 5$, $T_1 = T_2 = 20$, $t = 3$ and $L = 20$. It is clearly illustrated that the new scheme is more efficient than any flexible and robust non-interactive sealed-bid auction scheme with bid privacy.

7 Conclusion

A correct, fair, private and robust homomorphic sealed-bid auction scheme is proposed. It employs ElGamal encryption and an original batched bid validity

check mechanism to achieve high efficiency without compromising flexibility and robustness. The new scheme is more efficient than any flexible and robust non-interactive sealed-bid auction scheme with bid privacy.

References

1. M Abe and K Suzuki. M+1-st price auction using homomorphic encryption. In *PKC '02*, volume 2288 of *Lecture Notes in Computer Science*, pages 115–124.
2. M Bellare, J Garay, and T Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250.
3. C Boyd and C Pavlovski. Attacking and repairing batch verification schemes. In *ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 58–71.
4. C Cachin. Efficient private bidding and auctions with an oblivious third party. In *the 6th ACM Conference on Computer and Communications Security*, 1999.
5. D Chaum and T Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105.
6. K Chida, K Kobayashi, and H Morita. Efficient sealed-bid auctions for massive numbers of bidders with lump comparison. In *ISC '01*, volume 2200 of *Lecture Notes in Computer Science*, pages 408–419.
7. R Cramer, I B. Damgård, and B Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187.
8. R Cramer, I Damgård, and J Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299.
9. M Franklin and M Reiter. The design and implementation of a secure auction service. In *IEEE Transactions on Software Engineering*, volume 5, pages 302–312, May 1996.
10. J Groth. A verifiable secret shuffle of homomorphic encryptions. In *PKC '03*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160.
11. M Jakobsson and A Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 143–161.
12. A Juels and M Szydlo. A two-server, sealed-bid auction protocol. In *FC '02*, volume 2357 of *Lecture Notes in Computer Science*, pages 72–86.
13. H Kikuchi, M Harkavy, and J Tygar. Multi-round anonymous auction. In *Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, pages 62–69, June 1998.
14. H Kikuchi. (m+1)st-price auction. In *FC '01*, volume 2339 of *Lecture Notes in Computer Science*, pages 291–298.
15. H Kikuchi, S Hotta, K Abe, and S Nakanishi. Distributed auction servers resolving winner and winning bid without revealing privacy of bids. In *IEEE NGITA '00*, pages 307–312, July 2000.
16. K Kurosawa and W Ogata. Bit-slice auction circuit. In *ESORICS '02*, volume 2502 of *Lecture Notes in Computer Science*, pages 24–38.
17. H Lipmaa, N Asokan, and V Niemi. Secure vickrey auctions without thresh-old trust. In *FC '02*, volume 2357 of *Lecture Notes in Computer Science*, pages 87–101.

18. M Naor, B Pinkas, and R Sumner. Privacy perserving auctions and mechanism design. In *ACM Conference on Electronic Commerce 1999*, pages 129–139, 1999.
19. K Omote and A Miyaji. A second-price sealed-bid auction with the discriminant of the p-th root. In *FC '02*, volume 2357 of *Lecture Notes in Computer Science*, pages 57–71.
20. P Paillier. Public key cryptosystem based on composite degree residuosity classes. In *EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238.
21. T Pedersen. Distributed provers with applications to undeniable signatures. In *EUROCRYPT '91*, pages 221–242. Lecture Notes in Computer Science 547.
22. T Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *EUROCRYPT '91*, pages 129–140. Lecture Notes in Computer Science 547.
23. K Peng, C Boyd, and E Dawson. A multiplicative homomorphic sealed-bid auction based on Goldwasser-Micali encryption. In *ISC '05*, volume 3650 of *Lecture Notes in Computer Science*, pages 374–388.
24. K Peng, C Boyd, and E Dawson. Optimization of electronic first-bid sealed-bid auction based on homomorphic secret sharing. In *Mycrypt '05*, volume 3715 of *Lecture Notes in Computer Science*, pages 84–98.
25. K Peng, C Boyd, and E Dawson. Batch verification of validity of bids in homomorphic e-auction. *Computer Communications 29 (2006) 2798-2805*, 2006.
26. K Peng, C Boyd, E Dawson, and K Viswanathan. Robust, privacy protecting and publicly verifiable sealed-bid auction. In *ICICS '02*, volume 2513 of *Lecture Notes in Computer Science*, pages 147 – 159.
27. K Peng, C Boyd, E Dawson, and K Viswanathan. A correct, private and efficient mix network. In *PKC '04*, volume 2947 of *Lecture Notes in Computer Science*, pages 439–454.
28. K Peng, C Boyd, E Dawson, and K Viswanathan. Non-interactive auction scheme with strong privacy. In *ICISC '02*, volume 2587 of *Lecture Notes in Computer Science*, pages 407 – 420.
29. K Sako. An auction scheme which hides the bids of losers. In *PKC '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 422–432.
30. K Sakurai and S Miyazaki. A bulletin-board based digital auction scheme with bidding down strategy -towards anonymous electronic bidding without anonymous channels nor trusted centers. In *Proc. IWCTE '99*, pages 180–187, Hong Kong, 1999. City University of Hong Kong Press.
31. K Suzuki, K Kobayashi, and H Morita. Efficient sealed-bid auction using hash chain. In *ICISC '00*, volume 2015 of *Lecture Notes in Computer Science*, pages 183–191.
32. Y Watanabe and H Imai. Reducing the round complexity of a sealed-bid auction protocol with an off-line ttp. In *ACM STOC 2000*, pages 80–86. 2000.

How to Prevent DPA and Fault Attack in a Unified Way for ECC Scalar Multiplication – Ring Extension Method

Yoo-Jin Baek and Ihor Vasyltsov

Embedded SW Center, Samsung Electronics, Korea
{yoojin.baek, ihor.vasiltsov}@samsung.com

Abstract. The elliptic curve cryptosystem(ECC) is increasingly being used in practice due to its shorter key sizes and efficient realizations. However, ECC is also known to be vulnerable to various side channel attacks, including power attacks and fault injection attacks. This paper proposes new countermeasures for ECC scalar multiplications against differential power attacks and fault attacks. The basic idea of proposed countermeasures lies in extending the definition field of an elliptic curve to its random extension ring and performing the required elliptic curve operations over the ring. Moreover, new methods perform a point validation check in a small subring of the extension ring to give an efficient fault attack countermeasure.

1 Introduction

While increasingly being used in practice due to its shorter key sizes and lower processing time, ECC [13,19] is also known to be vulnerable to various side channel attacks, including power attacks [14,7] and fault injection attacks [2,4,5].

The power attack finds out some secret information from power consumption signals. Several power attacks against ECC have been introduced so far, including the simple power attack(SPA) and the differential power attack(DPA). Accordingly, various countermeasures have also been proposed. For example, the double-and-add always method [7] and the Montgomery powering ladder [12,15,20] can be used as an SPA countermeasure. On the other hand, some random blinding techniques are proposed as a DPA countermeasure in [6,7,11].

The fault injection attack tries to find out secret information after introducing some faults inside a device and then analyzing the resulting faulty outputs. Hence, its typical countermeasure is to check the validity of input or output values, which, in ECC, may amount to checking if input or output points of an elliptic curve scalar multiplication satisfy the original elliptic curve equation.

This paper proposes new ECC countermeasures against such power attacks and fault attacks, which are mainly based on the idea of the Shamir's method [22]. More precisely, proposed countermeasures extend the definition field of an elliptic curve to its random extension ring, modify the elliptic curve equation accordingly and then perform the required elliptic curve operations over the extension ring and the modified curve. A fault detection mechanism is also inserted

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 225–237, 2007.

into the algorithms, which takes place in a small subring of the extension ring, not in the original field, to minimize the computational overhead.

This paper is organized as follows. In Section 2, we briefly give an overview of elliptic curves. We continue with the basics in Section 3 by going through various attack methods and their countermeasures. Section 4 is the place in which we clearly present new countermeasures and analyze their computational cost.

Comparison with Previous Countermeasures. There are some well-known DPA countermeasures, among which the followings will be considered: the random exponent blinding method using the property $dP = (d + r \cdot \#E)P$, the random message blinding technique such that $dP = d(P + R) - dR$ and the random point representation method using the property that $(x; y; z) = (rx; ry; rz)$ for any nonzero field element r in the projective coordinate [7], the random elliptic curve isomorphism method and the random field isomorphism method in [11], the $2P^*$ method and the multiplier randomization method in [6]. Firstly, it is noted in [1,9] that the random point representation method, the random elliptic curve isomorphism method, the random field isomorphism method and the $2P^*$ method may be weak to special kinds of power attacks, say, the refined power-analysis attack(RPA) and the zero-value point attack(ZPA). Secondly, the multiplier randomization technique has a disadvantage in memory usage when applied to a window method. More precisely, if applied to a window method of width w, it requires the storage of $O(2^{w+1})$ elliptic curve points, which is comparable with that of $O(2^w)$ points for the proposed algorithms. Thirdly, the random exponent blinding method may be weak to a new kind of fault attacks, the sign change fault attack [4]. Lastly, the random message blinding technique of its original form (i.e. one first chooses a random point R and stores it with the point $S = dR$. And then, for each new execution, one refreshes them by computing $R \leftarrow (-1)^b 2R$ and $S \leftarrow (-1)^b 2S$ for a random bit b.) was shown to be weak to the doubling attack [8]. Other realizations of the method may be considered to avoid the doubling attack. For example, the refreshing phase can be performed such a way that $R \leftarrow (-1)^b rR$ and $S \leftarrow (-1)^b rS$ for a fixed integer $r \neq 2$ and a random bit b. But, even in this case, if a fault can be injected in the refreshing phase, the scheme may also leak the secret scalar d. On the other hand, the proposed countermeasures are secure to all side-channel attacks mentioned above and have reasonable performance penalty (as the size of the definition field increases). See Table 1 for more details.

Another countermeasure to be considered is the method in [4] which is devised to prevent the so called sign change fault attack and adopts the Shamir's idea for its implementation as well, i.e. it transforms the definition field to an extension ring. But, compared with proposed methods, the transformation is not a random one, i.e. the resulting extension ring has a fixed form, hence it cannot be used as a DPA countermeasure. Also, it performs two ECC scalar multiplications in its execution to counter the sign change fault attack, one over the extended ring and the other over its subring, while proposed methods perform only one scalar multiplication over an extension ring.

In summary, the previous countermeasures have their own disadvantages in performance or in security if they are used individually, hence they must be combined with each other to give a more complete countermeasure. For example, a message blinding technique can be combined with a exponent blinding method. And the proposed countermeasures have contributions as providing an alternative to message blinding methods, not to mention an efficient unification of DPA and fault attack countermeasures.

2 Elliptic Curves

For a power of prime q, let $\mathbb{F}_q$ denote the finite field with q elements and let $\overline{\mathbb{F}}_q$ be its algebraic closure. An *elliptic curve* E over $\mathbb{F}_q$ is the set of points $(x, y) \in \overline{\mathbb{F}}_q \times \overline{\mathbb{F}}_q$ which satisfy the following nonsingular Weierstrass equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in \mathbb{F}_q \tag{1}$$

plus a point at infinity $\mathcal{O}$. And, for any extension field K of $\mathbb{F}_q$ in $\overline{\mathbb{F}}_q$, $E(K)$ is defined to be subset of E consisting of K-rational points (including $\mathcal{O}$), i.e.

$$E(K) = \{(x, y) \in E | x, y \in K\} \cup \{\mathcal{O}\}.$$

For convenience, $E(K)$ is occasionally denoted by E as long as there is no confusion.

It is well known that $E(K)$ forms an Abelian group under a certain addition rule. For example, if the characteristic of q is 2 and the elliptic curve is not supersingular, the equation (1) can be transformed to

$$y^2 + xy = x^3 + ax^2 + b, \tag{2}$$

using an appropriate elliptic curve isomorphism and its addition formula can be explicitly given by: $\mathcal{O}$ acts as an identity element and for $P = (x_1, y_1) \in E(K) \setminus \{\mathcal{O}\}$ and $Q = (x_2, y_2) \in E(K) \setminus \{\mathcal{O}, -\mathcal{P}\}$, $-P = (x_1, x_1 + y_1)$ and $P + Q = (x_3, y_3)$ is defined by

$$x_3 = \begin{cases} (\dfrac{y_1 + y_2}{x_1 + x_2})^2 + \dfrac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, & \text{if } P \neq Q \\ x_1^2 + \dfrac{b}{x_1^2}, & \text{if } P = Q, \end{cases} \tag{3}$$

$$y_3 = \begin{cases} (\dfrac{y_1 + y_2}{x_1 + x_2})(x_1 + x_3) + x_3 + y_1, & \text{if } P \neq Q \\ x_1^2 + (x_1 + \dfrac{y_1}{x_1})x_3 + x_3, & \text{if } P = Q. \end{cases} \tag{4}$$

A similar addition rule for other elliptic curves can be found in [17].

One of the most time-consuming operations in ECC is the multiplication by a scalar. If k is a positive integer and P is an elliptic curve point, the scalar multiplication kP is the operation adding k copies of P and $(-k)P$ is defined as $k(-P)$.

3 Attack Models

3.1 Power Attacks

The power attack, which was first introduced by Kocher et al. [14], tries to recover some secret information from power consumption signals. Several power attacks against ECC have been proposed so far, which include the simple power attack(SPA), the differential power attack(DPA) [7], the refined power-analysis attack(RPA) [9], the zero-value point attack(ZPA) [1] and the doubling attack [8].

SPA observes power signals for a single execution of cryptographic operations, from which it tries to distinguish between various cryptographic primitives (for example, point addition and point doubling in ECC). Accordingly, the double-and-add always method [7] and the Montgomery method [12,15,20] have been introduced as its countermeasure.

DPA collects the power consumption data and uses statistical tools to get some useful information from these data. As its countermeasure, a variety of methods were introduced so far, including the random exponent blinding method, the random message blinding technique and the random point representation method [7]. Also, the random elliptic curve isomorphism method which transforms an elliptic curve into its isomorphic curve, the random field isomorphism technique converting the definition field of an elliptic curve into its isomorphic field [11], the $2P^*$ method which changes the representation of $2P$ (not P) into its isomorphic form and the multiplier randomization method using the property that $d = \lfloor d/r \rfloor r + (d \mod r)$ [6] play their roles as a DPA-countermeasure. But, as noted in Introduction, some methods have disadvantage in performance and others have weakness against some other side channel attacks.

RPA, using the property that processing points with zero coordinates has a different power consumption profiling, first chooses a special point of which a coordinate is equal to zero and then inputs into a target device a point which is equal to the special point if it is multiplied by a specific scalar. ZPA refines RPA and uses a zero-value register instead of a zero-value coordinate. Both attacks, however, work only if the intermediate results of corresponding scalar multiplication algorithms can partially or wholly be guessed. Hence, they can be avoided if appropriate message and/or exponent randomization techniques are applied to scalar multiplication algorithms.

The doubling attack is based on the hypothesis that an adversary can tell whether two intermediate results of two distinct operations are identical or not from their power curves. Hence, if random message and/or exponent blinding techniques are used, the attack can also be avoided as for RPA and ZPA.

3.2 Fault Attacks

Fault attacks [2,4,5] are very powerful cryptanalytic tools, the basic idea of which is to induce some intentional faults inside a device and then to analyze the faulty outputs to get some meaningful information. Hence, a typical countermeasure

against them is to check the validity of input or output values, which, in ECC, may amount to checking if an input or output points of elliptic curve scalar multiplications satisfy the original elliptic curve equation.

Another brilliant fault attack countermeasure, especially for the RSA cryptosystem, was proposed by Shamir [22]. Since proposed countermeasures are strongly related to his idea, the method will briefly be introduced here. For a given ciphertext c, a modulus $N = pq$ and a secret exponent d, to securely calculate $m = c^d \pmod N$ against fault attacks, the Shamir's method first chooses a small random prime r and calculates two quantities m'_p and m'_q with $m'_p = m^{d \pmod{(p-1)(r-1)}} \pmod{pr}$ and $m'_q = m^{d \pmod{(q-1)(r-1)}} \pmod{qr}$. And, after checking the validity of $m'_p = m'_q \pmod r$, it computes m_p and m_q such that $m_p = m'_p \pmod p$ and $m_q = m'_q \pmod q$. Finally, one can easily get the plaintext m using the Chinese remainder theorem(CRT) since m must satisfy the equations $m = m_p \pmod p$ and $m = m_q \pmod q$. Even though it may be weak to another kind of fault attacks which induces faults in the CRT recombination phase, the Shamir's method is original in that it avoids an expensive modular exponentiation in the checking step.

Recently, a new kind of fault attacks, the so called sign change fault attack, was introduced in [4]. The attack is especially worth being considered since it tries to make a sign change in an elliptic curve point representation and so the changed point does not leave the curve, which means that the attack is not easily detectable in the usual point validation scheme.

4 New DPA and Fault Attack Countermeasures for ECC

This section proposes new DPA and fault attack countermeasures for ECC which are based on the idea of the Shamir's method [22]. However, unlike the Shamir's method which acts as a fault attack countermeasure for the RSA cryptosystem, the new methods work for ECC scalar multiplications and serves as a DPA and fault attack countermeasure.

4.1 Prime Field Case

To propose a new countermeasure for an elliptic curve over a prime field, we begin with presenting some adding and doubling formulae for the elliptic curve of the form

$$Y^2Z + pYZ^3 = X^3 + aXZ^4 + bZ^6$$

over the ring $\mathbb{Z}_{pr} \simeq \mathbb{Z}/(pr)$. Note that the curve is presented in the Jacobian coordinate and the equation is equivalent to $y^2 + py = x^3 + ax + b$ in the affine coordinate.

For $P_1 = (X_1; Y_1; Z_1)$ and $P_2 = (X_2; Y_2; Z_2)$ with $P_1, P_2 \neq \mathcal{O}$, $P_3 := P_1 + P_2 = (X_3; Y_3; Z_3)$ can be obtained as follows: if $P_1 \neq \pm P_2$,

$$\begin{cases} Z_3 &= Z_1Z_2(X_2Z_1^2 - X_1Z_2^2) \\ X_3 &= (Y_2Z_1^3 - Y_1Z_2^3)^2 - (X_2Z_1^2 - X_1Z_2^2)^2(X_2Z_1^2 + X_1Z_2^2) \\ Y_3 &= (Y_2Z_1^3 - Y_1Z_2^3)\big(X_1Z_2^2(X_2Z_1^2 - X_1Z_2^2)^2 - X_3\big) \\ & \quad -(Y_1 + pZ_1^3)\big(Z_2^3(X_2Z_1^2 - X_1Z_2^2)^3\big) \end{cases}, \tag{5}$$

which will be denoted as $P_3 = \text{ADD-JP}(P_1, P_2)$ and if $P_1 = P_2$,

$$\begin{cases} Z_3 &= Z_1(2Y_1 + pZ_1^3) \\ X_3 &= (3X_1^2 + aZ_1^4)^2 - 2X_1(2Y_1 + pZ_1^3)^2 \\ Y_3 &= (3X_1^2 + aZ_1^4)\big(X_1(2Y_1 + pZ_1^3)^2 - X_3\big) - (Y_1 + pZ_1^3)(2Y_1 + pZ_1^3)^3 \end{cases}, \tag{6}$$

which $P_3 = \text{DBL-JP}(P_1)$ stands for. Note that ADD-JP(P_1, P_2) and DBL-JP(P_1) require 19 and 13 $\mathbb{Z}_{pr}$-multiplications, respectively, which is comparable with 16 and 10 $\mathbb{F}_p$-multiplications for an elliptic curve $Y^2Z = X^3 + aXZ^4 + bZ^6$ over $\mathbb{F}_p$ [3]. Hence, the computational overhead of ADD-JP(P_1, P_2) and DBL-JP(P_1) over the usual may be estimated as $6/(16{+}10) = 23\%$, disregarding the effect of the increased ring size. Using these formulae, the following SPA-resistant scalar multiplication algorithm can be obtained:

Algorithm 1. (Double-and-Add Always Method [7])

Input:
$E : Y^2 + pYZ^3 = X^3 + aXZ^4 + bZ^6$, an elliptic curve over $\mathbb{Z}_{pr}$
$P = (x; y; 1) \in E(\mathbb{Z}_{pr})$
$d = \sum_{i=0}^{n-1} d_i 2^i$: a secret scalar
Output: $Q = dP \in E(\mathbb{Z}_{pr})$

1. $Q[0] \leftarrow P$
2. For $i = n - 2$ to 0 by -1, do
   2.1 $Q[0] \leftarrow \text{DBL} - \text{JP}(Q[0])$
   2.2 $Q[1] \leftarrow \text{ADD} - \text{JP}(Q[0], P)$
   2.3 $Q[0] \leftarrow Q[d_i]$
3. Return $Q[0]$.

Remark 1. If the above 23% computational overhead is not acceptable, more efficient SPA countermeasures may be considered. For example, the Montgomery powering ladder in [12,15,20] performs a scalar multiplication using only X- and Z-coordinates of elliptic curve points. Consequently, for an elliptic curve $E : Y^2Z + pYZ^2 = X^3 + aXZ^2 + bZ^3$ and $P = (x; y; 1), P_1 = (X_1; Y_1; Z_1), P_2 = (X_2; Y_2; Z_2) \in E(\mathbb{Z}_{pr})$ with $P = P_2 - P_1$ (in the ordinary projective coordinate), the $X-$ and $Z-$coordinates of $P_3 := P_1 + P_2 = (X_3; Y_3; Z_3)$ can be obtained as: if $P_1 \neq \pm P_2$,

$$\begin{cases} X_3 &= 2X_1^2X_2Z_2 + 2X_1X_2^2Z_1 + 2aX_1Z_1Z_2^2 + (4b + p^2)Z_1^2Z_2^2 \\ & \quad +2aX_2Z_1^2Z_2 - x(X_2Z_1 - X_1Z_2)^2 \\ Z_3 &= (X_2Z_1 - X_1Z_2)^2 \end{cases}, \tag{7}$$

and if $P_1 = P_2$,

$$\begin{cases} X_3 = (X_1^2 - aZ_1^2)^2 - 2X_1Z_1^3(4b + p^2) \\ Z_3 = Z_1\big(4X_1^3 + 4aX_1Z_1^2 + (4b + p^2)Z_1^3\big) \end{cases}. \tag{8}$$

Note that the corresponding formulae for an elliptic curve $Y^2Z = X^3 + aXZ^2 + bZ^3$ over $\mathbb{F}_p$ have the form

$$\begin{cases} X_3 = 2(X_1^2X_2Z_2 + X_1X_2^2Z_1 + aX_1Z_1Z_2^2 + 2bZ_1^2Z_2^2 + aX_2Z_1^2Z_2) \\ \qquad -x(X_2Z_1 - X_1Z_2)^2 \\ Z_3 = (X_2Z_1 - X_1Z_2)^2 \end{cases}, \tag{9}$$

and

$$\begin{cases} X_3 = (X_1^2 - aZ_1^2)^2 - 8bX_1Z_1^3 \\ Z_3 = 4Z_1\big(X_1^3 + aX_1Z_1^2 + bZ_1^3\big) \end{cases}. \tag{10}$$

Hence, disregarding the effect of the increased ring size, the extra work of formulae (7) and (8) over (9) and (10) is just the computation of p^2. However, since p is known in advance and so the value p^2 can be precomputed, the overhead may be considered as negligible.

Now, let E be an elliptic curve defined over $\mathbb{F}_p$ such that

$$E : Y^2Z = X^3 + aXZ^4 + bZ^6.$$

To compute $dP \in E(\mathbb{F}_p)$ for $P \in E(\mathbb{F}_p)$ and an integer d of size $\log p$, we first choose a random small integer r and let

$$B = y^2 + py - x^3 - ax \pmod{pr}$$

for $P = (x; y; 1)$. Clearly, for the elliptic curve E' over $\mathbb{Z}_{pr}$ defined by

$$E' : Y^2Z + pYZ^3 = X^3 + aXZ^4 + BZ^6\,, \tag{11}$$

the point P is contained in $E'(\mathbb{Z}_{pr})$ and B is equal to b modulo p, which implies that E' is exactly the same curve with E over $\mathbb{F}_p$. Now, a subsequent scalar multiplication dP is performed on the curve E' and the resulting point will be reduced modulo p later. The detailed algorithm is given as follows:

Algorithm 2. (New Countermeasure for Prime Field)

Input:
p, a prime defining the field $\mathbb{F}_p$
$E : Y^2Z = X^3 + aXZ^4 + bZ^6$, an elliptic curve over $\mathbb{F}_p$
$P = (x; y; 1) \in E(\mathbb{F}_p)$
d, a secret scalar

Output: $dP \in E(\mathbb{F}_p)$

1. Choose a small random integer r.
2. $B \leftarrow y^2 + py - x^3 - ax \pmod{pr}$
3. Let $E' : Y^2Z + pYZ^3 = X^3 + aXZ^4 + BZ^6$.
4. Calculate dP over $E'/\mathbb{Z}_{pr}$, using Algorithm 1.
5. Check if $Y^2 + pYZ^3 = X^3 + aXZ^4 + BZ^6 \pmod{r}$ for $dP = (X; Y; Z)$. If the equality does not hold, return $\mathcal{O}$.
6. Return $(X \mod p; Y \mod p; Z \mod p)$.

At this point, we analyze the security of Algorithm 2. First, the reason why the algorithm provides a resistance against DPA is that the representation of intermediate points has a random redundancy because the scalar multiplication in Step 4 takes place in a random extension ring $\mathbb{Z}_{pr}$. Also, even though a coordinate of a point has a zero value in $\mathbb{F}_p$, it is not necessary for it to be zero in $\mathbb{Z}_{pr}$, which is the origin of RPA- and ZPA-resistance of the above algorithm. Finally, since Algorithm 2 performs a validity check at Step 5, it gives rigidity to fault attacks. However, it is emphasized that the check occurs in $\mathbb{Z}_r$, not in $\mathbb{Z}_{pr}$, which saves the computation time in some degree.

There are some remarks about Algorithm 2. First, the scalar multiplication in Step 4 must be performed in the projective coordinate, not in the affine coordinate. The reason is that the denominators appearing in a point addition or a point doubling formula (for example, $x_1 + x_2$ or x_1 in equations (3) and (4)) may not have the multiplicative inverse over $\mathbb{Z}_{pr}$, even though it is not zero in $\mathbb{F}_p$. Second, the factor pYZ^3 in the definition equation (11) of E' is required for the resistance to the so called sign change fault attack [4]. Briefly speaking, the attack uses the property that even if the sign of the Y-coordinate of elliptic curve points is changed, the resulting point is also contained in the curve. Hence, to avoid it, the method in [4] performs two scalar multiplications, one over an extended ring and the other over its subring. However, the proposed countermeasure adds the additional factor pYZ^3 into an elliptic curve equation E', hence, for $Q = (X; Y; Z) \in E'$, $Q' = (X; -Y; Z)$ is generally not contained in E', which violates the prerequisite of the sign change fault attack.

To estimate the computational complexity of Algorithm 2, we analyze each step of Algorithm 2 in some detail. Clearly, the amount of work for Step 1, 5 and 6 is small enough, compared that of Step 4 which performs a full ECC scalar multiplication. Especially, Step 5 and 6 perform a few modular arithmetics with modulus less than pr. Step 2 in its original form is not required in Algorithm 2, because the adding and doubling formulae in equations (5) and (6) do not require the parameter B but the only place using B is Step 5. However, Step 5 only needs the value $B \pmod r$, not $B \pmod{pr}$, hence Step 2 may be changed into the form $B \leftarrow y^2 + py - x^3 - ax \pmod r$, the work of which is negligible if r is chosen to be small to some extent. Clearly, if different adding and doubling formulae are used in Step 4, then Step 2 may be indispensable. For the estimation of Step 4 computation, we need consider two factors, one from the increased ring size and the other from the increased number of required ring multiplications. For the first factor, note that a multiplication over a ring R takes time $O((\log \#R)^2)$, where $\#R$ stands for the size of R. Hence, for an integer d of size $\log p$, calculating dP over $E'(\mathbb{Z}_{pr})$ consumes time $O(\log p(\log p + \log r)^2)$ and so the computational overhead from the first factor can be estimated as

$$\frac{\log p(\log p + \log r)^2 - (\log p)^3}{(\log p)^3} = \frac{2 \log p \log r + (\log r)^2}{(\log p)^2}.$$

As noted earlier, the overhead from the second factor is about 23%. In summary, the total computational overhead estimation of Algorithm 2 is

$$\frac{2\log p \log r + (\log r)^2}{(\log p)^2} + 0.23$$

and this estimation is listed in Table 1 for various NIST recommendation curves [21] and an exemplary random integer r of size 30-bit. As the table shows, the bigger size the definition field has, the lower performance overhead the new countermeasure has. It is also emphasized that if other SPA countermeasures are adopted in Algorithm 2, we can get another countermeasure with lower performance degradation. For example, if the Montgomery method in Remark 1 is used for an SPA countermeasure, the second part for the above performance penalty estimation may be ignored.

Remark 2. In general, generating new random points in elliptic curves, especially over a prime field, is known to be very expensive since it must involve a square-root finding algorithm. For example, it corresponds to computing g^{k+1} in $\mathbb{F}_p$ for $p = 3k + 4$ [10]. Hence, any DPA countermeasure requiring new random points for its execution [7,16] must solve this undesirable problem. The proposed method, however, requires a random integer with moderate size, not a random elliptic curve point, and hence avoids the problem.

4.2 Binary Field Case

A similar reasoning can be applied to a binary field case, but in this case, a random irreducible binary polynomial $r(z)$ must be chosen and an extension ring $\mathbb{F}_2[z]/(f(z) * r(z))$ of $\mathbb{F}_{2^m} \simeq \mathbb{F}_2[z]/(f(z))$ has to be considered. The requirement of irreducibility of $r(z)$ is coming from the following reason: the doubling formula of a binary elliptic curve $Y^2 + XYZ = X^3 + aX^2Z^2 + BZ^6$ in the Jacobian projective coordinate requires a value C such that $C^4 = B$ [10], which can easily be computed in a binary field, but not in an arbitrary ring. For example, C can be calculated as $C = B^{2^{m-1}}$ over $\mathbb{F}_{2^m}$. Hence, over $\mathbb{F}_2[z]/(f(z) * r(z))$ with an irreducible polynomial $r(z)$, we first compute $C_1 = B^{2^{\deg(f)-1}}$ and $C_2 = B^{2^{\deg(r)-1}}$ and then combine them to get C such that $C = C_1 \pmod{f(z)}$ and $C = C_2 \pmod{r(z)}$, using the Chinese Remainder Theorem. Here, $\deg(f)$ denotes the degree of the polynomial $f(z)$. Note that the resulting value C satisfies the equation $C^4 = B$ in $\mathbb{F}_2[z]/(f(z) * r(z))$.

In the subsequent, we need adding and doubling formulae for the elliptic curve of the form

$$Y^2 + XYZ = X^3 + aX^2Z^2 + BZ^6$$

in the Jacobian coordinate over the ring $\mathbb{F}_2[z]/(f(z) * r(z))$ [3]: for $P_1 = (X_1; Y_1; Z_1)$ and $P_2 = (X_2; Y_2; Z_2)$ with $P_1, P_2 \neq \mathcal{O}$, we can get $P_3 := P_1 + P_2 = (X_3; Y_3; Z_3)$ as follows: if $P_1 \neq \pm P_2$,

$$\begin{cases} Z_3 = Z_1Z_2(X_1Z_2^2 + X_2Z_1^2) \\ X_3 = aZ_3^2 + (Y_1Z_2^3 + Y_2Z_1^3)(Y_1Z_2^3 + Y_2Z_1^3 + Z_3) + (X_1Z_2^2 + X_2Z_1^2)^3 \\ Y_3 = \big((Y_1Z_2^3 + Y_2Z_1^3)X_2 + Z_1Y_2(X_1Z_2^2 + X_2Z_1^2)\big)\big((X_1Z_2^2 + X_2Z_1^2)Z_1\big)^2 + \\ \quad (Y_1Z_2^3 + Y_2Z_1^3 + Z_3)X_3 \end{cases},$$

which will be expressed as $P_3 = \text{ADD-JP}(P_1, P_2)$ and if $P_1 = P_2$,

$$\begin{cases} Z_3 = X_1Z_1^2 \\ X_3 = (X_1 + CZ_1^2)^4 \\ Y_3 = X_1^4Z_3 + (Z_3 + X_1^2 + Y_1Z_1)X_3 \end{cases}$$

for $C = \sqrt[4]{B} \pmod{f(z) * r(z)}$ and this equation will be denoted as $P_3 = \text{DBL-JP}(P_1)$. Using these formulae, the following SPA resistant scalar multiplication algorithm can be obtained:

Algorithm 3. (Modified Double-and-Add Always Method)

Input:
$Y^2 + XYZ = X^3 + aX^2Z^2 + BZ^6$, an elliptic curve over $\mathbb{F}_2[z]/(f(z) * r(z))$
$P = (x; y; 1) \in E(\mathbb{F}_2[z]/(f(z) * r(z)))$
$C \in \mathbb{F}_2[z]/(f(z) * r(z))$ such that $B = C^4 \pmod{f(z) * r(z)}$
$d = \sum_{i=0}^{n-1} d_i 2^i$, a secret scalar
Output: $Q = dP \in E(\mathbb{F}_2[z]/(f(z) * r(z)))$

1. $Q[0] \leftarrow P$
2. For $i = n - 2$ to 0 by -1, do
   2.1 $Q[0] \leftarrow \text{DBL} - \text{JP}(Q[0])$
   2.2 $Q[1] \leftarrow \text{ADD} - \text{JP}(Q[0], P)$
   2.3 $Q[0] \leftarrow Q[d_i]$
3. Return $Q[0]$.

Of course, as in the prime field case, more efficient SPA countermeasures can be given using, for example, the Montgomery powering ladder.

Now, let E be an elliptic curve defined over $\mathbb{F}_{2^m} \simeq \mathbb{F}_2[z]/(f(z))$ such that

$$E : Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6.$$

Note that, in this case, the risk of the sign change fault attack need not be considered, since there is already an XYZ factor in the equation. As in the prime field case, a random irreducible binary polynomial $r(z)$ is first chosen and B is computed as

$$B = y^2 + xy + x^3 + ax^2 \pmod{f(z) * r(z)}$$

for $P = (x; y; 1)$. Clearly, for the elliptic curve E' over $\mathbb{F}_2[z]/(f(z) * r(z))$ defined by

$$E' : Y^2 + XYZ = X^3 + aX^2Z^2 + BZ^6,$$

P is contained in $E'(\mathbb{F}_2[z]/(f(z)*r(z))$ and $B = b \pmod{f(z)}$. Hence, over $\mathbb{F}_{2^m}$, E' is exactly same with E. Now, to get dP, a scalar multiplication is performed on the curve E' and the resulting point will be reduced modulo $f(z)$ later to get a final result. The detailed algorithm is given by:

Algorithm 4. (New Countermeasure for Binary Field)

Input:
$f(z)$, an irreducible polynomial defining a field $\mathbb{F}_{2^m} \simeq \mathbb{F}_2[z]/(f(z))$
$E : Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6$, an elliptic curve over $\mathbb{F}_{2^m}$
$P = (x; y; 1) \in E(\mathbb{F}_{2^m})$
d, the secret exponent
Output: $dP \in E(\mathbb{F}_{2^m})$

1. Choose a random irreducible binary polynomial $r(z)$ of small degree.
2. $B \leftarrow y^2 + xy - x^3 - ax^2 \pmod{f(z) * r(z)}$
3. Let $E' : Y^2 + XYZ = X^3 + aX^2Z^2 + BZ^6$.
4. $fInv(z) \leftarrow f(z)^{-1} \pmod{r(z)}$
5. $C_1 \leftarrow b^{2^{m-2}}, C_2 \leftarrow (B \pmod{r(z)})^{2^{\deg(r)-2}}$
6. $C \leftarrow C_1 + (C_2 - C_1) * fInv(z) * f(z)$
7. Calculate $dP = (X; Y; Z)$ for $P = (x; y; 1)$ over $E'(\mathbb{F}_2[z]/(f(z) * r(z)))$, using Algorithm 3.
8. Check if $Y^2 + XYZ = X^3 + aX^2Z^2 + BZ^6 \pmod{r(z)}$ for $dP = (X; Y; Z)$. If the equality does not hold, return $\mathcal{O}$.
9. Return $(X \mod f(z); Y \mod f(z); Z \mod f(z))$.

The same reasoning as for Algorithm 2 can be applied to the resistance to various side channel attacks of Algorithm 4.

To estimate its computational complexity, each step of Algorithm 4 is analyzed in some detail as follows: first, compared with Step 7, the amount of work for Step 8 and 9 is small enough because they perform a few modular arithmetics with polynomials of degree less than $\deg(f(z)*r(z))$. For Step 1, note that generating a random binary irreducible polynomial of degree l takes an expected running time of $O(l^3(\log l))$ binary operations [18], which is negligible for some small l. As in Algorithm 2, Algorithm 4 does not actually need $B \pmod{f(z) * r(z)}$ but $B \pmod{r(z)}$ in Step 5, and so Step 2 may be changed into $B \leftarrow y^2+xy-x^3-ax^2 \pmod{r(z)}$. Hence, the overhead from Step 2 can be made be small. For Step 4 which performs an inverse operation, if $r(z)$ is chosen to have a small degree, its work is not problematic as well. Since Step 5 requires only squarings over binary fields, it is almost free, if a normal basis for binary fields is used. Step 6 must perform 2 regular multiplications for polynomials of degree at most m, hence its amount of work is small, compared with that of Step 7. Consequently, the total computational overhead of Algorithm 4 can be approximated to that from Step 7. However, unlike the prime field case, there is no overhead coming from the

Table 1. Computational Overhead for New Countermeasures

Curve over Prime Field	Computational Overhead	Curves over Binary Field	Computational Overhead
P-192	57 %	B-163	40 %
P-224	52 %	B-233	27 %
P-256	48 %	B-283	22 %
P-384	39 %	B-409	15 %
P-521	35 %	B-571	11 %

increased number of ring multiplications in Step 7. Hence, its overhead can be estimated to be

$$\frac{m(m+\deg(r))^2 - m^3}{m^3} = \frac{2 * \deg(r) * m + \deg(r)^2}{m^2},$$

which is due to the increased ring size. And, this estimation is summarized in Table 1 for an exemplary polynomial $r(z)$ of degree 30.

Remark 3. Since r and $r(z)$ are randomly chosen in Algorithm 2 and 4 respectively, the resulting extension ring does not have any special structure, even though the original field is very specific. For example, the polynomial $f(z)$ defining a finite field $\mathbb{F}_{2^m} \simeq \mathbb{F}_2[z]/(f(z))$ is required to be a trinomial or a pentanomial in many ECC standards. Therefore, if the proposed countermeasures are used for the standards, any gain of computational effort cannot be expected from this kind of structure. But there are also many cases to deal with elliptic curves over a non-specific finite field. For example, consider a system which was first developed to support only the RSA cryptosystem so it only has a hardware module supporting the Montgomery multiplication. Later, if the system is required to support ECC with minimum hardware size and backward compatibility, it is preferable to use the Montgomery multiplier for the arithmetics of $\mathbb{F}_p$ (and $\mathbb{F}_{2^m}$). In this case, the system cannot use the special structure of prime p.

5 Conclusion

This paper proposed new DPA and fault attack countermeasures for ECC scalar multiplications. The basic idea of new countermeasures lies in converting the definition field of elliptic curves into its random extension ring and performing the required operations in the ring. The reason why proposed methods provide resistance to DPA is the introduction of randomness in EC point representations. Moreover, they give an efficient fault attack countermeasure since they perform a validation check in a small subring of the extension ring. An estimation for computational complexity of new algorithms was also provided in the paper.

Acknowledgement. The authors thank the anonymous referees for their helpful comments.

References

1. T. Akishita and T. Takagi, *Zero-Value Point Attacks on Elliptic Curve Cryptosystem*, ISC 2003, LNCS vol. 2851, Springer-Verlag, 2003, pp. 218–233.
2. I. Biehl, B. Meyer, and V. Muller, *Differential Fault Attacks on Elliptic Curve Cryptosystems*, CRYPTO 2000, LNCS vol. 1880, Springer-Verlag, 2000, pp. 131–146.
3. I.F. Blake, G. Seroussi and N.P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.
4. J. Blömer, M. Otto and J.-P. Seifert, *Sign Change Fault Attacks on Elliptic Curve Cryptosystems*, FDTC 2006, LNCS vol. 4236, Springer-Verlag, 2006, pp. 36–52.
5. D. Boneh, R.A. Demillo, and R.J. Lipton, *On the Importance of Checking Cryptographic Protocols for Faults*, EUROCRYPT '97, LNCS vol. 1233, Springer-Verlag, 1997, pp. 37–51.
6. M. Ciet and M. Joye, *(Virtually) Free Randomization Techniques for Elliptic Curve Cryptography*, ICICS 2003, LNCS vol. 2836, Springer-Verlag, 2003, pp. 348–359.
7. J.-S. Coron, *Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems*, CHES 1999, LNCS vol. 1717, Springer-Verlag, 1999, pp. 292–302.
8. P. Fouque and F. Valette, *The Doubling Attack - Why Upwards Is Better Than Downwards*, CHES 2003, LNCS vol. 2779, Springer-Verlag, 2003, pp. 269–280.
9. L. Goubin, *A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems*, PKC 2003, LNCS vol. 2567, Springer-Verlag, 2003, pp. 199–210.
10. Institute of Electrical and Electronics Engineers, *IEEE P1363: IEEE Standard Specifications for Public-Key Cryptography*, 2000.
11. M. Joye and C. Tymen, *Protections against Differential Analysis for Elliptic Curve Cryptography - An Algebraic Approach*, CHES 2001, LNCS vol. 2162, Springer-Verlag, 2001, pp. 377–390.
12. M. Joye and S.-M. Yen, *The Montgomery Powering Ladder*, CHES 2002, LNCS vol. 2523, Springer-Verlag, 2002, pp. 291–302.
13. N. Koblitz, *Elliptic Curve Cryptosystems*, Mathematics Computation, 48, 1987, pp. 203–209.
14. P. Kocher, J. Jaffe, and B. Jun, *Differential Power Analysis*, CRYPTO '99, LNCS vol. 1666, Springer-Verlag, 1999, pp. 388–397.
15. J. López and R. Dahab, *Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation*, CHES 1999, LNCS vol. 1717, Springer-Verlag, 1999, pp. 316–327.
16. H. Mamiya, A. Miyaji and H. Morimoto, *Efficient Countermeasures Against RPA, DPA, and SPA*, CHES 2004, LNCS vol. 3156, Springer-Verlag, 2004, pp. 343–356.
17. A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer, 1993.
18. A.J. Menezes, P.C.van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
19. V.S. Miller, *Use of Elliptic Curves in Cryptography*, CRYPTO 85, LNCS vol. 218, Springer-Verlag, 1986, pp. 417–426.
20. P.L. Montgomery, *Speeding the Pollard and Elliptic Curve Methods of Factorization*, Mathematics of Computation, 48, 1987, pp. 243–264.
21. National Institute of Standards and Technology, *Recommended Elliptic Curves for Federal Government Use, Appendix to FIPS 186-2*, 2000.
22. A. Shamir, *How to check modular exponentiation*, Presented at the rump session of EUROCRYPT '97, 1997.

Secure Signed Radix-r Recoding Methods for Constrained-Embedded Devices

Dong-Guk Han[1], Sung-Kyoung Kim[2], Ho Won Kim[1], Kyo IL Chung[1], and Jongin Lim[2]

[1] Electronics and Telecommunications Research Institute(ETRI), {christa, khw, kyoil}@etri.re.kr
[2] Graduate School of Information Management and Security, Korea University {likesk, jilim}@cist.korea.ac.kr

Abstract. This paper presents two recoding methods for a radix-r representation of a secret scalar which are resistant to SPA. These recoding methods are left-to-right so they can be interleaved with a left-to-right scalar multiplication, removing the need to store both a scalar and its recoding. Next, we show the ideas of left-to-right recoding for a radix-r representation lead to simplified recoding methods for a binary representation. In general our proposed algorithms asymptotically require additional $(w + 1)$-digit and w-bit of RAM in the case of width-w radix-r representation and a special case when $r = 2$, respectively, which is independent from the digit (bit) size n of the scalar and considerably reduces the required space comparing with previous methods which require n-digit (bit) of RAM additional memory to store the recoded scalar. Consequently, thanks to its left-to-right nature, the scalar multiplication based on it is by far more convenient with respect to memory consumption.

Keywords: Side channel attacks, elliptic curve cryptosystems, pairing-based cryptosystems, left-to-right, right-to-left.

1 Introduction

Side channel attacks on implementations of cryptosystems use side channel information such as timings and power consumption measurements to reveal information that is supposed to be kept secret [29,30]. In both elliptic curve cryptosystems and pairing based cryptosystems, a particular common target for side channel attacks are algorithms used for scalar multiplication which is the most dominant computation part. As a general practice, countermeasures must always be used against simple power analysis (SPA), even if using one-time keys. Several countermeasures have been proposed to resist SPA that generates a scalar sequence with a fixed pattern, e.g., $|0...0y|0...0y|...|0...0y|$, where a digit y is chosen from $\{\pm 1, \pm 2, ..., \pm 2^{w-1}, -2^w\}$ in [35] and from $\{\pm 1, \pm 3, ..., \pm(2^w - 1)\}$ in [40,32,47]. These utilized signed binary representation of a secret scalar, however, recently Han-Takagi extend the fixed pattern designed for a binary representation to a radix-r representation for pairing based cryptosystems [16].Here, y is

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 238–253, 2007.

one of the digits contained in $\{\pm 1, \pm 2, ..., \pm(r^w - 1)\} \setminus \{\pm r, \pm 2r, ..., \pm(r^w - r)\}$. The main disadvantage of Han-Takagi's recoding algorithm is that is right-to-left, so it must be computed and stored before the scalar multiplication (see Section 2.1, 2.2).

Let $\Lambda_r = \{0, 1, \ldots, r-1\}$, $D_r = \{\pm 1, \pm 2, \ldots, \pm(r-1)\}$, $\Lambda_{w,r} = \{0, 1, \ldots, r^w - 1\}$, and $D_{w,r} = \{\pm 1, \pm 2, ..., \pm(r^w - 1)\} \setminus \{\pm r, \pm 2r, ..., \pm(r^w - r)\}$. Here, Λ_r, D_r, $\Lambda_{w,r}$, and $D_{w,r}$ are called digit sets.

In this paper, we first present two recoding methods for a radix-r representation which are resistant to SPA. The first one generates D_r representation from a given integer represented by the digit set Λ_r (see Algorithm 2), and the other extends it to the width-w version, which is from $\Lambda_{w,r}$ to $D_{w,r}$ (see Algorithm 4). These recoding methods are left-to-right so they can be interleaved with a left-to-right scalar multiplication, removing the need to store both the scalar and its recoding. The proposed recoding methods recursively need to stores just two consecutive input digits (k_{i+1}, k_i) to generates i-th recoded digit k'_i, namely the new one is carry-free (see Section 4).

Next, we show the ideas of left-to-right recoding for a radix-r representation lead to simplified recoding methods for a binary representation, which only need to store single input bit k_{i+1} to generates i-th recoded digit k'_i in contrast to the radix-r versions (see Algorithm 5, 6 in Section 5).

In general our proposed algorithms asymptotically require additional $(w+1)$-digit and w-bit of RAM in the case of width-w radix-r representation and a special case when $r = 2$, respectively, which is independent from the digit (bit) size n of the scalar and dramatically reduces the required space comparing with previous methods [35,40,32,16] which require n-digit (bit) of RAM additional memory to store the recoded scalar. Consequently, thanks to its left-to-right nature, the scalar multiplication based on it is by far more convenient with respect to memory consumption.

2 Scalar Multiplication

The scalar multiplication kP for a given point P and a scalar k is the most dominant computation part of not only elliptic curve cryptosystems (ECC), but also pairing based cryptosystems (PBC). In general, the scalar k is represented by the (signed) binary chain in ECC, but the radix-r representation is utilized for the efficient implementation of PBC [48]. Because several efficient arithmetic for elliptic curve with characteristic three have been investigated [5,4,18,41,46] and an efficient implementation of Tate pairing for hyper-elliptic curves has been constructed over general characteristic r [14]. Recently, a novel method for computing the Tate pairing efficiently over finite field of characteristic five has been proposed [19]. In these cases, the radix-r representation can be exploited to achieve faster scalar multiplication where the radix-r is equal to the characteristic of finite fields $\mathbf{F}_q$, i.e. $q = r^m$ for some positive integer m. In this paper we mainly focus on efficiency and security of a scalar multiplication over finite fields

Algorithm 1. Radix-r Left-to-Right_Method

Input: A point P, and $k = \sum_{i=0}^{n-1} k_i r^i$, $k_i \in \{0, \pm 1, \cdots, \pm(r-1)\}$.
Output: $Q = kP$.
1: **Pre-computation** $|a|P$ for all $a \in \{1, 2, \cdots, r-1\}$.
2: $Q \leftarrow \mathcal{O}$
3: **for** $j = n-1$ down to 0 **do**
4: $\quad Q \leftarrow rQ$
5: $\quad$ **if** $k_j > 0$ **then** $Q \leftarrow \mathrm{ECADD}(Q, k_j P)$
6: $\quad$ **if** $k_j < 0$ **then** $Q \leftarrow \mathrm{ECADD}(Q, -|k_j|P)$
7: **end for**
8: **return** Q

$\mathbf{F}_{r^m}$. The following algorithm is a standard method to compute elliptic scalar multiplication. $\mathcal{O}$ stands for the point at infinity.

The generalized non-adjacent form (gNAF) [12] and radix-r width-w non-adjacent form (wrNAF) [48] are known as efficient classes of radix r representation because the number of non-zero digits of them are smaller. These recoding methods are carried out from the right to left (i.e., from the most significant position). It is known that gNAF has minimal hamming weight among all signed radix-r representation with digit set $\{0, \pm 1, ..., \pm(r-1)\}$, i.e., the number of non-trivial digits (except $\{0, \pm 1\}$ and ignoring their sign) is $r-2$. The average density of non-zero digits (non-zero density) is asymptotically $\frac{r-1}{r+1}$. In the case of wrNAF which uses the digit set $\{0, \pm 1, ..., \pm\lfloor\frac{r^w-1}{2}\rfloor\} \setminus \{\pm r, \pm 2r, ..., \pm\lfloor\frac{r^{w-1}-1}{2}\rfloor\}$, the non-zero density of it is asymptotically $\frac{r-1}{w(r-1)+1}$. The number of non-trivial digits is $\frac{(r-1)r^{w-1}-2}{2}$. In [48], the number of non-trivial elements and the non-zero density of the conventional sliding window technique for any radix-r larger than 2 (called gSWF) have been investigated, and the results are $(r-1)r^{w-1}-1$ and $\frac{r-1}{w(r-1)+1}$, respectively. From these results, wrNAF is superior to gSWF due to much less storage space requirement, even though both gSWF and wrNAF have equivalent computational performance.

2.1 Advantages of the Scalar Multiplication Carried Out from Left-to-Right

In general, performing scalar multiplication is categorized into two main concepts: left-to-right and right-to-left. Thought both methods provide the same efficiency, the left-to-right method is preferable due to the following reasons [38]:

- The left-to-right method can be adjusted for general representations of k like gSWF or wrNAF in a more efficient way than the right-to-left method.
- The elliptic curve addition and subtraction carried out in step 5 and 6 in Algorithm 1 respectively has the fixed input tP, $t \in \{1, ..., r-2\}$. Therefore it is possible to speed up these steps if tP is expressed in affine coordinates for each t, since some operations are negligible in this case.
- The right-to-left method needs an auxiliary register for storing $r^i P$.

If Algorithm 1 is carried out with gNAF (or wrNAF) representation, a recoding stage, which transforms a given integer to gNAF (or wrNAF) representation, should be added to it. As these recoding methods are done from right to left, it is necessary to finish the recoding and to store the recoded string before starting the left-to-right evaluation stage. In other words, the modified Algorithm 1 can be composed of three stages: *pre-computation* stage, *recoding* stage, and *evaluation* stage.

2.2 Why Left-to-Right Recoding Is More Preferable?

While having looked around ahead, the left-to-right evaluation stage is more natural choice in the case of scalar multiplication. If a given integer string is recoded from right-to-left, e.g. [12,10,44,48], we require additional n-digit (i.e. exponential size $O(n)$) RAM for the right-to-left exponent recoding, where n is the digit size of the scalar. The other way, if recoding of an exponent is carried out from left to right then the recoding and evaluation stage may be merged to obtain an efficient scalar multiplication on the fly, without storing the recoded exponent at all. Thus in connection with memory constraint devices such as smart cards left-to-right recoding schemes are by far more valuable. Several left-to-right recoding techniques have been presented in [2,37,25,26,42].

3 Side Channel Attacks

Side channel attacks (SCA) are allowed to access the additional information linked to the operations using the secret key, e.g., timings, power consumptions, etc [29,30]. The attack aims at guessing the secret key (or some related information). For example, Algorithm 1 can be broken by SCA. For the sake of simplicity, we consider with a special case $r = 2$. It calculates an elliptic addition (ECADD) if and only if the j-th digit k_j is not zero. The standard implementation of ECADD is different from that of the elliptic doubling (ECDBL) [13], and thus the ECADD in the scalar multiplication can be detected using SCA.

If the attacker is allowed to observe the side channel information only a few times, it is called the simple power analysis (SPA). The above SCA is a typical example of SPA. In SPA, the sequence of operations such as ECADD and ECDBL is detected by using side channel information, and the secret is revealed from the sequence. If the attacker can analyze the side channel information several times using a statistical tool, it is called the differential power analysis (DPA). The standard DPA utilizes the correlation function that can distinguish whether a specific bit is related to the observed calculation. In order to resist DPA, we need to randomize the parameters of elliptic curves.

In general, preventing against SPA is essential because of the following two reasons.

- In some cryptographic applications a secret parameter is used only one-time. For example, in the signature generation of ECDSA [1] a secret scalar is chosen randomly and used as a one-time key pair generation. In this case, the attacker does not use the technique of averaging, i.e., DPA is not available.

– Needless to say, SPA countermeasures do not secure against DPA thanks to statistical analysis of several power traces. On the other hand, DPA countermeasures are useless if it is not secure against SPA, so SPA countermeasures should always be used.

We should note that the immunity against DPA is easily obtained by combining with countermeasures of the data randomization type such as randomized projective coordinates method [11,39], random isomorphic curves method [24,22], and the blinding point method [20,36,21].

3.1 SPA Countermeasures Generating a Fixed Pattern of Operations

In this section, we explain several representations of a scalar which are utilized as SPA countermeasures. Indeed, by changing the representation of the scalar without changing its value, one can gain control on the operation sequence, and ultimately, on side-channel information leakage. It aims at generating a scalar sequence that has a fixed pattern, e.g. $|0...0y|0...0y|...|0...0y|$, where y is chosen from a pre-computed table. They try to eliminate the zero runs by additional pre-computing points.

Signed Binary Representations. Let a positive integer e be given in 2^w-ary digits where $w \geq 2$ is a small integer, i.e. $e = \sum_{j=0}^{n} b_j 2^{wj}$ with $b_j \in \{0, 1, ..., 2^w - 1\}$. In order to make a fixed pattern signed representation are advantageous when the computation of inverses is easy. In particular, this is attractive on elliptic curves where $-P$ can be computed from P for almost free.

– In [35], Möller proposed a representation $e' = \sum_{j=0}^{n'} b'_i 2^{wj}$ such that $e = e'$ with $b'_j \in \{\pm 1, \pm 2, ..., \pm 2^{w-1}, -2^w\}$. Here, $n' = n$ or $n' = n+1$. The number of non-trivial digits is 2^{w-1} (except $\{0, \pm 1\}$ and ignoring their sign). The complexity with window size w for computing $e'P$ for a n'-bit scalar e' are as follows;
  - *Pre-computation* stage: $(2^{w-2} - 1)A + (2^{w-2} + 1)D$,
  - *Evaluation* stage: $(\lceil \frac{n'}{w} \rceil - 1)A + \left(w \cdot (\lceil \frac{n'}{w} \rceil - 1)\right) D$,

  where A and D denote the complexity of point addition/subtraction and point doubling, respectively.
– In [40], Okeya-Takagi introduced a new representation with a different digit set $\{\pm 1, \pm 3, ..., \pm(2^w - 1)\}$. Similar result has been proposed in [32]. Contrary to existing methods [35,40,32], recently Theriault has proposed a representation which is carried out from the left to right [47]. It uses the very same pre-computation table utilized in [40,32]. The number of non-trivial digits of them is $2^{w-1} - 1$ (except $\{\pm 1\}$ and ignoring their sign), which is smaller than that of Möller's scheme by one. The complexity of [40,32,47] for computing $e'P$ for a n'-bit scalar e' are as follows;
  - *Pre-computation* stage: $(2^{w-1} - 2)A + D$,
  - *Evaluation* stage: $(\lceil \frac{n'}{w} \rceil - 1)A + \left(w \cdot (\lceil \frac{n'}{w} \rceil - 1)\right) D$.

Signed Radix-r Representation. In [16], Han-Takagi extend Okeya-Takagi scheme to the radix-r representation. It also scans from the least significant digits, i.e. *right to left recoding*. If we find a zero digit, the following conversion is performed: $(\underbrace{0,...,0}_{l},x)_r = (1,\underbrace{\overline{r-1},...,\overline{r-1}}_{l-1},\overline{r-x})_r$ for a positive integer l and $x \in \{1,2,...,r-1\}$. If we apply the width-w right-to-left sliding window method to this chain, we obtain the radix-r analogue of Okeya-Takagi scheme, which has the fixed pattern

$$|\underbrace{0,...,0}_{w-1},y|\underbrace{0,...,0}_{w-1},y|...|\underbrace{0,...,0}_{w-1},y| \tag{1}$$

for $y \in \{\pm 1, \pm 2, ..., \pm(r^w-1)\} \setminus \{\pm r, \pm 2r, ..., \pm(r^w-r)\}$. The size of non-trivial digits is equal to $(r-1)r^{w-1}-1$ (except $\{\pm 1\}$ and ignoring their sign). The complexity of it with window size w for computing kP for a n-digit scalar k are approximately

- *Pre-computation* stage: $\left(\frac{(r-1)r^{w-1}-2}{2}\right)A + \left(\frac{(r-1)r^{w-1}}{2}\right)D$,
- *Evaluation* stage:
  $\left(\lceil\frac{n}{w}\rceil - 1 + w\cdot(\lceil\frac{n}{w}\rceil-1)\cdot\left(\frac{\lceil\log_2 r\rceil}{3}\right)\right)A + \left(w\cdot(\lceil\frac{n}{w}\rceil-1)\cdot\lceil\log_2 r\rceil\right)D$,

where the results of evaluation stage when rQ is computed for some points Q are calculated by using the non-adjacent form (NAF) of r and a signed double-and-add method. The average density of non-zero bits for NAF is 1/3. Note that in characteristic r, point r-tupling (rP) for some curves may be done efficiently because the r-th powering operation is linear in characteristic r. For example, in [5] point tripling for a supersingular curve over characteristic 3 has been proposed and it leads to a triple-and-add scalar multiplication algorithm much faster than the double-and-add method.

4 SPA Resistant Radix-r Left-to-Right Recodings

The major disadvantage of Han-Takagi's recoding algorithm is that is right-to-left, so it must be computed and stored before the scalar multiplication. In this section we present two recoding methods which are resistant to SPA. These recoding methods are left-to-right so they can be interleaved with a left-to-right scalar multiplication, removing the need to store both a scalar and its recoding.

4.1 Notations and Assumptions

In this section we define some notations and an assumption used for this paper.

- Λ_r=$\{0,1,\ldots,r-1\}$, D_r=$\{\pm 1,\pm 2,\ldots,\pm(r-1)\}$, $\Lambda_{w,r} = \{0,1,\ldots,r^w-1\}$, and $D_{w,r}$=$\{\pm 1,\pm 2,...,\pm(r^w-1)\}\setminus\{\pm r,\pm 2r,...,\pm(r^w-r)\}$. Here, Λ_r, D_r, $\Lambda_{w,r}$, and $D_{w,r}$ are called digit sets.
- Let us denote $-a$ by $\overline{a}$ for any positive integer a.

- Given an integer k, it is represented using the radix-r representation with length n, i.e. $k = \sum_{j=0}^{n-1} k_j r^j$ with $k_j \in \Lambda_r$ and $k_{n-1} \neq 0$.
- $(a_{l-1}, a_{l-2}, ..., a_1, a_0)_r := \sum_{j=0}^{l-1} a_j r^j$, where a digit a_j is allowed to take a negative value (e.g., $a_j \in D_r \cup \{0\}$).

Assumption 1. *We assume* $k \bmod r \neq 0$*, namely* $k_0 \neq 0$*.*

It is always possible to force the least significant digit of k, i.e. k_0, to be nonzero. If $k_0 = 0$ or 1, then the scalar is converted to $k' = k+1$. If not, it is converted to $k' = k-1$. This procedure provides $k' \bmod r \neq 0$. Then, the scalar multiplication kP of a given point P is recovered by performing subtraction $(k'P - P)$ or addition $(k'P + P)$, respectively.

Assumption 2. *ECADD and ECDBL are distinguishable by a single measurement of power consumption, whereas ECADD and ECSUB are indistinguishable.*

The standard implementation of ECADD is different from that of ECDBL [13], however, ECADD and ECSUB are very similar, since $\mathrm{ECSUB}(Q, P) = \mathrm{ECADD}(Q, -P)$[1]. Therefore, the above assumption is realistic.

4.2 Left to Right Recoding from Λ_r to D_r

In this section we propose a left-to-right recoding which translates a given integer k represented with the digit set Λ_r to a recoded integer $k' = \sum_{j=0}^{n-1} k'_j r^j$ with $k'_j \in D_r$ such that $k = k'$.

A natural Right-to-Left recoding: When we consider a representation of k as one of its signed radix-r representations, i.e. using the digit set $D_r \cup \{0\}$, different signed radix-r representations of k can be obtained by using following local conversion rules of two consecutive digits;

$$\mathit{Conversion}\ 1: \begin{cases} (0,1)_r \Leftrightarrow (1,\overline{r-1})_r & (0,\bar{1})_r \Leftrightarrow (\bar{1}, r-1)_r \\ (0,2)_r \Leftrightarrow (1,\overline{r-2})_r & (0,\bar{2})_r \Leftrightarrow (\bar{1}, r-2)_r \\ \vdots & \vdots \\ (0,r-1)_r \Leftrightarrow (1,\bar{1})_r & (0,\overline{r-1})_r \Leftrightarrow (\bar{1},1)_r \end{cases}$$

It is an easy task to generate a recoded integer k' with *Conversion* 1, however, the direction of recoding is *right to left* because of the carry-over ± 1 occurring. For example, consider $(1,0,2,0,0,1,0,2)_3$ which is one of radix-3 representations of 8-digit. The process of recoding is as follows: $(1,0,2,0,0,1,0,2)_3 \Rightarrow (1,0,2,0,0,1,\underline{1,\bar{1}})_3 \Rightarrow (1,0,2,0,\underline{1,\bar{2}},1,\bar{1})_3 \Rightarrow (1,0,2,\underline{1,\bar{2}},\bar{2},1,\bar{1})_3 \Rightarrow (1,\underline{1,\bar{1}},1,\bar{2},\bar{2},1,\bar{1})_3$. Namely, we obtained a D_3 representation of $(1,0,2,0,0,1,\underline{0,\bar{2}})_3$ which is converted from the least significant bit.

[1] Let $p = (x, y)$. Then, $-P = (x, -y)$ and $(x, x + y)$ when the characteristic of $\mathbf{F}_q$ is not 2, 3 and is 2, respectively.

Proposed Left-to-Right Recoding: In contrast to the previous generation method, we present a recoding rule recursively, which needs to store just two consecutive input digits (k_{i+1}, k_i) to generates i-th recoded digit k'_i, namely the new one is carry-free. In order to consider k'_{n-1} we assume $k_n = 1$.

$$\textbf{Recoding 1:} \quad k'_i = \begin{cases} k_i & \text{if } k_{i+1} \cdot k_i \neq 0; \\ 1 & \text{if } k_{i+1} \neq 0 \text{ and } k_i = 0; \\ k_i - r & \text{if } k_{i+1} = 0 \text{ and } k_i \neq 0; \\ 1 - r & \text{if } k_{i+1} = 0 \text{ and } k_i = 0. \end{cases}$$

Define

$$\mathbb{RECODE}(a, b) : \Lambda_r \times \Lambda_r \longrightarrow D_r$$

as a mapping with two input digits in Λ_r outputs one of D_r according to the above recoding rules. For example, $\mathbb{RECODE}(0, 1) = \overline{2}$ and $\mathbb{RECODE}(2, 0) = 1$ when $r = 3$. Define a multiple mapping for a positive integer $l \geq 2$

$$\begin{aligned} \mathbb{MRECODE}[a_l, ..., a_1, a_0] : \underbrace{\Lambda_r \times \cdots \times \Lambda_r}_{(l+1)-tuple} &\longrightarrow \underbrace{D_r \times \cdots \times D_r}_{l-tuple} \\ (a_l, ..., a_1, a_0) &\longmapsto (b_{l-1}, ..., b_1, b_0) \end{aligned}$$

such that b_i, the i-th element of $\mathbb{MRECODE}[a_l, ..., a_1, a_0]$, is determined as $b_i = \mathbb{RECODE}(a_{i+1}, a_i)$.

Theorem 1. *For a given integer k represented with Λ_r, a recoded integer k' by Recoding 1 is the same as the D_r representation of k recoded by Conversion 1, say k''. Consequently, $k' = k''$.*

Proof. From the recoding rule of Recoding 1, clearly k'_i is an element of D_r, thus, k' is a D_r representation. Write k as

$$\begin{aligned} k &= (k_{n-1}, k_{n-2}, ..., k_1, k_0)_r \\ &= \mathbf{A}^{m-1} \| \mathbf{A}^{m-2} \| \cdots \| \mathbf{A}^1 \| \mathbf{A}^0, \text{ where } \mathbf{A}^i = \begin{cases} \textit{block of nonzero digits, or} \\ (\underbrace{0, 0, \ldots, 0}_{t \geq 1}, x), \ x \in \Lambda_r \setminus \{0\} \end{cases} \end{aligned}$$

for any integer $m \geq 1$. When $m = 1$, that is $\mathbf{A}^0$ is a n-digit string of nonzero digits, then clear. Consider the case when $m > 1$.

- *When k is recoded by Conversion 1*:
  If $\mathbf{A}^i$ is the first case, i.e. a block of nonzero digits, then there is no change from Conversion 1 in $\mathbf{A}^i$. On the other hand, if $\mathbf{A}^i = (0, 0, \ldots, 0, x)$, then it is converted to $(1, \overline{r-1}, \ldots, \overline{r-1}, \overline{r-x})$ by Conversion 1.
- *When k is recoded by Recoding 1*:
  If $\mathbf{A}^i$ is a block of nonzero digits, it is not changed from Recoding 1. Note that let $k_n = 1$ to consider $\mathbf{A}^{m-1}$. In the case $\mathbf{A}^i = (0, 0, \ldots, 0, x)$, as $\mathbf{A}_0^{i+1} \neq 0$ the recoded result of $\mathbf{A}^i$ by Recoding 1 is $(1, \overline{r-1}, \ldots, \overline{r-1}, \overline{r-x})$, where $\mathbf{A}_0^{i+1}$ denotes the first digit of $\mathbf{A}^{i+1}$.

Algorithm 2. Radix-r *Left-to-Right* recoding

Input: A scalar $k = (k_{n-1}, \cdots, k_1, k_0)_r$ with $k_i \in \Lambda_r$ and $k \bmod r \neq 0$.
Output: $k' = (k'_{l-1}, \cdots, k'_1, k'_0)_r$ with length l and $k'_i \in D_r$.
1: **let** $k_l = 1$ and $\{k_{l-1}, \ldots, k_n\}$ be all 0
2: **for** $j = l - 1$ down to 0 **do**
3: $\quad k'_j := \mathbb{RECODE}(k_{j+1}, k_j)$
4: **end for**
5: **return** k'

When k is recoded by Conversion 1 we can see that there is no carry from $\mathbf{A}^i$ to $\mathbf{A}^{i+1}$ for $0 \leq i \leq m-2$. As Recoding 1 is operated without carry, we need not consider a carry during the process of recoding k. Therefore, the representations of k' and k'' are exactly same. □

The previous 8-digit radix-3 representation $k = \sum_{j=0}^{7} k_j 3^j = (1, 0, 2, 0, 0, 1, 0, 2)_3$ can be recoded into D_3 representation from *left to right* direction by Recoding 1. Let $k_8 = 1$.

$$\begin{aligned} k = (1,0,2,0,0,1,0,2)_3 &\Rightarrow (1,*,*,*,*,*,*,*)_3 \Rightarrow (1,1,*,*,*,*,*,*)_3 \\ &\Rightarrow (1,1,\overline{1},*,*,*,*,*)_3 \Rightarrow (1,1,\overline{1},1,*,*,*,*)_3 \\ &\Rightarrow (1,1,\overline{1},1,\overline{2},*,*,*)_3 \Rightarrow (1,1,\overline{1},1,\overline{2},\overline{2},*,*)_3 \\ &\Rightarrow (1,1,\overline{1},1,\overline{2},\overline{2},1,*)_3 \Rightarrow (1,1,\overline{1},1,\overline{2},\overline{2},1,\overline{1})_3 \end{aligned}$$

By using Conversion 1 we can make a D_r representation which has a wanted length of digits, e.g., if we want l (> 8) digits representation for $(1,0,2,0,0,1,0,2)_3$ then the most simple way is $(1, \underbrace{\overline{2}, \overline{2}, \cdots, \overline{2}, \overline{2}}_{(l-8)\text{-time}}, 1, \overline{1}, 1, \overline{2}, \overline{2}, 1, \overline{1})_3$.

Similarly, Algorithm 2 generates a D_r representation which has a wanted length of digits l for a given radix-r representation of k.

Algorithm 3. Left-to-Right_Method based on Recoding 1

Input: A point P, and $k = (k_{n-1}, \cdots, k_1, k_0)_r$, with $k_i \in \Lambda_r$ and $k \bmod r \neq 0$.
Output: $Q = kP$.
1: **Pre-computation** $|a|P$ for all positive $a \in \Lambda_r$.
2: $Q \leftarrow \mathcal{O}$ and $k_n \leftarrow 1$
3: **for** $j = n - 1$ down to 0 **do**
4: $\quad Q \leftarrow rQ$
5: $\quad k'_j := \mathbb{RECODE}(k_{j+1}, k_j)$
6: $\quad$ **if** $k'_j > 0$ **then** $Q \leftarrow \mathrm{ECADD}(Q, k'_j P)$
7: $\quad$ **if** $k'_j < 0$ **then** $Q \leftarrow \mathrm{ECADD}(Q, -|k'_j|P)$
8: **end for**
9: **return** Q

Scalar Multiplication based on Recoding 1. Algorithm 3 merges the recoding stage and evaluation stag of scalar multiplication kP. The advantage of Algorithm 3 is that it reduces the memory requirement since it does not store the converted representation of k.

4.3 Extension to Higher Width: From $\Lambda_{w,r}$ to $D_{w,r}$

Algorithm 2 can be easily extended to a fixed window algorithm. Let w (≥ 2) be the window size in digits and $d = \lceil \frac{n}{w} \rceil$. The crucial observation is that as the generation $\Lambda_r \longmapsto D_r$ can be performed left-to-right, the combination of this generation and left-to-right fixed window method leads to a complete left-to-right recoding from $\Lambda_{w,r}$ to $D_{w,r}$.

Given a n-digit scalar k, encode k using Algorithm 2 and represent the result in base r^w as

$$k = \sum_{j=0}^{d-1} \mathbf{B}^j r^{wj} = (\mathbf{B}^{d-1}, ..., \mathbf{B}^1, \mathbf{B}^0)_{r^w} \ with\ \mathbf{B}^i \in D_{w,r}.$$

In more detail, $\mathbf{B}^i$ is obtained as follows; let $k_{dw} = 1$ and $\{k_{dw-1}, ..., k_n\}$ be all 0. Write $\mathbf{B}^i = (\mathbf{B}^i_{w-1}, ..., \mathbf{B}^i_1, \mathbf{B}^i_0)_r$. Then $\mathbf{B}^i$ is defined as $\mathbb{MRECODE}[k_{(i+1)w}, k_{(i+1)w-1}, ..., k_{iw+1}, k_{iw}]$, i.e., $\mathbf{B}^i_j = \mathbb{RECODE}(k_{iw+j}, k_{iw+(j+1)})$ for $0 \leq j \leq w-1$. As $\mathbf{B}^i_j \in D_r$, clearly $\mathbf{B}^i \in D_{w,r}$.

Algorithm 4. Width-w Radix-r *Left-to-Right* recoding

Input: width w, a scalar $k = (k_{n-1}, \cdots, k_1, k_0)_r$ with $k_i \in \Lambda_r$ and $k \bmod r \neq 0$.
Output: $(\mathbf{B}^{d-1}, ..., \mathbf{B}^1, \mathbf{B}^0)_{r^w}$.

1: **let** $k_{dw} = 1$ and $\{k_{dw-1}, \ldots, k_n\}$ be all 0
2: **for** $j = d - 1$ down to 0 **do**
3: $\quad \mathbf{B}^j = \mathbb{MRECODE}[k_{(j+1)w}, k_{(j+1)w-1}, ..., k_{jw+1}, k_{jw}]$
4: **end for**
5: **return** $(\mathbf{B}^{d-1}, ..., \mathbf{B}^1, \mathbf{B}^0)_{r^w}$

For example, 11-digit radix-3 representation $k = (1, 0, 0, 1, 1, 1, 0, 0, 0, 2, 1)_3$ can be recoded into $D_{2,3}$ representation from *left to right* direction by Algorithm 4.

$$k = (0, 1, \|0, 0, \|1, 1, \|1, 0, \|0, 0, \|2, 1)_3$$

$$\begin{aligned}
(\mathbb{MRECODE}[1, 0, 1] = (1, \bar{2})_3) &\Rightarrow (1, *, *, *, *, *)_{3^2}, \\
(\mathbb{MRECODE}[1, 0, 0] = (1, \bar{2})_3) &\Rightarrow (1, 1, *, *, *, *)_{3^2}, \\
(\mathbb{MRECODE}[0, 1, 1] = (\bar{2}, 1)_3) &\Rightarrow (1, 1, \bar{5}, *, *, *)_{3^2}, \\
(\mathbb{MRECODE}[1, 1, 0] = (1, 1)_3) &\Rightarrow (1, 1, \bar{5}, 4, *, *)_{3^2}, \\
(\mathbb{MRECODE}[0, 0, 0] = (\bar{2}, \bar{2})_3) &\Rightarrow (1, 1, \bar{5}, 4, \bar{8}, *)_{3^2}, \\
(\mathbb{MRECODE}[0, 2, 1] = (\bar{1}, 1)_3) &\Rightarrow (1, 1, \bar{5}, 4, \bar{8}, \bar{2})_{3^2}.
\end{aligned}$$

Due to the limitation of space, the algorithm which merges the recoding and evaluation stags of scalar multiplication kP for general width w can be found in appendix.

As we mentioned before, the scalar multiplication based on Algorithm 4 computes kP through the fixed pattern $|\underbrace{0,...,0}_{w-1},y|\underbrace{0,...,0}_{w-1},y|...|\underbrace{0,...,0}_{w-1},y|$ for $y \in D_{w,r}$. Under the Assumption 2 the attacker obtains the identical sequence $|\underbrace{R...R}_{w}A|\underbrace{R...R}_{w}A|...|\underbrace{R...R}_{w}A|$ for all the scalars, where R and A denote rQ and elliptic addition, respectively. Therefore, he/she cannot detect the secret scalar by using SPA. The complexity of it for a n-digit scalar k are approximately

- *Pre-computation* stage: $\left(\frac{(r-1)r^{w-1}-2}{2}\right)A + \left(\frac{(r-1)r^{w-1}}{2}\right)D$,
- *Evaluation* stage:
  $\left(\lceil\frac{n}{w}\rceil - 1 + w\cdot\left(\lceil\frac{n}{w}\rceil - 1\right)\cdot\left(\frac{\lceil\log_2 r\rceil}{3}\right)\right)A + \left(w\cdot\left(\lceil\frac{n}{w}\rceil - 1\right)\cdot\lceil\log_2 r\rceil\right)D$,

which is same to that of signed radix-r representation proposed by Han-Takagi [16] (see Section 3.1).

Remark. The security and the computational complexity of the proposed recoding methods are the same to that of Han-Takagi recoding method. However, as the proposed recoding method generates from the left-to-right direction we can merge recoding step and evaluation step of scalar multiplication, also we can avoid the need to record the new representation. Thus, the proposed recoding methods reduce the memory required to perform the scalar multiplication.

5 Special Case: Binary Left-to-Right Recodings

In this section we show how the ideas of the previous sections lead to simple recoding methods from Λ_2 to D_2 and its width-w version from $\Lambda_{w,2}$ to $D_{w,2}$, i.e. $r = 2$.

5.1 Left to Right Recoding from Λ_2 to D_2

Recoding 1 described in section 4.2 is naturally simplified in Table 1. We simplify the carry-free recoding rule recursively, which needs to store only **one** input bit k_{i+1} to generates i-th recoded bit k'_i in contrast to Recoding 1 which requires **two** input digits k_{i+1}, k_i. Similar to section 4.2, define

$$\mathbb{RECODE}_2(a) : \Lambda_2 \longrightarrow D_2$$

as a mapping from Λ_2 to D_2 such that $\mathbb{RECODE}_2(1) = 1$ and $\mathbb{RECODE}_2(0) = \overline{1}$. Define a multiple mapping

$$\mathbb{MRECODE}_2[a_{l-1},...,a_1,a_0] : \underbrace{\Lambda_2 \times \cdots \times \Lambda_2}_{l-tuple} \longrightarrow \underbrace{D_2 \times \cdots \times D_2}_{l-tuple}$$
$$(a_{l-1},...,a_1,a_0) \longmapsto (b_{l-1},...,b_1,b_0)$$

such that b_i, the i-th element of $\mathbb{MRECODE}_2[a_{l-1},...,a_1,a_0]$, is $\mathbb{RECODE}_2(a_i)$.

Table 1. Table of Recoding 1 from general radix-r to a special case $r = 2$

Inputs		Recoding 1	Recoding 1 when $r = 2$
k_{i+1}	k_i	Output k'_i	Output k'_i
$\neq 0$	$\neq 0$	k_i	1
$\neq 0$	0	1	1
0	$\neq 0$	$k_i - r$	-1
0	0	$1 - r$	-1

Note that in Algorithm 5, Step 3 can be paraphrased like as **if** $k_{i+1} = 0$ **then** $k'_i = -1$ **else** $k'_i = 1$.

5.2 Extension to Higher Width: From $\Lambda_{w,2}$ to $D_{w,2}$

In the following we show how the Algorithm 5 is modified into a fixed window binary version by using the mapping $\mathbb{MRECODE}_2[\cdots]$ described in the previous section. Given a n-bit scalar k, encode k using Algorithm 5 and represent the result in base 2^w as $k = \sum_{j=0}^{d-1} \mathbf{C}^j 2^{wj} = (\mathbf{C}^{d-1}, ..., \mathbf{C}^1, \mathbf{C}^0)_{2^w}$ *with* $\mathbf{C}^i \in D_{w,2}$. Algorithm 6 generates $\mathbf{C}^i$ from left-to-right. Let w (≥ 2) be the window size in bits and $d = \lceil \frac{n}{w} \rceil$.

Algorithm 5. Binary *Left-to-Right* recoding

Input: A scalar $k = (k_{n-1}, \cdots, k_1, k_0)_2$, with $k_i \in \Lambda_2$ and $k \bmod 2 \neq 0$.
Output: $k' = (k'_{l-1}, \cdots, k'_1, k'_0)$ with length l and $k'_i \in D_2$.
1: **let** $k_l = 1$ and $\{k_{l-1}, \ldots, k_n\}$ be all 0
2: **for** $j = l - 1$ down to 0 **do**
3: $\quad k'_j :=\mathbb{RECODE}_2(k_{j+1})$
4: **end for**
5: **return** k'

Algorithm 6. Width-w Binary *Left-to-Right* recoding

Input: width w, a scalar $k = (k_{n-1}, \cdots, k_1, k_0)_2$ with $k_i \in \Lambda_2$ and $k \bmod 2 \neq 0$.
Output: $(\mathbf{C}^{d-1}, ..., \mathbf{C}^1, \mathbf{C}^0)_{2^w}$.
1: **let** $k_{dw} = 1$ and $\{k_{dw-1}, \ldots, k_n\}$ be all 0
2: **for** $j = d - 1$ down to 0 **do**
3: $\quad \mathbf{C}^j = \mathbb{MRECODE}_2[k_{(j+1)w}, k_{(j+1)w-1}, ..., k_{jw+1}]$
4: **end for**
5: **return** $(\mathbf{C}^{d-1}, ..., \mathbf{C}^1, \mathbf{C}^0)_{2^w}$

6 Conclusion

We presented several recoding methods which are resistant to SPA attacks. These recodings are left-to-right so they can be interleaved with a left-to-right scalar

multiplication, removing the need to store both a scalar and its recoding. Especially, it was an unsolved problem to generate a signed radix-r representation recoded by left-to-right direction for a general width w, however, in this paper we introduced a solution of it. The proposed recoding methods are conceptually easy to understand and it is quite simple to implement them. It should be kept in mind that these recoding methods do not ensure in any way the security against differential power analysis, so countermeasures against these attacks should also be used if the secret key is used more than once.

Acknowledgements

Sung-Kyoung Kim was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2006-(C1090-0603-0025)).

References

1. ANSI X9.62, "Public Key Cryptography for the Financial Services Industry," *The Elliptic Curve Digital Signature Algorithm (ECDSA),* 333, 1999.
2. R.M. Avanzi, "A Note on the Signed Sliding Window Integer Recoding and a Left-to-Right Analogue," *SAC 2004*, LNCS 3357, pp.130-143, 2004.
3. M. Aydos, T. Yank, and C.K. Koc, "High-speed implementation of an ECC-based wireless authentication protocol on an ARM microprocessor," *IEE Proceedings Communications*, vol. 148, Issue 5, pp.273-279, Oct., 2001.
4. G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar, and T. Wollinger, "Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications," *CT-RSA 2003*, LNCS 2612, pp.158-175, 2003.
5. P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," *CRYPTO 2002*, LNCS 2442, pp.354-368, 2002.
6. P.S.L.M. Barreto, S. Galbraith, C. OhEigeartaigh and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties," Preprint 2005, to appear in Designs, Codes and Cryptography.
7. G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar, and T. Wollinger, "Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications," *CT-RSA 2003*, LNCS 2612, pp.158-175, 2003.
8. D.Boneh and M.Franklin, "Identity-Based Encryption from the weil Pairing," *Crypto 2001*, LNCS 2139, pp.213-229, 2001.
9. D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *ASIACRYPT 2001*, LNCS 2248, pp.514-532, 2001.
10. I. Blake, G. Seroussi, and N. Smart, "Elliptic Curves in Cryptogarphy," Cambridge University Press, 1999.
11. J.S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," *CHES 1999*, LNCS 1717, pp.292-302, 1999.
12. W. Clark and J. Liang, "On Arithmetic Weight for a General Radix Representation of Integers," *IEEE Transaction on IT*, IT-19, pp.823-826, 1973.

13. H. Cohen, A. Miyaji, and T. Ono, "Efficient Elliptic Curve Exponentiation Using Mixed Coordinates," *ASIACRYPT 1998*, LNCS 1514, pp.51-65, 1998.
14. I. Duursma and H -S .Lee, "Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$," *ASIACRYPT 2003*, LNCS 2894, pp.111-123, 2003.
15. S. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate pairing," *ANTS V*, LNCS 2369, pp.324-337, 2002.
16. D.-G. Han and T. Takagi, "Some Analysis of Radix-r Representations," Cryptography ePrint Archive, Report 2005/402, 2005. http://eprint.iacr.org/2005/402.
17. K. Harrison, D. Page, and N. Smart, "Software Implementation of Finite Fields of Characteristic Three," *LMS Journal of Computation and Mathematics*, Vol.5, pp.181-193, 2002.
18. K. Harrison, D. Page, and N. Smart, "Software Implementation of Finite Fields of Characteristic Three," *LMS Journal of Computation and Mathematics*, Vol.5, pp.181-193, 2002.
19. R. Harasawa, Y. Sueyoshi, and A. Kudo, "Ate pairing for $y^2 = x^5 - \alpha x$ in Characteristic five," Cryptography ePrint Archive, Report 2006/202, 2006. htt://eprint.iacr.org/2006/202.
20. K. Itoh, T. Izu and M. Takenaka, "Efficient Countermeasure against Power Analysis for Elliptic Curve Cryptosystems," *CARDIS 2004*, pp.99-114, 2004.
21. K. Itoh, T. Izu and M. Takenaka, "Improving the Randomized Initial Point Countermeasure Against DPA," *Applied Cryptography and Network Security (ACNS 2006)*, LNCS 3989, pp.459-469, 2006.
22. T. Izu and T. Takagi, "A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks," *PKC 2002*, LNCS 2274, pp.280-296, 2002.
23. A. Joux, "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems (survey)," *ANTS V*, LNCS 2369, pp.20-32, 2002.
24. M. Joye and C. Tymen, "Protections against differential analysis for elliptic curve cryptography: An algebraic approach," *CHES 2001*, LNCS 2162, pp.377-390, 2001.
25. M. Joye and S. Yen, "Optimal Left-to-Right Binary Signed-Digit Recoding," *IEEE Trans. Computers*, vol. 49, pp.740-748, July, 2000.
26. M. Joye and S. Yen, "New Minimal Modified Radix-r Representation with Applications to Smart Cards," *PKC 2002*, LNCS 2274, pp.375-384, 2002.
27. A. Joux, "A one round protocol for tripartite Diffie-Hellman," *ANTS V*, LNCS 1838, pp.385-394, 2000.
28. N. Koblitz, "Elliptic curve cryptosystems," *In Mathematics of Computation*, volume 48, pp.203-209, 1987.
29. P. Kocher,"Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *CRYPTO 1996*, LNCS 1109, pp.104-113, 1996.
30. P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis," *CRYPTO 1999*, LNCS 1666, pp.388-397, 1999.
31. S. Kwon, "Efficient Tate Pairing Computation for Elliptic Curves over Binary Fields,"*ACISP 2005*, LNCS 3574, pp.134-145, 2005.
32. K. Lauter, "The advantages of elliptic curve cryptography for wireless security," *IEEE Wireless Communications*, vol. 11, Issue 1, pp.62-67, Feb., 2004.
33. C. Lim and P. Lee, "More Flexible Exponentiation with Precomputation," *CRYPTO 1994*, LNCS 839, pp.95-107, 1994.
34. V.S. Miller, "Use of elliptic curves in cryptography," *CRYPTO 1985*, LNCS 218, pp.417-426, 1986.

35. B. Möller, "Securing Elliptic Curve Point Multiplication against Side-Channel Attacks," *ISC 2001*, LNCS 2200, pp.324-334, 2001.
36. H. Mamiya, A. Miyaji, H. Morimoto, "Efficient Countermeasures against RPA, DPA, and SPA," *CHES 2004*, LNCS 3156, pp.343-356, 2004.
37. J.A. Muir and D.R. Stinson, "New Minimal Weight Representations for Left-to-Right Window Methods," *CT-RSA 2005*, LNCS 3376, pp.366-383, 2005.
38. K. Okeya, K. Schmidt-Samoa, C. Spahn, and T. Takagi, "Signed Binary Representations Revisited," *CRYPTO 2004*, LNCS 3152, pp.123-139, 2004.
39. K. Okeya, K. Miyazaki, and K. Sakurai, "A Fast Scalar Multiplication Method with Randomized Projective Coordinates on a Montgomery-form Elliptic Curve Secure against Side Channel Attacks," *ICISC 2001*, LNCS 2288, pp.428-439, 2002.
40. K. Okeya and T. Takagi, "The Width-w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks," *CT-RSA 2003*, LNCS 2612, pp.328-342, 2003.
41. D. Page, and N. Smart, "Hardware Implementation of Finite Fields of Characteristic Three," *CHES 2002*, LNCS 2523, pp.529-539, 2002.
42. X. Ruan and R. Katti, "Left-to-Right Optimal Signed-Binary Representation of a Pair of Integers," *IEEE Trans. Computers*, vol. 54, pp.124-131, July, 2005.
43. M. Scott, "Computing the Tate Pairing," *CT-RSA 2005*, LNCS 3376, pp.293-304, 2005.
44. J.A. Solinas, "Efficient Arithmetic on Koblitz Curves," Designs, Codes and Cryptography, 19, pp.195-249, 2000.
45. M. Scott, N. Costigan, and W. Abdulwahab, "Implemetation Cryptographic Pairings on Smartcards," Cryptography ePrint Archive, Report 2006/144, 2006. http://eprint.iacr.org/2006/144.
46. N. Smart and J. Westwood, "Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three," *Applicable Algebra in Engineering, Communication and Computing*, Vol.13, No.6, pp.485-497, 2003.
47. N. Theriault, "SPA Resistant Left-to-Right Integer Recodings," *SAC 2005*, LNCS 3897, pp.345-358, 2006.
48. T. Takagi, S.-M. Yen, B.-C. Wu, "Radix-r Non-Adjacent Form," *ISC 2004*, LNCS 3225, pp.99-110, 2004.
49. X. Tian, D. Wong, and R. Zhu, "Analysis and Improvement of an Authenticated Key Exchange Protocol for Sensor Networks," *IEEE Communications letters*, vol. 9, pp. 970-972, November, 2005.
50. C. Vuillaume and K. Okeya, "Flexible Exponentiation with Resistance to Side Channel Attacks," *Applied Cyptography and Network Security*, *ACNS'06*, LNCS 3989, pp.268-283, 2006.

Appendix A

Scalar Multiplication with Width w. Algorithm 7 merges the recoding stage and evaluation stag of scalar multiplication kP for general width w.

Algorithm 7. Left-to-Right_Method based on Algorithm 6

Input: A point P, width w, a scalar $k = (k_{n-1}, \cdots, k_1, k_0)_r$ with $k_i \in \Lambda_r$ and $k \bmod r \neq 0$.
Output: $Q = kP$.

Pre-computation $|a|P$ for all positive $a \in D_{w,r}$.
let $k_{dw} = 1$ and $\{k_{dw-1}, \ldots, k_n\}$ be all 0
$Q \leftarrow \mathbb{MRECODE}[k_{dw}, k_{dw-1}, ..., k_{(d-1)w+1}, k_{(d-1)w}]$
for $j = d - 2$ down to 0 **do**
    for $i = 0$ up to $w - 1$ **do**
        $Q \leftarrow rQ$
    end for
    $k'_j :=\mathbb{MRECODE}[k_{(j+1)w}, k_{(j+1)w-1}, ..., k_{jw+1}, k_{jw}]$
    if $k'_j > 0$ **then** $Q \leftarrow \mathrm{ECADD}(Q, k'_j P)$
    if $k'_j < 0$ **then** $Q \leftarrow \mathrm{ECADD}(Q, -|k'_j|P)$
end for
return Q

Some Efficient Algorithms for the Final Exponentiation of η_T Pairing

Masaaki Shirase[1], Tsuyoshi Takagi[1], and Eiji Okamoto[2]

[1] Future University-Hakodate, Japan
[2] University of Tsukuba, Japan

Abstract. Recently Tate pairing and its variations are attracted in cryptography. Their operations consist of a main iteration loop and a final exponentiation. The final exponentiation is necessary for generating a unique value of the bilinear pairing in the extension fields. The speed of the main loop has become fast by the recent improvements, *e.g.*, the Duursma-Lee algorithm and η_T pairing. In this paper we discuss how to enhance the speed of the final exponentiation of the η_T pairing in the extension field $\mathbb{F}_{3^{6n}}$. Indeed, we propose some efficient algorithms using the torus $T_2(\mathbb{F}_{3^{3n}})$ that can efficiently compute an inversion and a powering by 3^n+1. Consequently, the total processing cost of computing the η_T pairing can be reduced by 16% for $n = 97$.

Keywords: Tate pairing, η_T pairing, final exponentiation, torus.

1 Introduction

Bilinear pairings deliver us new cryptographic applications such as identity-based encryptions [5], short signatures [7], and efficient broadcast encryptions [6]. Recently Duursma and Lee [8] proposed an efficient algorithm for computing Tate pairing. The Duursma-Lee algorithm uses the supersingular curves,

$$E^b(\mathbb{F}_{3^n}) : y^2 = x^3 - x + b \text{ with } b \in \{-1, 1\}. \tag{1}$$

Kwon proposed an efficient variation of the Duursma-Lee algorithm that requires no cube root operation [12]. Barreto *et. al.* proposed the η_T pairing [3], which reduces the number of the main loop in the Duursma-Lee algorithm to half. Beuchat *et. al.* presented a faster variation of η_T pairing without a cube root operation [4]. Currently the η_T pairing is one of the fastest algorithms for computing the bilinear pairing.

Both the Duursma-Lee algorithm and the η_T pairing require the "final exponentiation", *i.e.*, A^s for $A \in \mathbb{F}_{3^{6n}}$ and some integer s, since the resulting element by the pairing algorithms is contained in the quotient group $\mathbb{F}^*_{3^{6n}}/\mathbb{F}^*_{3^{3n}}$. The final exponentiations for the Duursma-Lee algorithm and the η_T pairing are $A^{3^{3n}-1}$ and A^W with $W = (3^{3n}-1)(3^n+1)(3^n+1-b3^{(n+1)/2})$, respectively. The η_T pairing without the final exponentiation is about twice faster than the Duursma-Lee algorithm, but the final exponentiation in the η_T pairing causes

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 254–268, 2007.

a relatively large overhead. For example, Shu *et. al.* [14] estimated that the η_T pairing with the final exponentiation is as fast as the Duursma-Lee algorithm in hardware. Ronan *et. al.* reported that the straightforward implementation of the final exponentiation is more than 35% of the whole algorithm [13]. In Section 4, we estimated that the currently fastest final exponentiation [3] is about 25% of the whole algorithm.

In this paper we try to reduce the cost of the final exponentiation of the η_T pairing. Barreto *et. al.* proposed an efficient calculation for the final exponentiation using Frobenius mapping [1]. We propose that we use not Frobenius but also Torus T_2 for it. Note that $A^{3^{3n}-1}$ is an element in the torus $T_2(\mathbb{F}_{3^{3n}})$, which is a subgroup of $\mathbb{F}^*_{3^{6n}}$. We show that an inversion and a powering by (3^n+1)-th in $T_2(\mathbb{F}_{3^{3n}})$ are efficiently computed for the basis $\{1, \sigma\}$ of $\mathbb{F}_{3^{6n}}$ over $\mathbb{F}_{3^{3n}}$ with $\sigma^2 + 1 = 0$. We then present an efficient algorithm for the final exponentiation $A^W = B^{(3^n+1)(3^n+1-b3^{(n+1)/2})}$ with $B = A^{3^{3n}-1}$ of the η_T pairing in the torus $T_2(\mathbb{F}_{3^{3n}})$, which can be computed with 36 multiplications in $\mathbb{F}_{3^n}$ plus other negligible operations. Consequently, the final exponentiation of our proposed scheme requires only about 13% of the whole η_T pairing, which achieves about 16% faster η_T pairing than the previous known algorithms.

On the other hand, Granger *et. al.* presented an encoding method of $\mathbb{F}^*_{3^{6n}}/\mathbb{F}^*_{3^{3n}}$ [10], which eliminates the final exponentiation from the Duursma-Lee algorithm. We call it the GPS encoding according to the authors' name. In this paper, we discuss how to apply the GPS encoding to the η_T pairing. The η_T pairing with the GPS encoding can be faster depending on the information of b.

The remainder of this paper is organized as follows: In Section 2 we explain Tate pairing and the η_T pairing. In Section 3 we describe several representations (including the Torus $T_2(\mathbb{F}_{3^{3n}})$ and the GPS encoding) of group $\mathbb{F}^*_{3^{6n}}$ and apply them to the efficient computation of the final exponentiation for the Duursma-Lee algorithm. In Section 4 we propose new efficient algorithms of computing the final exponentiation for the η_T pairing and how to apply the GPS encoding to the η_T pairing. In Section 5 we conclude this paper.

2 Tate Pairing and η_T Pairing

In this section we explain about Tate pairing and its efficient variations, namely the Duursma-Lee algorithm and the η_T pairing.

2.1 Tate Pairing

Let $\mathbb{F}_q$ be a finite field with q elements, where q be a power of the characteristic p. Let E be elliptic curves defined over $\mathbb{F}_q$, and let $\mathcal{O}_E$ be the point at infinity. Let l be a positive integer relatively prime to q with $l \,|\, \#E(\mathbb{F}_q)$, and let k be the minimal positive integer with $l \,|\, (q^k - 1)$. This k is called the embedded degree. Then Tate pairing is a map

$$\langle \cdot, \cdot \rangle_l : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_{q^k})/lE(\mathbb{F}_{q^k}) \to \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^l,$$

which satisfies the bilinearity $\langle P, aQ\rangle_l = \langle aP, Q\rangle_l = \langle P, Q\rangle_l^a$ for any integer $a \neq 0$, and is non-degenerate, *i.e.*, there exists a $Q \in E(\mathbb{F}_{q^k})$ such that $\langle P, Q\rangle_l \notin (\mathbb{F}_{q^k}^*)^l$ for $P \in E(\mathbb{F}_{q^k})[l] \setminus \{\mathcal{O}_E\}$.

It is typically selected that l and q^k are about 160 bits and 1024 bits, respectively. One of the most efficient classes for computing the bilinear map is constructed over supersingular elliptic curves. The embedded degree k of supersingular elliptic curves is one of 4, 6, or 2 for characteristic 2, 3 or $p > 3$, respectively. This paper deals with the case of characteristic 3 and uses elliptic curves formed by (1). It is known that $\#E^b(\mathbb{F}_{3^n}) = 3^n + 1 + b'3^{(n+1)/2}$, where b' is defined as

$$b' = \begin{cases} b & \text{if } n \equiv 1, 11 \pmod{12}, \\ -b & \text{if } n \equiv 5, 7 \pmod{12}. \end{cases}$$

Note that we have $n \equiv 1, 5, 7, 11 \pmod{12}$ since n has to be coprime to 6 [8].

We require an injection ψ from $E(\mathbb{F}_{3^n})[l]$ to $E(\mathbb{F}_{3^{6n}})/lE(\mathbb{F}_{3^{6n}})$ since $\langle P, Q\rangle_l$ is defined for points $P \in E(\mathbb{F}_{3^n})[l]$ and $Q \in E(\mathbb{F}_{3^{6n}})/lE(\mathbb{F}_{3^{6n}})$. This ψ is sometimes called as the distortion map. In the case of characteristic three, the distortion map is defined as $\psi(x, y) = (-x + \rho, y\ \sigma)$ for $(x, y) \in E(\mathbb{F}_{3^n})$, where σ and ρ satisfy

$$\sigma^2 = -1 \text{ and } \rho^3 = \rho + b.$$

We usually select the basis $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ of $\mathbb{F}_{3^{6n}}$ over $\mathbb{F}_{3^n}$, where ρ and σ are utilized in the distortion map. Every element A in $\mathbb{F}_{3^{6n}}$ is then represented as $A = a_0 + a_1\sigma + a_2\rho + a_3\sigma\rho + a_4\rho^2 + a_5\sigma\rho^2$ for some $a_i \in \mathbb{F}_{3^n}$. Moreover an element A_0 in $\mathbb{F}_{3^{3n}}$ is represented as $A_0 = a_0 + a_2\rho + a_4\rho^2$. We denote by M_k, C_k and I_k the computational cost of multiplication, cubing, and inversion in $\mathbb{F}_{3^{kn}}$, respectively. Then the following relationships

$$M_6 = 3M_3,\ M_3 = 6M_1,\ C_6 = 2C_3,\ C_3 = 3C_1,\ I_6 = 5M_3 + I_3,\ I_3 = 8M_1 + I_1 \tag{2}$$

are held [11]. The computational costs appeared in this paper are estimated using Eq. (2). The computational cost is estimated without considering the costs of addition and subtraction which are usually negligible. Beuchat *et. al.* pointed out A^{3^n} in $\mathbb{F}_{3^{6n}}$ can be computed virtually for free [4] (see Appendix B). Therefore we have

$$\text{the cost of } A^{3^n+1} (= A^{3^n} \cdot A) \text{ is } M_6 = 18M_1. \tag{3}$$

The resulting value $\langle P, \psi(Q)\rangle_l$ of Tate pairing is contained in the quotient group $\mathbb{F}_{3^{6n}}^*/(\mathbb{F}_{3^{6n}}^*)^l$. Then there are many choices for representing elements in a coset of the quotient group. Indeed $A, B \in \mathbb{F}_{3^{6n}}^*$ are contained in the same coset, if they satisfies $B = A \cdot C^l$ for some $C \in \mathbb{F}_{3^{6n}}^*$. We are able to eliminate this *ambiguity* by using the final exponentiation. The final exponentiation tries to compute the $((3^{6n} - 1)/l)$-th powering to the output from the Tate pairing. Therefore we also deploy the modified Tate pairing $\hat{e}(P, Q)$ defined by

$$\hat{e} : E(\mathbb{F}_{3^n})[l] \times E(\mathbb{F}_{3^n})[l] \to \mathbb{F}_{3^{6n}}^*,\ (P, Q) \mapsto \hat{e}(P, Q) = \langle P, \psi(Q)\rangle_l^{(3^{6n}-1)/l},$$

whose value in $\mathbb{F}_{3^{6n}}^*$ can be uniquely determined.

Granger *et. al.* proposed another technique to remove the ambiguity [10]. In this paper we denote by *GPS encoding* the technique proposed by Granger *et. al.* according to the authors' name (refer Sections 3.2).

2.2 Efficient Pairings on Supersingular Curves over $\mathbb{F}_{3^n}$

We explain about some efficient algorithms for computing the bilinear pairing over supersingular curves with characteristic three. Algorithm 1 is the Duursma-Lee algorithm which outputs $\langle P, \psi(Q)\rangle_{3^{3n}+1}$ for $P, Q \in E^b(\mathbb{F}_{3^n})$ [12]. The Duursma-Lee algorithm has n interactions in the main loop and the whole computational cost is $15nM_1 + (10n+2)C_1$. Note that the final exponentiation of the Duursma-Lee algorithm uses the powering to $(3^{6n}-1)/(3^{3n}+1) = (3^{3n}-1)$.

Algorithm 1. Duursma-Lee Algorithm [12]

input: $P = (x_p, y_p),\ Q = (x_q, y_q) \in E^b(\mathbb{F}_{3^n})[l]$
output: $\langle P, \psi(Q)\rangle_{3^{3n}+1} \in \mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^{3^{3n}+1}$

$R_0 \leftarrow 1$ (in $\mathbb{F}_{3^{6n}}$), $x_q \leftarrow x_q^3,\ y_q \leftarrow y_q^3$ (in $\mathbb{F}_{3^n}$), $d \leftarrow (bn \bmod 3)$
for $i \leftarrow 0$ **to** $n-1$ **do**
  $x_p \leftarrow x_p^9,\ y_p \leftarrow y_p^9$ (in $\mathbb{F}_{3^n}$)
  $r_0 \leftarrow x_p + x_q + d$ (in $\mathbb{F}_{3^n}$)
  $R_1 \leftarrow -r_0^2 - y_p y_q \sigma - r_0\rho - \rho^2$ (in $\mathbb{F}_{3^{6n}}$)
  $R_0 \leftarrow R_0^3$ (in $\mathbb{F}_{3^{6n}}$)
  $R_0 \leftarrow R_0 R_1$ (in $\mathbb{F}_{3^{6n}}$)
  $y_q \leftarrow -y_q$ (in $\mathbb{F}_{3^n}$)
  $d \leftarrow ((d-b) \bmod 3)$
end for
return R_0 (Cost: $15nM_1 + (10n+2)C_1$)

Algorithm 2. Computation of $\eta_T(P,Q)^{3^{(n+1)/2}}$ for $n \equiv 1 \pmod{12}$ [4]

input: $P = (x_p, y_p), Q = (x_q, y_q) \in E^b(\mathbb{F}_{3^n})[l]$
output: $\eta_T(P,Q)^{3^{(n+1)/2}} \in \mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^{3^n+1+b'3^{(n+1)/2}}$

if $b = 1$ **then** $y_p \leftarrow -y_p$
$d \leftarrow b$ (in $\mathbb{F}_3$), $R_0 \leftarrow -y_p(x_p + x_q + b) + y_q\sigma + y_p\rho$ (in $\mathbb{F}_{3^{6n}}$)
for $i \leftarrow 0$ **to** $(n-1)/2$ **do**
  $r_0 \leftarrow x_p + x_q + d$ (in $\mathbb{F}_{3^n}$)
  $R_1 \leftarrow -r_0^2 + y_p y_q \sigma - r_0\rho - \rho^2$ (in $\mathbb{F}_{3^{6n}}$)
  $R_0 \leftarrow R_0 R_1$ (in $\mathbb{F}_{3^{6n}}$)
  $y_p \leftarrow -y_p$ (in $\mathbb{F}_{3^n}$)
  $x_q \leftarrow x_q^9,\ y_q \leftarrow y_q^9$ (in $\mathbb{F}_{3^n}$)
  $R_0 \leftarrow R_0^3$ (in $\mathbb{F}_{3^{6n}}$)
  $d \leftarrow ((d-b) \bmod 3)$
end for
return R_0 (Cost: $(7.5n+8.5)M_1 + (5n+5)C_1$)

Next Barreto *et. al.* introduced the η_T pairing [3]. The η_T pairing is also defined on supersingular elliptic curves formed by (1) for $n \equiv 1, 5 \pmod 6$ in the case of characteristic three. Beuchat *et. al.* proposed a variation of the η_T pairing (Algorithm 2), which requires no cube root calculation and outputs $\eta_T(P,Q)^{3^{(n+1)/2}}$ in the case of $n \equiv 1 \pmod{12}$ [4]. The number of iterations in the main loop of the η_T pairing becomes $(n+1)/2$, which is half for the Duursma-Lee algorithm. The computational cost of Algorithm 2 is $(7.5n + 8.5)M_1 + (5n + 5)C_1$.

Note that the η_T pairing itself does not satisfy the bilinearity. Therefore we have to compute the final exponentiation with W-th powering with

$$W = (3^{3n} - 1)(3^n + 1)(3^n + 1 - b'3^{(n+1)/2}) = (3^{6n} - 1)/\#E^b(\mathbb{F}_{3^n}). \quad (4)$$

This powering function by W is the final exponentiation in the η_T pairing.

We note that $(\eta_T(P,Q)^{3^{(n+1)/2}})^W$ is a bilinear and non-degenerate pairing as well as the modified Tate pairing or the η_T pairing with final exponentiation, where W is given by Eq. (4). Then we can use $(\eta_T(P,Q)^{3^{(n+1)/2}})^W$ in almost cryptographic protocols which require a pairing without conversion to Tate pairing or $\eta_T(P,Q)^W$.

If necessary, we can calculate the modified Tate pairing from $(\eta_T(P,Q)^{3^{(n+1)/2}})^W$. First $\eta_T(P,Q)^W$ is obtained due to powering by $-3^{(n+1)/2}$. Next we use the following relationship between the modified Tate pairing and the η_T pairing,

$$(\eta_T(P,Q)^W)^{3T^2} = \hat{e}(P,Q)^Z,$$

where $T = -b'3^{(n+1)/2} - 1, Z = -b'3^{(n+3)/2}$ [3].

3 Efficient Final Exponentiation and GPS Encoding

In this section we present the final exponentiation of Tate pairing and the GPS encoding that requires no final exponentiation.

3.1 Efficient Final Exponentiation for Duursma-Lee Algorithm

We recall how to efficiently compute the final exponentiation of the Duursma-Lee algorithm, namely $A^{3^{3n}-1}$ for $A \in \mathbb{F}_{3^{6n}}$ [11].

The base of $\mathbb{F}_{3^{6n}}$ over $\mathbb{F}_{3^n}$ is fixed with $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ as we discussed in Section 2. Let A_0 and A_1 be elements in $\mathbb{F}_{3^{3n}}$ with $A_0 = a_0 + a_2\rho + a_4\rho^2$ and $A_1 = a_1 + a_3\rho + a_5\rho^2$. Then every element $A \in \mathbb{F}_{3^{6n}}$ is represented as

$$A = A_0 + A_1\sigma = a_0 + a_1\sigma + a_2\rho + a_3\sigma\rho + a_4\rho^2 + a_5\sigma\rho^2.$$

This means that $\mathbb{F}_{3^{6n}}$ is a quadratic extension from $\mathbb{F}_{3^{3n}}$ with the basis $\{1, \sigma\}$. It is easily to know that $\sigma^{3^{3n}} = -\sigma$ for $n \equiv 1, 5 \pmod 6$ which is a necessary condition for the Duursma-Lee algorithm and the η_T pairing algorithms. We then have the relationship

$$A^{3^{3n}} = (A_0 + A_1\sigma)^{3^{3n}} = A_0^{3^{3n}} + A_1^{3^{3n}}\sigma^{3^{3n}} = A_0 - A_1\sigma$$

for $A = A_0 + A_1\sigma \in \mathbb{F}_{3^{6n}}^*$. Therefore, the final exponentiation for the Duursma-Lee algorithm is performed as follows:

$$A^{3^{3n}-1} = \frac{A^{3^{3n}}}{A} = \frac{A_0 - A_1\sigma}{A_0 + A_1\sigma}\ .$$

Moreover $(A_0 + A_1\sigma) \cdot (A_0 - A_1\sigma) = A_0^2 + A_1^2 \in \mathbb{F}_{3^{3n}}^*$ yields the equation

$$A^{3^{3n}-1} = \frac{(A_0 - A_1\sigma)^2}{A_0^2 + A_1^2} = \frac{(A_0^2 - A_1^2) - 2A_0A_1\sigma}{A_0^2 + A_1^2}. \tag{5}$$

Then the computational cost of the final exponentiation for the Duursma-Lee algorithm is

$$5M_3 + I_3 = 30M_1 + I_3. \tag{6}$$

3.2 GPS Encoding in $\mathbb{F}_{3^{6n}}^*/\mathbb{F}_{3^{3n}}^*$

The GPS encoding is another technique of removing the ambiguity of representation from the cosets in a quotient group $\mathbb{F}_{3^{6n}}^*/(\mathbb{F}_{3^{6n}}^*)^l$ [10].

Denote by $\mathcal{G}$ be a quotient group resulting from the Duursma-Lee algorithm, namely $\mathcal{G} = \mathbb{F}_{3^{6n}}^*/(\mathbb{F}_{3^{6n}}^*)^{3^{3n}+1}$. This group $\mathcal{G}$ has a group law which is isomorphic to a subgroup of $\mathbb{F}_{3^{6n}}^*$. We then have the relationship $\mathcal{G} = \mathbb{F}_{3^{6n}}^*/\mathbb{F}_{3^{3n}}^*$ due to $\mathbb{F}_{3^{3n}}^* = (\mathbb{F}_{3^{6n}}^*)^{3^{3n}+1}$. In other words, both $A_0 + A_1\sigma$ and $(\lambda A_0) + (\lambda A_1)\sigma$ are contained in the same coset for any $\lambda \in \mathbb{F}_{3^{3n}}^*$. Especially $A_0 + A_1\sigma$ is equivalent to $A_0/A_1 + \sigma$ in $\mathcal{G}$ in the case of $A_1 \neq 0$. Therefore the map

$$\tau\ :\ \mathcal{G} \to \mathbb{F}_{3^{3n}} \cup \{\mathcal{O}\},\quad A_0 + A_1\sigma \mapsto \begin{cases} A_0/A_1 & \text{if } A_1 \neq 0 \\ \mathcal{O} & \text{if } A_1 = 0 \end{cases}$$

is a bijection and gives a representation for $\mathcal{G}$ without ambiguity, where $\mathcal{O}$ is the point at infinity. This representation for $\mathcal{G}$ is called the GPS encoding in this paper. The computational cost for computing the GPS encoding for a given $A \in \mathbb{F}_{3^{6n}}^*$ is

$$M_3 + I_3 = 6M_1 + I_3,$$

because the map τ is performed by one division in $\mathbb{F}_{3^{3n}}$ (= one inversion and one multiplication).

Table 1 gives a comparison of the final exponentiation with the GPS encoding for the Duursma-Lee algorithm.

Table 1. Final exponentiation and GPS encoding for the Duursma-Lee algorithm

	Output of Duursma-Lee algorithm	GPS encoding
Group	$\mathcal{G} = \mathbb{F}_{3^{6n}}^*/\mathbb{F}_{3^{3n}}^*$	$\mathbb{F}_{3^n}^* \cup \{\mathcal{O}\}$
Element	$A_0 + A_1\sigma,\ A_0, A_1 \in \mathbb{F}_{3^{3n}}$	A_0/A_1 (Cost: $6M_1 + I_3$)
Final exponentiation	$\frac{A_0 - A_1\sigma}{A_0 + A_1\sigma}$ (Cost: $30M_1 + I_3$)	–

GPS encoding requires no final exponentiation.

4 The Proposed Algorithm

In this section we present a new efficient final exponentiation and the GPS encoding for the η_T pairing.

4.1 Torus $T_2(\mathbb{F}_{3^{3n}})$

Granger *et. al.* introduced the torus $T_2(\mathbb{F}_{3^{3n}})$ for compressing the value of $\mathbb{F}_{3^{6n}}$ [10]. At first we describe the arithmetic of the torus.

Let L be an m-th extension field of a field k. Let $N_{L/F}$ be a norm map to field F with $k \subset F \subsetneq L$. The torus $T_m(k)$ is a subgroup of L^* defined by $T_m(k) = \cap_{k \subset F \subsetneq L} Ker[N_{L/F}]$. In the paper we especially deal with the $T_2(k) = Ker[N_{L/k}]$ in the case of $m = 2$, $k = \mathbb{F}_{3^{3n}}$, and $L = \mathbb{F}_{3^{6n}}$. Every element in $\mathbb{F}^*_{3^{6n}}$ is represented as $A = A_0 + A_1\sigma$ with $A_0, A_1 \in \mathbb{F}_{3^{3n}}$. The conjugate element of $A = A_0 + A_1\sigma$ in $\mathbb{F}^*_{3^{6n}}$ is $\bar{A} = A_0 - A_1\sigma$, and thus $N_{\mathbb{F}_{3^{6n}}/\mathbb{F}_{3^{3n}}}(A) = A\bar{A} = A_0^2 + A_1^2$. Therefore $T_2(\mathbb{F}_{3^{3n}})$ can be represented by

$$T_2(\mathbb{F}_{3^{3n}}) = \{A_0 + A_1\sigma \in \mathbb{F}^*_{3^{6n}} \ : \ A_0^2 + A_1^2 = 1\}.$$

The element $A_0 + A_1\sigma \in \mathbb{F}_{3^{6n}}$ can be compressed to the half using the relationship $A_0^2 + A_1^2 = 1$ (Refer [10] for the further results about the compression of the pairing value).

4.2 The Proposed Final Exponentiation

We point out that some operations in the torus $T_2(\mathbb{F}_{3^{3n}})$ can be computed efficiently. We then present a new efficient final exponentiation algorithm for the η_T pairing.

At first we can easily prove the following lemma.

Lemma 1. *The torus $T_2(\mathbb{F}_{3^{3n}})$ has following properties.*
(i) $A_0 - A_1\sigma = (A_0 + A_1\sigma)^{-1}$ for $A_0 + A_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$.
(ii) $(A_0 + A_1\sigma)^{3^{3n}-1} \in T_2(\mathbb{F}_{3^{3n}})$ for $A_0 + A_1\sigma \in \mathbb{F}^*_{3^{6n}}$.

Proof. (i) $A_0 - A_1\sigma$ is the inverse of $A_0 + A_1\sigma$ due to $(A_0 + A_1\sigma)(A_0 - A_1\sigma) = A_0^2 + A_1^2 = 1$ for $A_0 + A_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$.
(ii) The summation of a squaring of the constant term and that of the coefficient of Eq. (5) is equal to $\dfrac{(A_0^2 - A_1^2)^2 + (2A_0A_1)^2}{(A_0^2 + A_1^2)^2} = 1$, and thus we obtain $(A_0 + A_1)^{3^{3n}-1} \in T_2(\mathbb{F}_{3^{3n}})$. □

Therefore, the computational cost of the inversion in the torus $T_2(\mathbb{F}_{3^{3n}})$ is virtually for free.

Next let $A \in \mathbb{F}_{3^{6n}}$ be an output value from the η_T pairing. Note that $B = A^{3^{3n}-1}$ is contained in the torus $T_2(\mathbb{F}_{3^{3n}})$ due to Lemma 1. Then the final exponentiation A^W with $W = (3^{3n} - 1)(3^n + 1)(3^n + 1 - b'3^{(n+1)/2})$, can be computed as follows:

$$A^W = \begin{cases} D \cdot E^{-1} & \text{if } b' = 1 \\ D \cdot E & \text{if } b' = -1, \end{cases}$$

where $D = C^{3^n+1}$ and $E = C^{3^{(n+1)/2}}$ with $C = B^{3^n+1}$. It is easily to see that C, D and $E \in T_2(\mathbb{F}_{3^{3n}})$ since $T_2(\mathbb{F}_{3^{3n}})$ is a subgroup of $\mathbb{F}^*_{3^{6n}}$. The computation of $C^{3^{(n+1)/2}}$ can be efficiently performed by repeatedly calling the cubing algorithm in $\mathbb{F}_{3^n}$. On other hand we have the following lemma for the computation of X^{3^n+1} with $X \in T_2(\mathbb{F}_{3^{3n}})$ that requires no cubing.

Lemma 2. *Let $n \equiv 1, 5 \pmod 6$. For $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$ we can compute $Y = \Lambda(X) = X^{3^n+1} = Y_0 + Y_1\sigma$ with 9 multiplications in $\mathbb{F}_{3^n}$ as follows:*

Let $z_0 \sim z_8$ be defined as

$$\begin{array}{lll} z_0 = x_0x_4, & z_1 = x_1x_5, & z_2 = x_2x_4, \\ z_3 = x_3x_5, & z_4 = (x_0 + x_1)(x_4 - x_5), & z_5 = x_1x_2, \\ z_6 = x_0x_3, & z_7 = (x_0 + x_1)(x_2 + x_3), & z_8 = (x_2 + x_3)(x_4 - x_5), \end{array}$$

then, Y can be computed as following table, where $X_0 = x_0 + x_2\rho + x_4\rho^2, X_1 = x_1 + x_3\rho + x_5\rho^2$ and $Y_0 = y_0 + y_2\rho + y_4\rho^2, Y_1 = y_1 + y_3\rho + y_5\rho^2$ $(x_i, y_i \in \mathbb{F}_{3^n})$ for $i = 0, 1, ..., 5$.

Case of $n \equiv 1 \pmod 6$	*Case of $n \equiv 5 \pmod 6$*
$y_0 = 1 + z_0 + z_1 - bz_2 - bz_3$	$y_0 = 1 + z_0 + z_1 + bz_2 + bz_3$
$y_1 = z_1 + z_4 + bz_5 - z_0 - bz_6$	$y_1 = z_1 + z_4 - bz_5 - z_0 + bz_6$
$y_2 = z_7 - z_2 - z_3 - z_5 - z_6$	$y_2 = z_5 + z_6 - z_7$
$y_3 = bz_0 + z_3 + z_8 - z_2 - bz_1 - bz_4$	$y_3 = -bz_0 + z_3 + z_8 - z_2 + bz_1 + bz_4$
$y_4 = bz_2 + bz_3 + bz_7 - bz_5 - bz_6$	$y_4 = bz_2 + bz_3 + bz_7 - bz_5 - bz_6$
$y_5 = bz_3 + bz_8 - bz_2$	$y_5 = -bz_3 - bz_8 + bz_2$

Proof. Refer Appendix A. □

From Lemma 2 the proposed algorithm can be obtained. We describe the explicit algorithm of the proposed scheme in Algorithm 3. Note that although a computation of X^{3^n+1} takes $9M_1$ only for $X \in T_2(\mathbb{F}_{3^n})$ due to this lemma, X^{3^n+1} takes $18M_1$ for arbitrary $X \in \mathbb{F}^*_{3^{6n}}$ due to Eq. (3).

Proposition 1. *Algorithm 3 requires $66M_1 + (3n+3)C_1 + I_3$, where M_1, C_1, I_3 are the cost of multiplication in $\mathbb{F}_{3^n}$, cubing in $\mathbb{F}_{3^n}$, inversion in $\mathbb{F}_{3^{3n}}$, respectively.*

Proof. The computation of $B = A^{3^{3n}-1}$ is as expensive as that of the final exponentiation for the Duursma-Lee algorithm, namely $30M_1 + I_3$ from Eq. (6). The calculations of C and D are performed by a powering to the $(3^n + 1)$-th power. The calculation of E is performed by $(n + 1)/2$ cubings (its cost is $(n+1)/2 \cdot C_6 = (3n+3)C_1$). We have to calculate E^{-1} in the case of $b' = 1$, which requires no cost due to Lemma 1. Hence the proposed algorithm of computing the final exponentiation for the η_T pairing needs $(30M_1+I_3)+(3n+3)C_1+2C_T+M_6$, where $C_T = 9M_1$ is the cost of powering to $(3^n + 1)$-th in $T_2(\mathbb{F}_{3^{3n}})$. We thus obtain the cost estimation of this proposition. □

Algorithm 3. Proposed Final Exponentiation of η_T Pairing

input: $A = (a_0, a_1, a_2, a_3, a_4, a_5) \in \mathbb{F}_{3^{6n}}^*$, $b' \in \{-1, 1\}$
output: $A^W \in \mathbb{F}_{3^{6n}}^*$ for $W = (3^{3n}-1)(3^n+1)(3^n+1-b'3^{(n+1)/2})$
1: $B \leftarrow A^{3^{3n}-1}$ (in $\mathbb{F}_{3^{6n}}$) (Eq.(5))
2: $C \leftarrow B^{3^n+1} = \Lambda(B)$ (in $T_2(\mathbb{F}_{3^{3n}})$) (Lemma 2)
3: $D \leftarrow C^{3^n+1} = \Lambda(C)$ (in $T_2(\mathbb{F}_{3^{3n}})$) (Lemma 2)
4: $E \leftarrow C$
5: **for** $i \leftarrow 0$ **to** $(n-1)/2$ **do**
6: $E \leftarrow E^3$ in $\mathbb{F}_{3^{6n}}$ (in $\mathbb{F}_{3^{6n}}$)
7: **end for**
8 **if** $(b'=1)$ **then return** $D \cdot \overline{E}$ (in $\mathbb{F}_{3^{6n}}$) (Cost: $66M_1 + (3n+3)C_1 + I_3$)
9: **else return** $D \cdot E$ (in $\mathbb{F}_{3^{6n}}$) (Cost: $66M_1 + (3n+3)C_1 + I_3$)

4.3 How to Apply GPS Encoding to η_T Pairing

In this section we explain how to apply the GPS encoding to the η_T pairing.

The GPS encoding utilizes the arithmetic of the quotient group $\mathcal{G} = \mathbb{F}_{3^{6n}}^*/\mathbb{F}_{3^{3n}}^*$. Note that $\eta_T(P,Q)^V$ for $V = (3^n+1)(3^n+1-b'(3^{(n+1)/2}))$ is contained in $\mathcal{G}$. In order to compute the powering by V we have to compute in $\mathbb{F}_{3^{6n}}$ since $\eta_T(P,Q)$ is contained neither in $\mathcal{G}$ nor $T_2(\mathbb{F}_{3^{3n}})$. We have the relationship $\eta_T(P,Q)^V = CD^{-b'}$, where $C = B^{3^n+1}, D = B^{3^{(n+1)/2}}$, and $B = \eta_T(P,Q)^{3^n+1}$. Algorithm 4 shows the GPS encoding for the η_T pairing.

Algorithm 4. Proposed GPS Encoding of η_T Pairing

input: $A \in \mathbb{F}_{3^{6n}}^*/(\mathbb{F}_{3^{6n}}^*)^{3^n+1-b'3^{(n+1)/2}}$
output: GPS encoding of $A \in \mathbb{F}_{3^{6n}}^*$
1. $B \leftarrow A^{3^n+1}$ (in $\mathbb{F}_{3^{6n}}$)
2. $C \leftarrow B^{3^n+1}$ (in $\mathbb{F}_{3^{6n}}$)
3. $D \leftarrow B^{3^{(n+1)/2}}$ (in $\mathbb{F}_{3^{6n}}$)
4. **if** $b'=1$ **then** $E \leftarrow C \cdot D^{-1}$ (in $\mathbb{F}_{3^{6n}}$)
else $E \leftarrow C \cdot D$ (in $\mathbb{F}_{3^{6n}}$)
5. **return** E_0/E_1, where $E = E_0 + E_1\sigma$ (in $\mathbb{F}_{3^{3n}}$)

$$\left(\begin{array}{ll} \text{cost: } 90M_1 + (3n+3)C_1 + 2I_3 & \text{if } b'=1 \\ 60M_1 + (3n+3)C_1 + I_3 & \text{if } b'=-1 \end{array}\right)$$

We estimate the computational cost of Algorithm 4. Recall that $X^{3^n} \in \mathbb{F}_{3^{6n}}$ is computed virtually for free (see [4] or Appendix B). Therefore the cost of computing $X^{3^n+1} = X^{3^n} \cdot X$ is just $M_6 = 18M_1$. The total costs of both Step 1 and 2 are $36M_1$. The cost of $B^{3^{(n+1)/2}}$ is $((n+1)/2) \cdot C_6 = (3n+3)C_1$. The cost of $C \cdot D^{-1}$ is $M_6 + I_6 = 48M_1 + I_3$ and the cost of $C \cdot D$ is $M_6 = 18M_1$. The computation of E_0/E_1 which is same as the original GPS encoding takes $6M_1 + I_3$. Consequently the GPS encoding for the η_T pairing is $90M_1 + (3n+3)C_1 + 2I_1$ if $b'=1$ and $60M_1 + (3n+3)C_1 + I_1$ if $b'=-1$.

4.4 Comparison

Here we compare the computational cost of the proposed scheme with other schemes.

The computational cost of an exponentiation with cubings and multiplications by bit is $2nM_6/3+(n-1)C_6 = 12nM_1+6(n-1)C_1$ on average. The previously fastest method using Frobenius and no Torus proposed by [3] for computing the final exponentiation requires $10M_6+(n+3)C_6/2+I_6 = 210M_1+(3n+9)C_1+I_3$. We choose the relationship among C_1, M_1, I_3 for some n's as Table 2 [10].

Table 2. Relationship among C_1, M_1 and I_3

n	97	163	193	239	353
C_1	$0.1395M_1$	$0.0750M_1$	$0.0707M_1$	$0.0639M_1$	$0.0411M_1$
I_3	$15.73M_1$	$15.97M_1$	$15.47M_1$	$18.05M_1$	$17.21M_1$

Table 3. Comparison of several bilinear pairing algorithms

η_T pairing ($\eta_T(P,Q)^{3^{(n+1)/2}}$)		computational cost
Proposed final exponentiation		$\mathbf{66M_1 + (3n+3)C_1 + I_3}$
(Algorithm 3)	total cost	$(7.5n+74.5)M_1 + (8n+8)C_1 + I_3$
Proposed GPS encoding ($b'=1$)		$\mathbf{90M_1 + (3n+3)C_1 + 2I_3}$
(Algorithm 4)	total cost	$(7.5n+98.5)M_1 + (8n+8)C_1 + 2I_3$
Proposed GPS encoding ($b'=-1$)		$\mathbf{60M_1 + (3n+3)C_1 + I_3}$
(Algorithm 4)	total cost	$(7.5n+68.5)M_1 + (8n+8)C_1 + I_3$
Ordinary final exponentiation		$228M_1 + (3n+3)C_1 + I_3$
([3])	total cost	$(7.5n+236.5)M_1 + (8n+8)C_1 + I_3$

Duursma-Lee algorithm		computational cost
Final exponentiation		$30M_1 + I_3$
([11])	total cost	$(15n+30)M_1 + (10n+2)C_1 + I_3$
GPS encoding		$6M_1 + I_3$
([10])	total cost	$(15n+6)M_1 + (10n+2)C_1 + I_3$

Note that there is another relationship among them [9]. A trinomial basis is used in [10], and a normal basis is used in [9] for a basis of $\mathbb{F}_{3^n}$ over $\mathbb{F}_3$. If the normal basis is used, then C_1 becomes virtually for free, however, M_1 becomes considerably higher.

First we discuss for $n = 97$ corresponding standard security. The cost of the final exponentiation appeared in [3] is $267.6M_1$, which is about 25% of the total cost $1071.9M_1$ of the η_T pairing. On the other hand, the computational cost of our proposed final exponentiation is $122.8M_1$ and the total takes $927.1M_1$, namely it is about 13% of the whole η_T pairing. Therefore, the proposed scheme can compute the η_T pairing about 16% faster than the previously known

Table 4. Estimations of Table 3 using Table 2 for some n's (unit:M_1)

η_T pairing $(\eta_T(P,Q)^{3^{(n+1)/2}})$		$n = 97$	163	193	239	353
Proposed final exponentiation		122.7	118.9	122.6	130.1	126.9
(Algorithm 3)	total cost	927.1	1411.4	1647.2	2007.7	2855.6
Proposed GPS encoding ($b' = 1$)		162.5	158.8	162.1	172.1	168.1
(Algorithm 4)	total cost	966.8	1451.3	1686.7	2049.8	2896.8
Proposed GPS encoding ($b' = -1$)		116.7	112.9	116.6	124.1	120.9
(Algorithm 4)	total cost	921.1	1405.4	1641.2	2001.7	2849.6
Ordinary final exponentiation		267.6	263.3	267.0	274.4	271.1
([3])	total cost	1071.9	1555.8	1791.6	2152.1	2999.9

Duursma-Lee algorithm		97	163	193	239	353
Final exponentiation		45.7	46.0	45.5	48.0	47.2
([11])	total cost	1636.3	2613.4	3077.1	3785.9	5487.4
GPS encoding		21.7	22.0	21.5	24.0	23.2
([10])	total cost	1612.3	2589.4	3053.1	3761.9	5463.4

algorithms. Table 3 and Table 4 conclude a final exponentiation and the GPS encoding for the η_T pairing. If $b' = 1$, the cost of the GPS encoding and the total take $116,8M_1$ and $921.1M_1$, respectively. Hence, the GPS encoding for η_T is able to compute the η_T pairing moreover faster.

We notice that the larger extension degree n is, the smaller the ratio of the cost of final exponentiation is. Then the larger n is, the smaller the improvement rate is. For example, the improvement rate of whole cost of the η_T pairing is about 5% only for $n = 353$. However, it is non-negligible. Final exponentiation used torus or the GPS encoding for η_T gives more than 2 times of efficient final exponentiation for any n.

5 Conclusion

In this paper we presented some new efficient algorithms for the final exponentiation of the η_T pairing. We deploy new encoding techniques for the embedded group $\mathbb{F}_{3^{6n}}$, which allow us to efficiently perform the powering by $3^n + 1$ and the inversion. The total cost of computing the η_T pairing with $n = 97$ has become about 16% faster than the previously known methods.

Acknowledgments

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO), Japan. The authors would like to thank Steven Galbraith for his comments on the previous version.

References

1. P. Barreto, "Efficient algorithms for pairing-based cryptosystems," *CRYPTO 2002*, LNCS 3027, pp.354-368, 2002.
2. P. Barreto, H. Kim, B. Lynn and M. Scott, "Efficient algorithms for pairing-based cryptosystems," *CRYPTO 2002*, LNCS 2442, pp.354-368, 2002.
3. P. Barreto, S. Galbraith, C. Ó hÉigeartaigh and M. Scott, "Efficient pairing computation on supersingular abelian varieties," Cryptology ePrint Archive, Report 2004/375, 2004. (*Designs, Codes and Cryptography*, Springer–Verlag, to appear.)
4. J.-L. Beuchat, M. Shirase, T. Takagi and E. Okamoto, "An algorithm for the η_T pairing calculation in characteristic three and its hardware implementation," Cryptology ePrint Archive, Report 2006/327, 2006. (*Arith 18*, IEEE Computer Society, to appear.)
5. D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *SIAM Journal of Computing*, Vol. 32, No. 3, pp. 586-615, 2003.
6. D. Boneh, C. Gentry and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," *CRYPTO 2005*, LNCS 3621, pp.258-275, 2005.
7. D. Boneh, B. Lynn and H. Shacham, "Short signature from the Weil pairing," *Journal of Cryptology*, Vol. 17, No. 4, pp. 297-319, 2004.
8. I. Duursma and H. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," *ASIACRYPT 2003*, LNCS 2894, pp.111-123, 2003.
9. R. Granger, D. Page and M. Stam, "Hardware and software normal basis arithmetic for pairing-based cryptography in characteristic three," *IEEE Transaction on Computers*, Vol. 54, No. 7, July 2005, pp.852-860, 2005.
10. R. Granger, D. Page and M. Stam, "On Small characteristic algebraic tori in pairing-cased cryptography," *LMS Journal of Computation and Mathematics*, vol. 9, pp.64-85, 2006.
11. T. Kerins, W. Marnane, E. Popovici and P. Barreto, "Efficient hardware for the Tate pairing calculation in characteristic three," *CHES 2005*, LNCS 3659, pp.412-426, 2005.
12. S. Kwon, "Efficient Tate pairing computation for supersingular elliptic curves over binary fields," Cryptology ePrint Archive, Report 2004/303, 2004.
13. R. Ronan, C. Ó hÉigeartaigh, C. Murphy, T. Kerins and P. Barreto, "Hardware implementation of the η_T pairing in characteristic 3," Cryptology ePrint Archive, Report 2006/371, 2006.
14. C. Shu, S. Kwon and K. Gaj, "FPGA accelerated Tate pairing based cryptosystems over binary fields," Cryptology ePrint Archive, Report 2006/179, 2006.

A Proofs of Lemma 2

Lemma 2. *Let* $n \equiv 1, 5 \pmod 6$. *For* $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$ *we can compute* $Y = \Lambda(X) = X^{3^n+1} = Y_0 + Y_1\sigma$ *with 9 multiplications in* $\mathbb{F}_{3^n}$ *as follows:*
Let $z_0 \sim z_8$ *be defined as*

$$
\begin{array}{lll}
z_0 = x_0x_4, & z_1 = x_1x_5, & z_2 = x_2x_4, \\
z_3 = x_3x_5, & z_4 = (x_0 + x_1)(x_4 - x_5), & z_5 = x_1x_2, \\
z_6 = x_0x_3, & z_7 = (x_0 + x_1)(x_2 + x_3), & z_8 = (x_2 + x_3)(x_4 - x_5),
\end{array}
$$

then, Y can be computed as following table, where $X_0 = x_0 + x_2\rho + x_4\rho^2, X_1 = x_1 + x_3\rho + x_5\rho^2$ and $Y_0 = y_0 + y_2\rho + y_4\rho^2, Y_1 = y_1 + y_3\rho + y_5\rho^2$ $(x_i, y_i \in \mathbb{F}_{3^n})$ for $i = 0, 1, ..., 5$.

Case of $n \equiv 1 \pmod 6$	*Case of* $n \equiv 5 \pmod 6$
$y_0 = 1 + z_0 + z_1 - bz_2 - bz_3$	$y_0 = 1 + z_0 + z_1 + bz_2 + bz_3$
$y_1 = z_1 + z_4 + bz_5 - z_0 - bz_6$	$y_1 = z_1 + z_4 - bz_5 - z_0 + bz_6$
$y_2 = z_7 - z_2 - z_3 - z_5 - z_6$	$y_2 = z_5 + z_6 - z_7$
$y_3 = bz_0 + z_3 + z_8 - z_2 - bz_1 - bz_4$	$y_3 = -bz_0 + z_3 + z_8 - z_2 + bz_1 + bz_4$
$y_4 = bz_2 + bz_3 + bz_7 - bz_5 - bz_6$	$y_4 = bz_2 + bz_3 + bz_7 - bz_5 - bz_6$
$y_5 = bz_3 + bz_8 - bz_2$	$y_5 = -bz_3 - bz_8 + bz_2$

Proof. We prove Lemma 2 for $n \equiv 1 \pmod 6$. The proof for $n \equiv 5 \pmod 6$ is omitted since it is almost same for $n \equiv 1 \pmod 6$.

Note that $\rho^{3^n} = \rho + b$, $(\rho^2)^{3^n} = \rho^2 - b\rho + 1$ in the case of $n \equiv 1 \pmod 6$. For $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$ we have the relationship:

$$X^{3^n+1} = X_0^{3^n+1} + X_1^{3^n+1} + (X_0^{3^n}X_1 - X_1^{3^n}X_0)\sigma,$$

$$X_0^{3^n} = (x_0 + bx_2 + x_4) + (x_2 - bx_4)\rho + x_4\rho^2, \tag{7}$$

$$\begin{aligned} X_0^{3^n+1} = (x_0^2 + bx_0x_2 + x_0x_4 - bx_2x_4 - x_4^2) + (-x_0x_2 - bx_0x_4 + bx_2^2)\rho \\ +(-x_0x_4 + x_2^2 - x_4^2)\rho^2. \end{aligned} \tag{8}$$

We similarly see

$$X_1^{3^n} = (x_1 + bx_3 + x_5) + (x_3 - bx_5)\rho + x_5\rho^2, \tag{9}$$

$$\begin{aligned} X_1^{3^n+1} = (x_1^2 + bx_1x_3 + x_1x_5 - bx_3x_5 - x_5^2) + (-x_1x_3 - bx_1x_5 + bx_3^2)\rho \\ +(-x_1x_5 + x_3^2 - x_5^2)\rho^2. \end{aligned} \tag{10}$$

$X_0^2 + X_1^2 = 1$ is satisfied since $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$. This derives the following equation.

$$(x_0 + x_2\rho + x_4\rho^2)^2 + (x_1 + x_3\rho + x_5\rho^2)^2 = 1\ (= 1 + 0\rho + 0\rho^2).$$

This equation gives

$$\begin{cases} x_0^2 + x_1^2 = 1 + bx_2x_4 + bx_3x_5 \\ x_2^2 + x_3^2 = x_0x_4 + x_1x_5 - bx_0x_2 - bx_1x_3 - bx_2x_4 - bx_3x_5 \\ x_4^2 + x_5^2 = bx_0x_2 + bx_1x_3 + bx_2x_4 + bx_3x_5. \end{cases} \tag{11}$$

By Eqs. (8), (10), (11)

$$\begin{aligned} X_0^{3^n+1} + X_1^{3^n+1} &= y_0 + y_2\rho + y_4\rho^2 \\ &= (1 + x_0x_4 + x_1x_5 - bx_2x_4 - bx_3x_5) \\ &\quad +(x_0x_2 + x_1x_3 - x_2x_4 - x_3x_5)\rho \\ &\quad +(bx_0x_2 + bx_1x_3 + bx_2x_4 + bx_3x_5)\rho^2. \end{aligned}$$

And by Eqs. (7), (9), (11)

$$\begin{aligned} X_0^{3^n} X_1 - X_1^{3^n} X_0 &= y_1 + y_3\rho + y_5\rho^2 \\ &= (bx_1x_2 + x_1x_4 - bx_0x_3 - x_0x_5) \\ &\quad +(bx_0x_5 + x_3x_4 - bx_1x_4 - x_2x_5)\rho \\ &\quad +(bx_3x_4 - bx_2x_5)\rho^2. \end{aligned}$$

Moreover, we define $z_0, \ldots, z_8$ by

$$\begin{aligned} &z_0 = x_0x_4,\ z_1 = x_1x_5,\ z_2 = x_2x_4,\ z_3 = x_3x_5,\ z_4 = (x_0 + x_1)(x_4 - x_5), \\ &z_5 = x_1x_2,\ z_6 = x_0x_3,\ z_7 = (x_0 + x_1)(x_2 + x_3),\ z_8 = (x_2 + x_3)(x_4 - x_5). \end{aligned}$$

Then we have

$$\begin{aligned} y_0 &= 1 + x_0x_4 + x_1x_5 - bx_2x_4 - bx_3x_5 &&= 1 + z_0 + z_1 - bz_2 - bz_3, \\ y_1 &= bx_1x_2 + x_1x_4 - bx_0x_3 - x_0x_5 &&= z_1 + z_4 + bz_5 - z_0 - bz_6, \\ y_2 &= x_0x_2 + x_1x_3 - x_2x_4 - x_3x_5 &&= z_7 - z_2 - z_3 - z_5 - z_6, \\ y_3 &= bx_0x_5 + x_3x_4 - bx_1x_4 - x_2x_5 &&= bz_0 + z_3 + z_8 - z_2 - bz_1 - bz_4, \\ y_4 &= bx_0x_2 + bx_1x_3 + bx_2x_4 + bx_3x_5 &&= bz_2 + bz_3 + bz_7 - bz_5 - bz_6, \\ y_5 &= bz_3z_4 - bz_2z_5 &&= bz_3 + bz_8 - bz_2. \end{aligned}$$

Consequently Lemma 2 is showed. □

B Powering by 3^n and 3^n-th Root in $\mathbb{F}_{3^n}$

This section explains that a powering by 3^n or 3^n-th root in $\mathbb{F}_{3^{6n}}$ is computed virtually for free. Recall that $n \equiv 1, 5 \pmod 6$ is a necessary condition of the Duursma-Lee algorithm and the η_T pairing. We deal with only for $n \equiv 1 \pmod 6$, however, the discussion for $n \equiv 5 \pmod 6$ becomes almost same. It follows that

$$\sigma^{3^n} = \begin{cases} \sigma & \text{if } n \equiv 0 \pmod 2 \\ -\sigma & \text{if } n \equiv 1 \pmod 2 \end{cases}, \quad \rho^{3^n} = \begin{cases} \rho & \text{if } n \equiv 0 \pmod 3 \\ \rho + b & \text{if } n \equiv 1 \pmod 3 \\ \rho - b & \text{if } n \equiv 2 \pmod 3 \end{cases}.$$

If $n \equiv 5 \pmod 6$ then we have

$$\sigma^{3^n} = -\sigma,\ \rho^{3^n} = \rho + b,\ (\rho^2)^{3^n} = \rho^2 - b\rho + 1.$$

B.1 Powering by 3^n

Let $Y = X^{3^n}$ for $X \in \mathbb{F}_{3^{6n}}^*$, where $X = x_0 + x_1\sigma + x_2\rho + x_3\sigma\rho + x_4\rho^2 + x_5\sigma\rho^2$ and $Y = y_0 + y_1\sigma + y_2\rho + y_3\sigma\rho + y_4\rho^2 + y_5\sigma\rho^2$ for some $x_i,\ y_i \in \mathbb{F}_{3^n}$. Then we have

$$\begin{cases} y_0 = x_0 + bx_2 + x_4 \\ y_1 = -x_1 - bx_3 - x_5 \\ y_2 = x_2 - bx_4 \\ y_3 = -x_3 + bx_5 \\ y_4 = x_4 \\ y_5 = -x_5 \end{cases} \tag{12}$$

since

$$\begin{aligned}
&(x_0 + x_1\sigma + x_2\rho + x_3\sigma\rho + x_4\rho^2 + x_5\sigma\rho^2)^{3^n} \\
&= x_0 + x_1(\sigma)^{3^n} + x_2(\rho)^{3^n} + x_3(\sigma\rho)^{3^n} + x_4(\rho^2)^{3^n} + x_5(\sigma\rho^2)^{3^n} \\
&= x_0 + x_1(-\sigma)^{3^n} + x_2(\rho + b) + x_3(-\sigma\rho - b\sigma) + x_4(\rho^2 - b\rho + 1) \\
&\qquad + x_5(\sigma\rho^2 + b\sigma\rho - \sigma) \\
&= (x_0 + bx_2 + x_4) + (-x_1 - bx_3 - x_5)\sigma + (x_2 - bx_4)\rho \\
&\qquad + (-x_3 + bx_5)\sigma\rho + x_4\rho^2 - x_5\sigma\rho^2.
\end{aligned}$$

Note that $x_i^{3^n} = x_i$ and $y_i^{3^n} = y_i$ since $x_i,\ y_i \in \mathbb{F}_{3^n}$. Therefore a powering by 3^n is computed virtually for free.

B.2 3^n-th Root

Let $Y = \sqrt[3^n]{X}$ for $X \in \mathbb{F}^*_{3^{6n}}$, where $X = x_0 + x_1\sigma + x_2\rho + x_3\sigma\rho + x_4\rho^2 + x_5\sigma\rho^2$ and $Y = y_0 + y_1\sigma + y_2\rho + y_3\sigma\rho + y_4\rho^2 + y_5\sigma\rho^2$ for some $x_i,\ y_i \in \mathbb{F}_{3^n}$. Note that a 3^n-th root operation in characteristic three is uniquely determined. We have

$$\begin{cases}
x_0 = y_0 + by_2 + y_4 \\
x_1 = -y_1 - by_3 - y_5 \\
x_2 = y_2 - by_4 \\
x_3 = -y_3 + by_5 \\
x_4 = y_4 \\
x_5 = -y_5
\end{cases} \tag{13}$$

by Eq.(12) since $X = Y^{3^n}$. Solving Eq.(13) for each y_i gives

$$\begin{cases}
y_0 = x_0 - bx_2 + x_4 \\
y_1 = -x_1 + bx_3 - x_5 \\
y_2 = x_2 + bx_4 \\
y_3 = -x_3 - bx_5 \\
y_4 = x_4 \\
y_5 = -x_5.
\end{cases}$$

Therefore a 3^n-th root operation is computed virtually for free.

Towards Minimizing Memory Requirement for Implementation of Hyperelliptic Curve Cryptosystems

Pradeep Kumar Mishra[1], Pinakpani Pal[2], and Palash Sarkar[2]

[1] Centre for Information Security and Cryptography
University of Calgary
Calgary (Canada)
pradeep@math.ucalgary.ca
[2] Indian Statistical Institute
Kolkata (India)
{pinak, palash}@isical.ac.in

Abstract. Elliptic (ECC) and hyperelliptic curve cryptosystems (HECC) have emerged as cryptosystems for small hand-held and mobile devices. Extensive research has been carried out for their secure and efficient implementation on these devices. These devices come with very low amount of resources, efficient memory management is an important issue in all such implementations. HECC arithmetic is now generally performed using so called explicit formulas. The main goal of these formulas is to reduce the number of finite field operations (multiplications and squarings). On the other hand, reducing the memory requirement is also important. To the best of our knowledge, the literature on HECC implementation does not seriously consider this aspect. This is the first work to obtain memory efficient versions of various explicit formulas appearing in the literature. In certain cases, we are also able to determine the minimum memory requirement and obtain a memory optimal implementation. We believe that these formulas will be extremely useful to designers of HECC. Our basic technique is essentially an exhaustive search with heuristic strategies for improving the run-time.

Keywords: Elliptic and hyperelliptic curve cryptosystems, memory, explicit formula, divisor addition, divisor doubling, scalar multiplication.

1 Introduction

For one and half a decade or so, elliptic and hyperelliptic curve cryptosystems have occupied the center stage of public key cryptographic research. The main reason behind it is their versatility. These are amongst the most ideal cryptosystems to be implemented on small mobile devices with low computing power. There is no known sub-exponential algorithm to solve elliptic or hyperelliptic curve discrete logarithm problem for carefully chosen curves and other security parameters. This ensures a high level of security for smaller key length and makes these cryptosystems suitable for such small devices.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 269–283, 2007.

In these cryptosystems, the most dominant operation is the computation of so called scalar multiplication. Unless otherwise stated, in the current work, by a *point* we will generally mean a point on an elliptic curve or a divisor in the Jacobian of a hyperelliptic curve. Let X be a point and let m be a positive integer. The operation of computing mX is called the scalar multiplication. It is generally computed by a series of point doublings and additions. Research community has put extensive efforts to compute the scalar multiplication efficiently and securely.

The efficiency of scalar multiplication is intimately connected to the efficiency of point addition and doubling algorithms. The efficiency of these algorithms, on the other hand, depends upon the point representation. In affine coordinates, both these operations involve inversion of field elements which is considered a very expensive operation. To avoid inversions, various other coordinate systems like projective, Jacobian, modified Jacobian have been proposed for elliptic curves.

For hyperelliptic curves, Koblitz in his pioneering work [9], had proposed Cantor's algorithm to be used for divisor addition and doubling. Later, it was felt that the computation can be accelerated by fixing the genus of the curves and computing the parameters of the resultant divisor explicitly. Such an algorithm is called an explicit formula. Many proposals of explicit formula have come up in literature and the ones proposed by Lange in [10,11,12] are the currently known most efficient ones for general curves of genus 2.

ECC and HECC are considered to be the ideal cryptosystem for mobile devices. Mobile devices are generally equipped with very little computing power. Before trying to implement a cryptosystem on these devices one has to ensure that the resources, particularly memory, available on these devices are sufficient for the implementation. For ECC, the formulas are smaller, involving 7 to 15 multiplications and squarings in the underlying field in non-affine arithmetic. So it is simple to calculate the numbers of registers required to store the inputs, outputs and intermediate variables. In many works reported in literature on ECC, the authors have provided the number of registers required for the computation. These figures are obtained by manual checking. However, to our knowledge, there is no result stating that a particular ECC algorithm cannot be implemented in less than a certain amount of memory.

There has been no study of exact memory requirement for an implementation of HECC. In [2] the authors have briefly touched the topic. Besides that there has been no mention of memory requirement in any work on HECC so far. The point addition and doubling algorithm for ECC can be found in many papers as a sequence of three address codes, like, $R_i = R_j \ op \ R_k$, where R_i, R_j, R_k are register names or constants and *op* is an arithmetic operation. In the current work we will refer to this format as *Explicit Register Specified Format (ERSF)*. Looking at a formula in ERSF, one can know exactly how many registers will be required for its implementation. Unfortunately, no HECC explicit formulas occurring in the literature has been described in ERSF. All are described as a set of mathematical equations. We will refer to this format of representing a formula as *raw* format.

Probably, the reason behind all HECC formulas appearing in raw format only is the fact that HECC formulas are relatively complex ones compared to those of ECC. An HECC (genus 2) formula involves around 25 to 50 multiplication with or without an inversion. The first step for expressing such a formulas in ERSF is to know how many registers will be required. For a long formula it is difficult to manually find out how many registers will suffice. It is nearly impossible to say what is the minimal requirement.

The question is: *Given an explicit formula what is the minimum number of registers required to compute it sequentially or in parallel*? In the current work, we (in Section 3) provide an empirical solution to this problem. For elliptic curves, we checked for the general addition formula in Jacobian coordinates and found that it can not be executed with less than 7 registers. It has been reported in literature earlier that this operation can be done in 7 registers, each capable of containing a field element of the underlying field size. Our finding ensures that it can not be done in less.

We have used our programs to find the minimum register requirement for many formulas in HECC. The formulas proposed by Pelzl et al [18] for a special class of curves are very efficient ones. Their doubling formulas uses 10 registers and the addition uses 15. Similar formulas are proposed by Lange in [10]. These set of formulas use 11 and 15 registers for doubling and addition respectively. The doubling formula in [10] requires 6 curve constants to be stored. Thus, (baring the storage required for curve constants) for computing the scalar multiplication both set of formulas require 15 registers. Thus, although the formulas proposed in [18] are cheaper in number of operations, they use the same amount of memory as the ones in [10]. More recently Lange and Stevens [14] have come out with more efficient doubling formulas for certain class of curves over binary fields. These formulas not only require very few field operations per group operation but also are very memory efficient. All our findings have been described in Section 4. For more results involving several other formulae, reader can refer the full version [16] of the paper.

2 Background

Hyperelliptic curve cryptosystems were proposed by Koblitz [9] in 1987. In this section we provide a brief overview of hyperelliptic curves. For details, readers can refer to [1,15]. Let K be a field and let $\overline{K}$ be the algebraic closure of K. A *hyperelliptic curve* C of genus g over K is an equation of the form $C : y^2+h(x)y = f(x)$ where $h(x)$ in $K[x]$ is a polynomial of degree at most $g, f(x)$ in $K[x]$ is a monic polynomial of degree $2g + 1$, and there are no singular points (x, y) in $\overline{K} \times \overline{K}$. Elliptic curves are hyperelliptic curves of genus 1.

The elliptic curve group law does not apply to hyperelliptic curves. The groups used in hyperelliptic curve cryptosystems are the divisor class group, each group element represented by a special kind of divisor called *reduced divisor*. The beauty of the hyperelliptic curves is that the group of divisor classes is isomorphic to the group of ideal classes. That leads to a nice canonical

representation for each group element. Each group element can be represented by a pair of polynomials of small degree, $(u(x), v(x))$, where $deg(v) < deg(u) \leq g$ and u divides $v^2 - hv + f$. Koblitz in his pioneering work suggested to perform the group operation using Cantor's algorithm [3].

Cantor's algorithm for divisor class addition and doubling were quite complex for an efficient implementation. Later it was realized that the efficiency of group law algorithms can be enhanced by fixing the genus of the curve and computing the coefficients of the polynomials representing the resultant divisor directly from those of the input divisor(s). Thus the group law algorithms become sequences of field operation. Such an algorithm is called an explicit formula. Spallek [20] was the first attempt to compute divisor addition by explicit formula for genus 2 curves over fields of odd characteristic. Harley [7] improved the running time of the algorithm in [20]. Gaudry and Harley [6] observed that one can derive different explicit formula for divisor operations depending upon the weight of the divisors. Later many researchers came out with various explicit formula for various genera of hyperelliptic curves. An overview of most proposals can be found e. g. in [17].

In the current work we concentrate on curves of genus 2. For most general curves of genus 2, the explicit formulas proposed by Lange are the currently known most efficient ones. In [10], Lange's addition (HCADD) and doubling (HCDBL) involve inversion. Taking the lead from the different projective coordinates in ECC, Lange in [11], [12] has proposed explicit formulas in various coordinate systems. In [11] she has proposed formulas in "projective" coordinates. Introducing a new variable, a field element in the structure of a divisor, the inversion can be avoided in HCADD and HCDBL as in ECC. Again taking the lead from Chudonovski Jacobian coordinates in ECC, Lange has proposed her "new coordinates" in [12], a representation using weighted coordinates. This lead to faster HCADD and especially HCDBL. The latest version all these formulas with an extensive comparison of coordinate systems is available in [13].

More recently, Pelzl et al. [18] and Lange et al. [14] have proposed divisor addition and doubling algorithms for special classes of curves, in which doublings are quite cheaper. In Table 1, we provide the complexity of various formulas we investigated in the current work.

Register Sufficiency Problem. The problem of minimizing the number of intermediate variables required for executing a set of arithmetic formulas has been studied earlier. The problem is called the register sufficiency problem and the decision version is known to be NP-complete [19]. See [5, page 272] for further details. The minimization version has also been studied in the literature. According to the compendium of NP-optimization problems [4], there is an $O(\log^2 n)$ (where n is the number of operations) approximation algorithm for this problem [8]. We would like to emphasize that we are not providing any new solution to this computationally difficult problem. Our strategy is to essentially employ an exhaustive search technique with some heuristics for bringing down the runtime. Our main emphasis is to obtain memory efficient (in some cases, optimal) implementation of explicit formulas for perform HECC arithmetic.

Table 1. Complexity of Explicit Formulas

Name of Work	Char	Cost(HCADD)	Cost(HCDBL)	Cost (mHCADD)
Lange [10]	All	$1[i] + 22[m] + 3[s]$	$1[i] + 22[m] + 5[s]$	-
Lange [12]	Odd	$47[m] + 7[s]$	$34[m] + 7[s]$	$36[m] + 5[s]$
Lange [12]	Even $h_2 \neq 0$	$46[m] + 4[s]$	$35[m] + 6[s]$	$35[m] + 6[s]$
Lange [12]	Even $h_2 = 0$	$44[m] + 6[s]$	$29[m] + 6[s]$	$34[m] + 6[s]$
Pelzl et al [18]	Even	$1[i] + 21[m] + 3[s]$	$1[i] + 9[m] + 6[s]$	-
Lange et al [14]	Even	-	$1[i] + 5[m] + 6[s]$	-

3 Our Methodology

Our primary aim in this work is to answer the question:

Problem: Given an explicit formula $\mathcal{F}$, what is the minimum number of intermediate variables required to be stored to sequentially execute $\mathcal{F}$?

Let $\mathcal{F}$ be an explicit formula. Let $p_1, \ldots, p_k$ be the inputs to $\mathcal{F}$. We can look at $\mathcal{F}$ as a sequence of arithmetic operations, each having a unique id, like; $Id_i : p_i = q_i \ op_i \ r_i, k \leq i \leq n$, where op_i is one of the binary operations $\{+, -, *, /\}$ and q_i, r_i are among the p_j's $j < i$. In fact, explicit formula in literature generally occur in raw format. We can convert them into this form by a simple parser program.

We will call a sequence $S = \{Id_{i_1}, Id_{i_2}, \ldots, Id_{i_{n-k+1}}\}$ of operations id's (or simply $S = \{i_1, i_2, \ldots, i_{n-k+1}\}$) of $\mathcal{F}$ a *valid sequence* if $\mathcal{F}$ can be computed by executing its operations in the order as dictated by the sequence S. For example if $\mathcal{F} = \{Id_1, Id_2, Id_3, Id_4\}$, where $Id_1 : p_4 = x * y$, $Id_2 : p_5 = p_4 * z$, $Id_3 : p_6 = y * z$ and $Id_4 : p_7 = p_5 * p_6$, then there are only three valid sequences, namely, $\{1, 2, 3, 4\}$, $\{1, 3, 2, 4\}$ and $\{3, 1, 2, 4\}$. $\mathcal{F}$ can not be executed in any other order. Our interest is in knowing which valid sequence needs the minimum number of intermediate variables for executing the explicit formula $\mathcal{F}$.

Let $\mathcal{F}$ be an explicit formula and let $\mathcal{A}_0$ be the set of inputs to it. In $\mathcal{F}$, there are certain computations which can be computed from the set $\mathcal{A}_0$ of inputs to $\mathcal{F}$. After one or more of them are executed we get some intermediate values which can trigger some more operations of $\mathcal{F}$. Let V_0 be the set of computations in $\mathcal{F}$, which can be computed directly from the set $\mathcal{A}_0$ of inputs to $\mathcal{F}$. Let $|V_0| = \alpha_0$ be the size of the set V_0. So one can begin the execution of $\mathcal{F}$ starting from any one of these α_0 operations.

Consider a computation tree of the following type. There is a root node. The number of nodes at the first level is α_0, and each of the first level nodes correspond to one of the operations in V_0. The subtree rooted at a first level node correspond to the computation tree for completing the rest of the formula after the operation corresponding to this node has been executed. The leaf nodes of the computation tree correspond to the last operation in the particular sequence of operations. Clearly, all possible valid computation sequences for completing $\mathcal{F}$ are described by all possible paths from the root to the leaf nodes.

Our basic search strategy is essentially a depth first search of the above described computation tree. Any path from root to leaf gives a valid sequence requiring a particular number of registers (or intermediate variables) for its completion. A depth first search of the entire tree will clearly provide the minimum number of intermediate variables required to complete $\mathcal{F}$ and also a valid execution sequence which attains this minimum number. To bring down the running time we adopt the following four strategies:

1. **Neglecting the paths which requires same number of intermediate variables as the known one:** We use early abort strategy for improving the running time of the algorithm. As we get the first valid sequence we count the number of intermediate variables required to be stored to execute $\mathcal{F}$ by that sequence and store it in a variable, say β. While looking for another path by backtracking, we check the size of the set of intermediate variables after each step. If the current size is equal to the value stored in β, then we need not proceed along this path further. It is not going to yield a more economical path. So we abandon this path and look for another one. If a particular valid sequence needs less than β intermediate values, then we replace the value of β by this new value.
2. **Avoiding the count of the number of intermediate variables at each step:** Counting the number of variables at each step of the algorithm is a time consuming operation. Suppose the value stored currently in β is t. While looking for a new path, we save time by not counting the number of minimum variables till the path is t operations long. Because, if less than t operations have been executed then the number of intermediate variables can not be more than t.
3. **Backtracking several steps at a time:** After finding a valid sequence, instead of backtracking one step (to the last step) at a time, we can go back until a step b such that $\max_{0\leq i\leq b} |\mathcal{A}_i - \mathcal{A}_0| = \beta - 1$. This will reduce the task of going to each level and hence will aid to efficiency. Note that this does not affect the optimality of the final result. That is because the paths which we are skipping need at least β intermediate variables.
4. **Using ordered sets in place of V_j's:** Before starting the first step, we scan the explicit formula and make a frequency table of all the inputs and intermediate variables. Against name of each variable it contains the number of times it has been used in the formula, i.e. the number of times it appears in the right hand side of some equation in $\mathcal{F}_j$. We treat the sets V_j's as ordered sets and ordered according as priorities assigned to these equations. The highest priority is assigned to those, in which the input variables have lower frequency. Each time we choose an equation for computation, we update the frequency table by reducing the frequencies of the involved input variables by 1.

These optimization techniques for running time of the algorithm have paid high dividends. It is observed that an implementation using these techniques

runs much faster than the one without them. These techniques however, do not guarantee that an implementation of this search strategy will terminate in reasonable time for any large explicit formula. An explicit formula may contain a huge number of equations. In that case the program may run for a considerable duration of time and the last value of β may be accepted as the good minimal value. The actual minimum may be lesser.

3.1 The Forward and Reverse Programs

As said earlier, each explicit formula in raw format was modified to be a sequence of binary operations. This is the preprocessing done to each of the raw formula under consideration. This preprocessed formula was given as input to a program embodying the methodology described in the last section, which calculated the minimum number of intermediate variables required for sequential execution of the explicit formula. When the program terminates it outputs exactly how many intermediate variables are required for an execution of the formula and the corresponding valid sequence. As our methodology is an exhaustive search kind with some running time optimization measures, for a long input file it may take substantial amount of time for a complete execution. In order to get the results within a reasonable time, another program was employed. We will refer to this later program as the *reverse* program and the former as *forward* program. The *reverse* program initially takes $k = 1$ and using the same logic as *forward* checks if the explicit formula can be executed with k temporary locations. If not it reports this fact and tries again with k replaced by $k + 1$. When it gets an affirmative answer it outputs the corresponding value of k and the corresponding valid sequence. The *forward* program after obtaining a path requiring l intermediate variables looks for path needing less than l intermediate locations. If during the search process it comes across a path which also requires l locations, then *forward* abandons this path and look for a newer one. The *reverse* program uses the same logic, taking $k = 1, 2 \ldots$. We run both the programs with the same input file on two different machines. If either of *forward* or *reverse* terminates we get the result. Otherwise, if at some point of time *forward* reports the formulas can be executed with k variables and at the same time *reverse* reports it can not be done with less than $k-1$ intermediate locations, then also the conclusion follows.

The employment of *reverse* helped us to get to the conclusions quite early. Some of the explicit formula under consideration have more that 100 lines of three address codes (i.e. total number of arithmetic operations is more than 100). In spite of the speed up measures described above, there is no guarantee that *forward* will terminate. In fact, we ran *forward* program without any speed up measure on some longer inputs and found that it did not terminate in a week. Even after the optimization methods described above are employed, for some of the formulas given in [12] it did not terminate for three days. Of course it ran much faster. Surprisingly (because they have quite a small number of operations), for the doubling formulas in new coordinates in even characteristic, in both cases $h_2 \neq 0$ and $h_2 = 0$, *forward* did not terminate. So, we took help

of the *reverse* program to derive our conclusions. With the help of *reverse* we could get conclusion on any formula in less than two days.

4 Results

The addition (ECADD) and doubling (ECDBL) formulas in ECC have received much attention from the research community and the formulas are quite simpler in comparison to those in HECC. It has been reported by many researchers that the ECADD in mixed Jacobian coordinates for most general curves over fields of odd characteristic needs 7 registers for implementation. To our knowledge, there is no result stating that it can not be executed in less than seven registers. This aroused our curiosity for testing these formulas in our methodology. We experimented with ECADD and ECDBL in Jacobian coordinates and found that ECADD and ECDBL can be implemented with no less than 7 and 6 registers respectively. Both *forward* and *reverse* program reported this fact. As we intended this work to focus on HECC, we did not pay attention to other ECC algorithms.

We applied our methodology to many formula in HECC. First of all, we applied our methodology to Lange's formulas in new co-ordinates [12]. These formulas are the most efficient ones for general hyperelliptic curves of genus 2. All these formulas are inversion-free. However, the cost of avoiding the inversions is more than an inversion in binary fields. Hence for an implementation over binary fields, affine arithmetic still looks quite attractive. So we used our methodology to calculate the minimum number of registers required for implementing HCDBL and HCADD in affine coordinates also. We used the formulas presented in [10] which are the most efficient ones in affine coordinates for general curves of genus 2. Pelzl et al. [18], have proposed a very efficient HCDBL formula for a special class of curves. We investigated the memory requirement of the HCADD and HCDBL formulas presented in [18] also.

In [14], many efficient doubling formulas have been presented. Many situations, (like $deg(h) = 1$ or 2, $h_0 = 0$ or $\neq 0$, h_1 is small etc.) have been considered. If a particular variable is small then multiplication by that variable can be effected by some additions. The number of additions will depend upon the value of the small value. Hence we have not inquired these situations. When $deg(h) = 1$ and h_1^2 and h_1^{-1} are precomputed, the doubling formula proposed in [14] is very efficient. To compute a doubling $(1[i] + 9[m] + 5[s])$ one need only 7 registers. However to compute the scalar multiplication one has to couple it with an addition formula which requires 15 registers.

Note that HCADD and HCDBL for genus 2 curves have many special cases. The most general and also the most frequent case is the one in which the divisor(s) are of full weight, i.e. if $D = (u, v)$ is the divisor, then $deg(u) = 2, deg(v) = 1$. In the current work we concentrate on the most general and the frequent case only. The same methodology can be applied to other special cases as well. Also, the cost of various operations we have given in the Table 2 does not corroborate with the costs provided in the corresponding papers. That is because

authors generally avoid counting the multiplication and squaring of/with curve constants. In some formulas such operations occur in significant numbers. For example in even characteristic, doubling formula ($h_2 \neq 0$), there are 21 such multiplications/squarings. We have taken account of all of them.

We use the following naming convention for the name of various algorithms. The formulas presented in [10] and in [18] are in affine coordinates and hence a superscript $\mathcal{A}$ is used for them, e.g. HCADD$^{\mathcal{A}}$. The formulas in [12] are in Lange's new coordinates. For the formulas in these new coordinates over fields of even characteristic we will use the superscript $\mathcal{N}e$ and for those over fields of odd characteristic we will use superscript $\mathcal{N}o$. Divisor addition algorithms in mixed coordinates will be denoted by a suffix 'm' e.g. mHCADD$^{\mathcal{N}o}$.

Table 2. Register Requirement for Various Explicit Formulas

Algorithm	Proposed in	Characteristic	Cost	Regs Req
HCADD$^{\mathcal{A}}$	[10]	All	$1[i] + 22[m] + 3[s] + 44[a]$	15
HCDBL$^{\mathcal{A}}$	[10]	All	$1[i] + 22[m] + 5[s] + 56[a]$	11
HCADD$^{\mathcal{N}o}$	[12]	Odd	$49[m] + 7[s] + 34[a]$	23
mHCADD$^{\mathcal{N}o}$	[12]	Odd	$36[m] + 5[s] + 35[a]$	16
HCDBL$^{\mathcal{N}o}$	[12]	Odd	$36[m] + 7[s] + 41[a]$	20
HCADD$^{\mathcal{N}e}_{h_2 \neq 0}$	[12]	Even	$52[m] + 4[s] + 35[a]$	27
mHCADD$^{\mathcal{N}e}_{h_2 \neq 0}$	[12]	Even	$42[m] + 5[s] + 34[a]$	17
HCDBL$^{\mathcal{N}e}_{h_2 \neq 0}$	[12]	Even	$54[m] + 8[s] + 29[a]$	20
HCADD$^{\mathcal{N}e}_{h_2 = 0}$	[12]	Even	$47[m] + 6[s] + 37[a]$	27
mHCADD$^{\mathcal{N}e}_{h_2 = 0}$	[12]	Even	$37[m] + 6[s] + 30[a]$	22
HCDBL$^{\mathcal{N}e}_{h_2 = 0}$	[12]	Even	$40[m] + 6[s] + 27[a]$	16
HCADD$^{\mathcal{A}}$	[18]	Even	$1[i] + 21[m] + 3[s] + 30[a]$	15
HCDBL$^{\mathcal{A}}$	[18]	Even	$1[i] + 9[m] + 6[s] + 24[a]$	10
HCDBL$^{\mathcal{A}}_{deg(h)=1}$	[14]	Even	$1[i] + 5[m] + 9[s] + 10[a]$	7
HCDBL$^{\mathcal{A}}_{deg(h)=1,h_1=1}$	[14]	Even	$1[i] + 5[m] + 9[s] + 7[a]$	6
HCDBL$^{\mathcal{A}}_{deg(h)=2,h_0=0}$	[14]	Even	$1[i] + 17[m] + 5[s] + 31[a]$	10

We summarize our findings in Table 2. Due to space constraints, we are unable to provide all the formulas studied in this work in ERSF. These have been provided in the full version of the paper, which is available on-line. We are not providing the reference to the on-line full version here honouring the blind review process.

Observing Table 2, one can conclude that the formulas presented in [14] are the best for an implementation over binary fields. However, these are based on a special class of curves. An implementation of the formulas in [10], which are more general in nature, needs only one more register in doubling. In the scalar multiplication algorithm, sets of explicit formulas will require 15 registers each. The former has no curve parameter which is not zero or 1. The later requires storing of at most 6 curve parameters. Thus for an efficient implementation the formulas of [10] are suitable. It will require at most 21 registers (at most 6 registers for curve parameters).

For an implementation over fields of odd characteristic, clearly the formulas in [12] are the most suitable. In this representation mixed addition requires 20 registers and doubling requires 16. Besides 2 curve parameters are to be stored. So the scalar multiplication can be computed in 22 registers.

For addition formulas we do not reuse the registers containing the parameters of the base point. As stated above we also experimented reusing all registers. We provide our results in Table 3. It can be seen that number of registers goes down significantly if all registers are reused.

Table 3. Register Requirement: Register Reuse Vs No Reuse

Algorithm	Proposed in	#Registers (all reused)	#Registers (selective reuse)
$\text{HCADD}^{\mathcal{A}}$	[10]	13	15
$\text{HCADD}^{\mathcal{N}o}$	[12]	19	23
$\text{mHCADD}^{\mathcal{N}o}$	[12]	19	20
$\text{HCADD}^{\mathcal{N}e}_{h_2 \neq 0}$	[12]	23	27
$\text{mHCADD}^{\mathcal{N}e}_{h_2 \neq 0}$	[12]	18	20
$\text{HCADD}^{\mathcal{N}e}_{h_2 = 0}$	[12]	23	27
$\text{mHCADD}^{\mathcal{N}e}_{h_2 = 0}$	[12]	19	22
$\text{HCADD}^{\mathcal{A}}$	[18]	14	15

5 Possible Improvements and Conclusion

Although our register minimization technique produces minimum number of registers required for any explicit formula, its output depends upon the nature of the input file. The input file is generally a sequence of three address codes. There is a vast literature in compiler construction studies on efficient methods for converting an arithmetic formula into three address codes. Our parsing program which converted the explicit formulas into three address codes may not be the optimal one. Therefore there may be still some scope for improvement. Besides, the explicit formulas used for finding the minimum register requirements are best known algorithms. In future, researchers may come out with more efficient formulas. Thus the minimum register requirements reported in the current work may not be the best for hyperelliptic curve cryptosystems.

In a memory constrained small device, we may sacrifice a small amount efficiency for efficient memory usage. That is, instead of keeping a memory location occupied with a computed value which will be required much later, we can free the corresponding location to store other intermediate values and recompute the earlier value once again exactly when it is required. For example, suppose in an algorithm at step k a value $x = y \; op \; z$ is computed and used at Steps $k+1$ and $k+k_1$, where k_1 is not small. Also, suppose that at Step $k+k_1$, both y and z are alive. Then if memory is a concern, instead of storing the value of x for k_1 steps, one may prefer to free that memory at Step $k+2$ and recompute x just before the Step $k+k_1$. Thus one saves a memory location for some steps by recomputing one operation. It can be a cheap operation like addition. In this

way one can trade-off memory for some extra operations. In the current work, we have not gone for such optimizations. In an implementation on a small device, this kind of optimization can lead to better utilization of memory.

References

1. H. Cohen and G. Frey (Eds). *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC Press, 2005.
2. G. Bertoni, L. Breveglieri, T. Wollinger and C. Paar. Hyperelliptic Curve Cryptosystem: What is the Best Parallel Hardware Architecture?, In Nadia Nedjah (Ed.), *Embedded Cryptographic Hardware: Design and Security*, Nova Science Publishers, 2004.
3. D. G. Cantor. Computing in the Jacobian of a Hyperelliptic curve. In *Mathematics of Computation*, volume 48, pages 95-101, 1987.
4. A compendium of NP-optimization problems. `http://www.nada.kth.se/~viggo/problemlist/compendium.html`
5. M. R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman, San Francisco, 1979.
6. P. Gaudry and R. Harley Counting Points on Hyperelliptic Curves over Finite Fields. In *ANTS IV*, LNCS 1838, pp 297-312, Berlin, 2000, Springer-Verlag.
7. R. Harley. Fast Arithmetic on Genus 2 Curves. *Available at http://cristal.inria.fr/~harley/hyper,2000.*
8. P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, 726–737, 1990.
9. N. Koblitz. Hyperelliptic Cryptosystems. In *Journal of Cryptology*, 1: pages 139–150, 1989.
10. T. Lange. Efficient Arithmetic on Genus 2 Curves over Finite Fields via Explicit Formulae. Cryptology ePrint Archive, Report 2002/121, 2002. *http://eprint.iacr.org/.*
11. T. Lange. Inversion-free Arithmetic on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/147, 2002. *http://eprint.iacr.org/.*
12. T. Lange. Weighted Coordinates on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/153, 2002. `http://eprint.iacr.org/`.
13. T. Lange. Formulae for Arithmetic on Genus 2 Hyperelliptic Curves In *Applicable Algebra in Engineering, Communication and Computing*, Vol 15(5), pp: 295-328, 2005.
14. T. Lange and M. Stevens. Efficient Doubling on Genus 2 Curves over Binary Fields. In *Selected Area in Cryptography, 2004*, LNCS 3357, pp 170-181, Springer-Verlag, 2004.
15. A. Menezes, Y. Wu, R. Zuccherato. An Elementary Introduction to Hyperelliptic Curves. Technical Report CORR 96-19, University of Waterloo (1996), Canada. Available at http://www.cacr.math.uwaterloo.ca.
16. P. K. Mishra, P. Pal and P. Sarkar. Towards Minimizing Memory Requirement for Implementation of Hyperelliptic Curve Crytosystems, Cryptology ePrint Archive, Report 2006/204, 2006. `http://eprint.iacr.org/2006/204`.
17. P. K. Mishra and P. Sarkar *Parallelizing Explicit Formula for Arithmetic in the Jacobian of Hyperelliptic Curves (Extended Abstract)*, In Asiacrypt 2003, LNCS , pp 91-111, Springer-Verlag, 2003. Full version available at Cryptology ePrint Archive, Report 2003/180, 2003. `http://eprint.iacr.org/`

18. J. Pelzl and T. Wollinger and C. Paar. High Performance Arithmetic for Hyperelliptic Curve Cryptosystems of Genus Two, International Conference on Information Technology: Coding and Computing - ITCC, 2004, Las Vegas, USA.
19. R. Sethi. Complete register allocation problems. *SIAM Journal of Computing*, 4, 226–248, 1975.
20. A. M. Spallek. Kurven vom Geschletch 2 und ihre Anwendung in Public-Key-Kryptosystemen, PhD Thesis, Universität Gesamthochschule, Essen, 1994.

A HECC Formulae in ERSF with Selective Register Reuse

In this section, we will provide some of the most useful formulae for HECC in ERSF. The whole set of formulae in ERSF with minimum register requirement has been provided in the full version of the paper. In these Formulae, we have used selected register reuse strategy (see Section 4), which is most useful in the context of scalar multiplication.

Algorithm HCADD$^{\mathcal{A}}$ of [10]
Curve Constants Used: h_2, h_1, h_0, f_4.
Input Variables: $u_{10}, u_{11}, v_{10}, v_{11}, u_{20}, u_{21}, v_{20}, v_{21}$
Output Variables: up_0, up_1, vp_0, vp_1

Initialization			
1. $R_1 := u_{10}$	2. $R_2 := u_{11}$	3. $R_3 := v_{10}$	4. $R_4 := v_{11}$
5. $R_5 := u_{20}$	6. $R_6 := u_{21}$	7. $R_7 := v_{20}$	8. $R_8 := v_{21}$
9. $R_9 := R_5 - R_1$	10. $R_{10} := R_3 - R_7$	11. $R_{11} := R_4 - R_8$	12. $R_{12} := R_{10} + R_{11}$
13. $R_{13} := R_2 - R_6$	14. $R_{14} := R_2 * R_{13}$	15. $R_{14} := R_{14} + R_9$	16. $R_9 := R_9 * R_{14}$
17. $R_{10} := R_{14} * R_{10}$	18. $R_{15} := R_{13} * R_{13}$	19. $R_{15} := R_{15} * R_1$	20. $R_9 := R_9 + R_{15}$
21. $R_{11} := R_{13} * R_{11}$	22. $R_{13} := R_{14} + R_{13}$	23. $R_{12} := R_{13} * R_{12}$	24. $R_{12} := R_{12} - R_{10}$
25. $R_{13} := 1 + R_2$	26. $R_{13} := R_{11} * R_{13}$	27. $R_{11} := R_1 * R_{11}$	28. $R_{10} := R_{10} - R_{11}$
29. $R_{11} := R_{12} - R_{13}$	30. $R_{12} := R_9 * R_{11}$	31. $R_{11} := R_{11} * R_{11}$	32. $R_{12} := 1/R_{12}$
33. $R_{11} := R_{11} * R_{12}$	34. $R_{12} := R_9 * R_{12}$	35. $R_9 := R_9 * R_{12}$	36. $R_{10} := R_{10} * R_{12}$
37. $R_{12} := R_{10} - R_{13}$	38. $R_{13} := R_{10} - R_2$	39. $R_{14} := h2 * R_9$	40. $R_{12} := R_{12} + R_{14}$
41. $R_{12} := R_{13} * R_{12}$	42. $R_{12} := R_{12} - R_1$	43. $R_{13} := R_6 + R_6$	44. $R_{13} := R_{13} + R_{13}$
45. $R_{13} := R_{13} - f4$	46. $R_{14} := R_{10} + R_{10}$	47. $R_{13} := R_{14} - R_{13}$	48. $R_{14} := h2 * R_9$
49. $R_{13} := R_{13} + R_{14}$	50. $R_{14} := R_9 * R_9$	51. $R_{13} := R_{13} * R_{14}$	52. $R_{13} := R_{13} - R_{14}$
53. $R_{14} := R_5 * R_{10}$	54. $R_{15} := R_8 + R_8$	55. $R_{15} := h1 + R_{15}$	56. $R_9 := R_{15} * R_9$
57. $R_{15} := R_6 + R_{10}$	58. $R_6 := R_6 * R_{10}$	59. $R_5 := R_6 + R_5$	60. $R_6 := R_{12} + R_5$
61. $R_6 := R_6 + R_9$	62. $R_6 := R_6 + R_{13}$	63. $R_9 := R_{15} - R_{13}$	64. $R_{10} := R_{13} * R_9$
65. $R_{12} := h2 * R_{13}$	66. $R_{10} := R_{10} + R_6$	67. $R_5 := R_{10} - R_5$	68. $R_5 := R_5 * R_{11}$
69. $R_5 := R_5 - R_8$	70. $R_5 := R_5 - h1$	71. $R_5 := R_5 + R_{12}$	72. $R_8 := R_6 * R_9$
73. $R_9 := h2 * R_6$	74. $R_8 := R_8 - R_{14}$	75. $R_8 := R_8 * R_{11}$	76. $R_7 := R_8 - R_7$
77. $R_7 := R_7 - h_0$	78. $R_7 := R_7 + R_9$		
Output			
$up_0 := R_6$	$up_1 := R_{13}$	$vp_0 := R_7$	$vp_1 := R_5$

Number of registers used $= 15$

Algorithm HCDBL$^{\mathcal{A}}$ of [10]
Curve Constants: $h_2, h_1, h_0, f4, f_3, f_2$
Input Variables: u_1, u_0, v_1, v_0
Output Variables: up_0, up_1, vp_0, vp_1

Initialization			
1. $R_1 := u_1$	2. $R_2 := u_0$	3. $R_3 := v_1$	4. $R_4 := v_0$
5. $R_5 := f4 * R_1$	6. $R_6 := 2 * R_3$	7. $R_6 := h_1 + R_6$	8. $R_7 := h_2 * R_1$
9. $R_6 := R_6 - R_7$	10. $R_7 := 2 * R_4$	11. $R_7 := h_0 + R_7$	12. $R_8 := h_2 * R_2$
13. $R_7 := R_7 - R_8$	14. $R_8 := R_3 * R_3$	15. $R_9 := R_1 * R_1$	16. $R_5 := R_9 - R_5$
17. $R_9 := f_3 + R_9$	18. $R_5 := 2 * R_5$	19. $R_5 := R_5 + R_9$	20. $R_{10} := 2 * R_2$
21. $R_5 := R_5 - R_{10}$	22. $R_{10} := 2 * R_{10}$	23. $R_9 := R_{10} - R_9$	24. $R_{10} := R_3 * h_2$
25. $R_5 := R_5 - R_{10}$	26. $R_{10} := f4 * R_1$	27. $R_9 := R_9 + R_{10}$	28. $R_{10} := R_3 * h_2$
29. $R_9 := R_9 + R_{10}$	30. $R_9 := R_1 * R_9$	31. $R_9 := R_9 + f_2$	32. $R_8 := R_9 - R_8$
33. $R_9 := R_1 * R_6$	34. $R_{10} := R_7 - R_9$	35. $R_{10} := R_7 * R_{10}$	36. $R_7 := R_7 - R_9$
37. $R_9 := R_6 * R_6$	38. $R_9 := R_2 * R_9$	39. $R_9 := R_9 + R_{10}$	40. $R_{10} := 2 * f4$
41. $R_{10} := R_{10} * R_2$	42. $R_8 := R_8 - R_{10}$	43. $R_{10} := R_3 * h_1$	44. $R_8 := R_8 - R_{10}$
45. $R_{10} := R_4 * h_2$	46. $R_8 := R_8 - R_{10}$	47. $R_{10} := R_7 + R_6$	48. $R_7 := R_8 * R_7$
49. $R_6 := R_5 * R_6$	50. $R_5 := R_8 + R_5$	51. $R_5 := R_{10} * R_5$	52. $R_5 := R_5 - R_7$
53. $R_8 := 1 + R_1$	54. $R_8 := R_6 * R_8$	55. $R_5 := R_5 - R_8$	56. $R_6 := R_2 * R_6$
57. $R_6 := R_7 - R_6$	58. $R_7 := R_9 * R_5$	59. $R_5 := R_5 * R_5$	60. $R_7 := 1/R_7$
61. $R_5 := R_5 * R_7$	62. $R_7 := R_9 * R_7$	63. $R_8 := R_9 * R_7$	64. $R_6 := R_6 * R_7$
65. $R_7 := 2 * R_6$	66. $R_9 := R_8 * R_8$	67. $R_{10} := R_8 * h_2$	68. $R_7 := R_7 + R_{10}$
69. $R_7 := R_7 - R_9$	70. $R_{10} := R_6 - R_1$	71. $R_{10} := h_2 * R_{10}$	72. $R_{11} := 2 * R_3$
73. $R_{10} := R_{10} + R_{11}$	74. $R_{10} := R_{10} + h_1$	75. $R_8 := R_8 * R_{10}$	76. $R_{10} := R_1 + R_6$
77. $R_{10} := R_{10} - R_7$	78. $R_{11} := R_6 * R_6$	79. $R_8 := R_{11} + R_8$	80. $R_{11} := 2 * R_1$
81. $R_{11} := R_{11} - f4$	82. $R_9 := R_9 * R_{11}$	83. $R_8 := R_8 + R_9$	84. $R_1 := R_1 * R_6$
85. $R_1 := R_1 + R_2$	86. $R_2 := R_2 * R_6$	87. $R_6 := R_7 * R_{10}$	88. $R_9 := R_7 * h_2$
89. $R_6 := R_6 + R_8$	90. $R_1 := R_6 - R_1$	91. $R_1 := R_1 * R_5$	92. $R_1 := R_1 - R_3$
93. $R_1 := R_1 - h_1$	94. $R_1 := R_1 + R_9$	95. $R_3 := R_8 * R_{10}$	96. $R_6 := h_2 * R_8$
97. $R_2 := R_3 - R_2$	98. $R_2 := R_2 * R_5$	99. $R_2 := R_2 - R_4$	100. $R_2 := R_2 - h_0$
101. $R_2 := R_2 + R_6$			
Output			
$up_0 := R_8$	$up_1 := R_7$	$vp_0 := R_2$	$vp_1 := R_1$

Number of registers used $= 11$

Algorithm HCADD$^{\mathcal{N}o}$ of [12]
Curve Constants Used: None
Input Variables: $U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}$,
$U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}$
Output Variables: $Up_1, Up_0, Vp_1, Vp_0, Zp_1, Zp_2, zp_1, zp_2$

Initialization			
1. $R_1 := U_{11}$	2. $R_2 := U_{10}$	3. $R_3 := V_{11}$	4. $R_4 := V_{10}$
5. $R_5 := Z_{11}$	6. $R_6 := Z_{12}$	7. $R_7 := z_{11}$	8. $R_8 := U_{21}$
9. $R_9 := U_{20}$	10. $R_{10} := V_{21}$	11. $R_{11} := V_{20}$	12. $R_{12} := Z_{21}$
13. $R_{13} := Z_{22}$	14. $R_{14} := z_{21}$		
15. $R_{15} := R_1 * R_{14}$	16. $R_{16} := R_2 * R_{14}$	17. $R_{17} := R_5 * R_6$	18. $R_{17} := R_7 * R_{17}$
19. $R_{18} := R_{12} * R_{13}$	20. $R_{14} := R_{14} * R_{18}$	21. $R_{18} := R_3 * R_{14}$	22. $R_{14} := R_4 * R_{14}$
23. $R_{19} := R_7 + R_1$	24. $R_{10} := R_{10} * R_{17}$	25. $R_{11} := R_{11} * R_{17}$	26. $R_{17} := R_{18} - R_{10}$
27. $R_{14} := R_{14} - R_{11}$	28. $R_{18} := R_{14} + R_{17}$	29. $R_9 := R_9 * R_7$	30. $R_{16} := R_9 - R_{16}$
31. $R_{20} := R_{16} * R_7$	32. $R_8 := R_8 * R_7$	33. $R_{15} := R_{15} - R_8$	34. $R_{17} := R_{15} * R_{17}$
35. $R_{19} := R_{17} * R_{19}$	36. $R_{17} := R_2 * R_{17}$	37. $R_{21} := R_1 * R_{15}$	38. $R_{20} := R_{21} + R_{20}$
39. $R_{14} := R_{20} * R_{14}$	40. $R_{17} := R_{14} - R_{17}$	41. $R_{21} := R_7 * R_{15}$	42. $R_{21} := R_{20} + R_{21}$
43. $R_{18} := R_{21} * R_{18}$	44. $R_{14} := R_{18} - R_{14}$	45. $R_{14} := R_{14} - R_{19}$	46. $R_{18} := R_{16} * R_{20}$
47. $R_{13} := R_6 * R_{13}$	48. $R_{19} := R_{15} * R_{15}$	49. $R_{19} := R_{19} * R_2$	50. $R_{18} := R_{18} + R_{19}$
51. $R_{12} := R_5 * R_{12}$	52. $R_{19} := R_{12} * R_{12}$	53. $R_{13} := R_{13} * R_{19}$	54. $R_{13} := R_{13} * R_{18}$
55. $R_{20} := R_{14} * R_{19}$	56. $R_{19} := R_{17} * R_{19}$	57. $R_{18} := R_{18} * R_{20}$	58. $R_{17} := R_{17} * R_{20}$
59. $R_{10} := R_{18} * R_{10}$	60. $R_{11} := R_{18} * R_{11}$	61. $R_{18} := R_{14} * R_{14}$	62. $R_{14} := R_{14} * R_{20}$
63. $R_{16} := R_{16} * R_{14}$	64. $R_{21} := R_{12} * R_{13}$	65. $R_{12} := R_{12} * R_{12}$	66. $R_{13} := R_{13} * R_{13}$
67. $R_{22} := R_{15} + R_8$	68. $R_{18} := R_{18} * R_{22}$	69. $R_{22} := R_{17} + R_{17}$	70. $R_{18} := R_{18} - R_{22}$
71. $R_{18} := R_{15} * R_{18}$	72. $R_{22} := R_{17} + R_{14}$	73. $R_{17} := R_{17} * R_9$	74. $R_9 := R_9 + R_8$
75. $R_9 := R_{22} * R_9$	76. $R_9 := R_9 - R_{17}$	77. $R_{11} := R_{17} + R_{11}$	78. $R_{17} := R_{20} * R_{19}$
79. $R_{22} := R_{20} * R_{20}$	80. $R_{11} := R_{22} * R_{11}$	81. $R_{23} := R_{17} + R_{17}$	82. $R_{19} := R_{19} * R_{19}$
83. $R_{18} := R_{19} + R_{18}$	84. $R_{16} := R_{18} + R_{16}$	85. $R_{18} := R_{14} * R_8$	86. $R_8 := R_8 + R_8$
87. $R_9 := R_9 - R_{18}$	88. $R_{17} := R_{18} + R_{17}$	89. $R_8 := R_8 + R_{15}$	90. $R_8 := R_8 * R_{13}$
91. $R_{13} := R_{15} * R_{14}$	92. $R_{13} := R_{23} - R_{13}$	93. $R_{13} := R_{13} - R_{12}$	94. $R_{14} := R_{17} - R_{13}$
95. $R_{15} := R_{14} * R_{13}$	96. $R_9 := R_9 + R_{10}$	97. $R_{10} := R_{10} + R_{10}$	98. $R_{10} := R_{16} + R_{10}$
99. $R_8 := R_{10} + R_8$	100. $R_{10} := R_{14} * R_8$	101. $R_{10} := R_{10} - R_{11}$	102. $R_9 := R_9 - R_8$
103. $R_9 := R_{22} * R_9$	104. $R_9 := R_{15} - R_9$		
Output			
$Up_1 := R_{13}$	$Up_0 := R_8$	$Vp_1 := R_9$	$Vp_0 := R_{10}$
$Zp_1 := R_{20}$	$Zp_2 := R_{21}$	$zp_1 := R_{22}$	$zp_2 := R_{12}$

Number of registers used= 23

Algorithm mHCADD$^{\mathcal{N}o}$ of [12]
Curve Constants Used: $None$
Input Variables: $U_{10}, U_{11}, V_{10}, V_{11}, U_{20}, U_{21}, V_{20}, V_{21}, Z_{21}, Z_{22}, z_{21}, z_{22}$
Output Variables: $Up_0, Up_1, Vp_0, Vp_1, Zp_1, Zp_2, zp_1, zp_2$

Initialization			
1. $R_1 := U_{11}$	2. $R_2 := U_{10}$	3. $R_3 := V_{11}$	4. $R_4 := V_{10}$
5. $R_5 := U_{21}$	6. $R_6 := U_{20}$	7. $R_7 := V_{21}$	8. $R_8 := V_{20}$
9. $R_9 := Z_{21}$	10. $R_{10} := Z_{22}$	11. $R_{11} := z_{21}$	12. $R_{12} := z_{22}$
13. $R_{10} := R_9 * R_{10}$	14. $R_{13} := R_{11} * R_{10}$	15. $R_{14} := R_1 * R_{11}$	16. $R_{14} := R_{14} - R_5$
17. $R_{15} := R_2 * R_{11}$	18. $R_{15} := R_6 - R_{15}$	19. $R_{16} := R_1 * R_{14}$	20. $R_{16} := R_{16} + R_{15}$
21. $R_{15} := R_{15} * R_{16}$	22. $R_{17} := R_{14} * R_{14}$	23. $R_{17} := R_{17} * R_2$	24. $R_{15} := R_{15} + R_{17}$
25. $R_{10} := R_{15} * R_{10}$	26. $R_{17} := R_{10} * R_9$	27. $R_{10} := R_{10} * R_{10}$	28. $R_{18} := R_{17} * R_{17}$
29. $R_4 := R_4 * R_{13}$	30. $R_4 := R_4 - R_8$	31. $R_3 := R_3 * R_{13}$	32. $R_3 := R_3 - R_7$
33. $R_{13} := R_{16} + R_{14}$	34. $R_{16} := R_{16} * R_4$	35. $R_4 := R_4 + R_3$	36. $R_4 := R_{13} * R_4$
37. $R_3 := R_{14} * R_3$	38. $R_4 := R_4 - R_{16}$	39. $R_{13} := 1 + R_1$	40. $R_{13} := R_3 * R_{13}$
41. $R_3 := R_2 * R_3$	42. $R_3 := R_{16} - R_3$	43. $R_4 := R_4 - R_{13}$	44. $R_{13} := R_{15} * R_4$
45. $R_{15} := R_3 * R_{11}$	46. $R_9 := R_4 * R_9$	47. $R_{16} := R_9 * R_9$	48. $R_{19} := R_3 * R_4$
49. $R_8 := R_{13} * R_8$	50. $R_7 := R_{13} * R_7$	51. $R_{11} := R_{19} * R_{11}$	52. $R_{13} := R_4 * R_4$
53. $R_{20} := R_{13} + R_{19}$	54. $R_{19} := R_{19} * R_6$	55. $R_8 := R_{19} + R_8$	56. $R_8 := R_{16} * R_8$
57. $R_1 := R_1 * R_4$	58. $R_1 := R_3 - R_1$	59. $R_3 := R_{14} * R_4$	60. $R_3 := R_{15} - R_3$
61. $R_1 := R_1 * R_3$	62. $R_3 := R_{11} + R_{11}$	63. $R_4 := R_{13} * R_5$	64. $R_{11} := R_4 + R_{11}$
65. $R_{13} := R_{14} * R_{13}$	66. $R_3 := R_3 - R_{13}$	67. $R_3 := R_3 - R_{18}$	68. $R_{11} := R_{11} - R_3$
69. $R_{13} := R_{11} * R_3$	70. $R_6 := R_5 + R_6$	71. $R_6 := R_{20} * R_6$	72. $R_6 := R_6 - R_{19}$
73. $R_4 := R_6 - R_4$	74. $R_1 := R_1 + R_4$	75. $R_4 := R_4 + R_7$	76. $R_6 := R_7 + R_7$
77. $R_2 := R_2 * R_{16}$	78. $R_1 := R_1 - R_2$	79. $R_1 := R_1 + R_6$	80. $R_2 := R_5 + R_5$
81. $R_2 := R_2 + R_{14}$	82. $R_2 := R_2 * R_{10}$	83. $R_1 := R_1 + R_2$	84. $R_2 := R_{11} * R_1$
85. $R_2 := R_2 - R_8$	86. $R_4 := R_4 - R_1$	87. $R_4 := R_{16} * R_4$	88. $R_4 := R_{13} - R_4$
Output			
$Up_1 := R_3$	$Up_0 := R_1$	$Vp_1 := R_4$	$Vp_0 := R_2$
$Zp_1 := R_9$	$Zp_2 := R_{17}$	$zp_1 := R_{16}$	$zp_2 := R_{18}$

Number of registers used $= 20$

Achieving End-to-End Authentication in Intermediary-Enabled Multimedia Delivery Systems

Robert H. Deng and Yanjiang Yang

School of Information Systems
Singapore Management University, Singapore 178902
{robertdeng,yjyang}@smu.edu.sg

Abstract. Considerable research and experiment results in recent years have shown that the server-proxy-user architecture represents an efficient and scalable new paradigm for multimedia content delivery. However, not much effort has been spent on the security issues in such systems. In this paper, we study data authentication in multimedia content delivery, and in particular, we focus on achieving end-to-end authentication from the multimedia server to end users in the server-proxy-user architecture where intermediary proxies transcode multimedia content dynamically. We present a formal model for the end-to-end authentication problem, and propose a basic construction for generic data modality and prove its security. We further extend and tailor our basic technique to authenticate specific multimedia format, JPEG2000 code-streams.

1 Introduction

The introduction of intermediary transcoding proxies between a multimedia server and end users is increasingly becoming accepted as a promising paradigm for multimedia content delivery. In such a server-proxy-user system, as depicted in Figure 1, one or more intelligent intermediary proxies reside along the path from the multimedia server that provides multimedia content to end users who are content consumers. Each proxy serves a group of users and is entrusted by the multimedia server to perform certain transcoding operations upon the content according to the end users' specific capabilities, configurations, or preferences. For example, a proxy automatically downscales a high-quality image to a small "thumbnail" version, in replying to a user request from a handheld device with limited processing and display capabilities. In many cases, multimedia content may pass several proxies en route and undergo multiple transcoding operations before reaching end users.

A major advantage of the intermediary-enabled content delivery system is *scalability*. In a traditional server-user system, the server will inevitably become the system bottleneck as the size of user population increases. The introduction of intermediary proxies between the server and the users solves this problem by amortizing the processing of user requests from a single server to multiple

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 284–300, 2007.

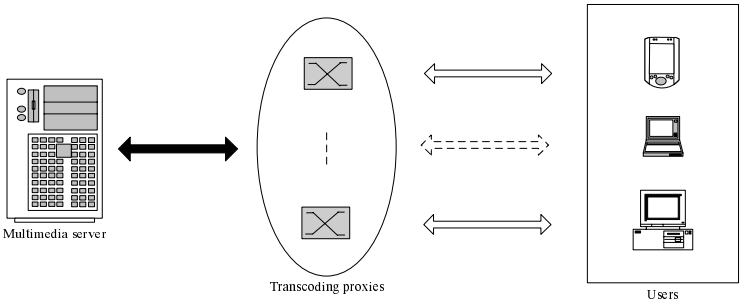


Fig. 1. A server-proxy-user multimedia content delivery system

proxies, which not only relieves the server from intense data processing, but also shortens the request response time due to caching of content at the proxies.

As consumption of multimedia content becomes increasingly a routine in our daily life, the server-proxy-user multimedia delivery systems are expected to find more and more applications. The above mentioned image downgrading scenario belongs to *dynamic content adaption* (e.g., [8,11,19]), where distinct pervasive computing devices require different levels of resolution of the server's multimedia content. Transcoding operations in these applications are typically *content downscaling*, i.e., multimedia streams originated from the server are the highest in quality and the richest in content, and intermediary proxies remove certain data items from the original multimedia streams in order for adapting them to the particular competence of client devices. *Multimedia Composition* (e.g., [14,31]) represent another application scenario, where the intermediary proxies are required to transcode and syndicate multiple content streams from the server (or different servers) into a single code-stream, and send the composite to the end users. Typical transcoding operations in this context involve more than content downscaling, and may also include *content alteration* (i.e., change a part of the original multimedia content). Other commonly used transcoding operations include *Content insertion*. For example, in *tiered multimedia distribution system* (e.g., [28,32]), a tier 1 primary content provider, which acts as the multimedia server, distributes multimedia content to a number of tier 2 affiliating providers, which act as the intermediary proxies; each of the affiliating providers may perform content downscaling, content alteration, and even *content insertion* of data specific to its own user population.

Our Contributions. Multimedia is playing an increasingly important role in sensitive application fields such as government, finance, health care and the law. It is critical and often a requirement to assure end users of the *authenticity* of the content they received in these applications. Our focus in this paper is achieving end-to-end authentication from the multimedia server to end users in the presence of intermediary transcoding proxies. As we have demonstrated above, content transcoding is inevitable in server-proxy-user multimedia delivery systems and such operations would definitely invalidate the original signature

on the content generated by the multimedia server. Therefore, verification of "intactness" and "completeness" of the original content is not the objective of our authentication scheme. Instead, *authentication* in our context refers to verifying "origin of data" as well as "authorization of data alterations".

Our contributions are three-fold. First, we present a formal system model for the end-to-end authentication problem in server-proxy-user multimedia delivery systems. Second, we propose a concrete construction for generic data content based on the Merkle hash tree [16] and the idea of sanitizable signature [1]. The Merkle hash tree allows efficient verification of a subset of data based on a signature over the whole set. A sanitizable signature allows authorized semi-trusted proxies to modify parts of a signed message without interacting with the original signer. The main difference between our construction and the sanitizable signature [1] is that the latter does not consider content downscaling, and the server-proxy-user multimedia delivery systems represent a new application domain. Finally, we apply and optimize the proposed authentication scheme for JPEG2000 code-streams. Our schemes have provable security. We emphasize that the cryptographic primitives proposed in [9,15,27], while not explicitly designed for authentication of multimedia content, are applicable to the server-proxy-user multimedia delivery systems, but can only address content downscaling.

Organization. We review related work in Section 2. In Section 3, we propose a formal system model for the end-to-end authentication issue in the server-proxy-user multimedia delivery systems. Section 4 presents our concrete constructions under the model: we first give a generic construction considering generic data modality, and examine its security; we then tailor the generic scheme to specific multimedia format, JPEG2000 code-streams. Concluding remarks are provided in Section 5.

2 Related Work

For the sake of clarity, we categorize existing work on multimedia authentication into two classes: the first class considers multimedia authentication in server-user systems while the other in server-proxy-user systems, and they have quite different focuses. Ours is clearly related to the latter class.

Multimedia Authentication in Server-User Scenarios. The focus is to efficiently authenticate multicast multimedia streams in the presence of lossy dissemination channels. A number of papers examined solutions to this problem using either secret key-only operations [20,21,23] or asymmetric cryptographic techniques [7,10,13,17,22,25,26]. The basic idea of the secret key-only approach is delayed disclosure of the symmetric keys by the multimedia server to the users so as to provide source authentication. As asymmetric cryptographic operations are expensive, the main objective of the asymmetric cryptography based approach is to achieve high efficiency by amortizing asymmetric operations over several data blocks. For example, reference [26] used reduced-size online/offline k-time signatures, [7] adopted a tree-based method and [17] followed a graph-based

method to reduce authentication overhead; References [10,13,22,25] employed error correcting codes such as erasure coding to strengthen authentication.

These methods essentially deal with authenticating multimedia streams when they suffer from stochastic packet losses during transmission. In other words, data manipulation suffered in these systems is in a uncontrolled manner. In contrast, authentication in server-proxy-user systems has a quite differing scenario to consider: data transcoding is performed in a controlled manner from the perspective of the multimedia server. As a result, the methods for server-user systems are not directly applicable to the server-proxy-user systems.

Multimedia Authentication in Server-Proxy-User Scenarios. Recently, several papers considered authentication of multimedia content in the server-proxy-user systems. References [4,5,6,24,30] proposed using the hash chain or the Merkle hash tree to achieve end-to-end authentication of multimedia content while allowing for content downscaling transcoding. The basic idea is that the multimedia server signs a single piece of authentication data derived from the underlying multimedia content by organizing the content into a chain or a Merkle hash tree, and this signature is still verifiable given a portion of the content together with some auxiliary authentication information related to the missing parts. The methods in [6] considered generic multimedia content by treating the content as a set of data packets, while [4,5,24,30] proposed solutions to specific multimedia modalities such as JPEG2000 images and MPEG-4 video streams. While of high efficiency, these proposals only handle content downscaling and do not address other transcoding operations such as content alteration and content insertion.

An authentication solution for composition (by the intermediary proxies) of MPEG-4 video streams was presented in [14], where the multimedia server signs the original multimedia streams prior to dissemination, and the intermediary proxies en route sign the parts they modified and then combine the previous signatures and the current signature into an aggregate signature [12]. The final resulting aggregate signature enables verification of all the signatures that have been aggregated. A clear advantage of aggregate signature is space efficiency, but it loses *locality*, i.e., it is not possible to pinpoint the parts that invalidate the final aggregate signature given that certain streams experienced illegitimate manipulations en route. In our constructions, our schemes can choose to offer locality to the parts that are expected to be altered.

The work most relates to and motivates ours is [28] that also considered achieving end-to-end authentication of multimedia content while accommodating transcoding operations by the intermediary proxies. The method they employed is that the multimedia server issues a signature upon the hash value of the underlying multimedia content by applying a *trapdoor* hash function, and the proxy knowing the secret trapdoor can find a collision of the hash value. Unfortunately, the trapdoor hash function they use has a serious weakness: publishing different transcoding results of the same multimedia content reveals the secret trapdoor, which means the intermediary proxy can only transcode a content once. This

clearly makes the method of limited use. In contrast, our constructions entitle an intermediary proxy unlimited times of transcoding by utilizing the trapdoor hash function used to generate sanitizable signatures [1]. Moreover, we consider not only generic multimedia data, but also specific multimedia formats such as JPEG2000 code-streams.

3 Formal System Model

Participants in a server-proxy-user multimedia delivery system include a multimedia server, a group of users, and one or several intermediary proxies located between the server and the users. The multimedia server is the originator of the multimedia content, the proxies are authorized by the server to perform certain transcoding operations on the content they receive, and the users are the ultimate consumers of the content. We stress that proxies are semi-trusted to the multimedia server in the sense that they can only transcode designated portions of a content as specified by the server. Informally, authentication in the server-proxy-user multimedia delivery systems is to enable users to verify the origin of the content (i.e., from the server) and to verify that content manipulations have been performed by the authorized intermediary proxies. For simplicity, we assume that there is a single proxy in the system. Generalization to multiple proxies is straightforward.

In our system, the intermediary proxy can perform the following transcoding operations.

- Content downscaling. The multimedia server authorises the intermediary proxy to drop some portions of the content. Normally, content downscaling performed by the proxy is to *harmlessly* downgrade the quality of the multimedia content in order to fit the end user's device. Of course, a malicious proxy or an attacker can always make the content un-renderable to the user's device by dropping the "core" portions of the content (e.g., dropping the R_0 subbase in an JPEG2000 code-stream as we shall see shortly). Such an attack constitutes a type of denial of service attack and is beyond the scope of this paper.
- Content alteration. The multimedia server authorizes the intermediary proxy to modify certain designated portions of the content (e.g., changing the subtitle of a movie from one language to another).
- Content insertion. The multimedia server authorizes the intermediary proxy to insert new content at designated locations of the content (e.g., adding a TV station logo). In our solution, the server inserts blank placeholders at the locations where the intermediary proxy is expected to insert new content; subsequently, the proxy replaces the blank placeholders with the content it wishes to insert. This actually means that content insertion is reduced to content alteration in our solution, and we thus do not explicitly distinguish them in the sequel.

3.1 Model

Our end-to-end authentication scheme for the server-proxy-user multimedia delivery systems consists of four efficient algorithms: key generation, signing, transcoding, and verification.

Key Generation: The key generation algorithm $KeyGen$ is a probabilistic polynomial-time algorithm that takes as input the security parameter 1^k, and outputs two key pairs (PK_S, SK_S) and (PK_P, SK_P) for the multimedia server and the intermediary proxy, respectively.

$$(PK_S, SK_S; PK_P, SK_P) \longleftarrow KeyGen(1^k)$$

where (PK_S, SK_S) is a key pair for the multimedia server to sign the original content, and (PK_P, SK_P) is a key pair used for transcoding by the proxy. The former can be a key pair for any digital signature scheme while the latter can be a key pair for any digital signature scheme or public key encryption scheme. The key generation algorithm may be invoked by a trusted third party (e.g., CA) in a preprocessing step.

Sign: The signing algorithm $Sign$ takes as input a multimedia content m, the private key of the multimedia server SK_S, the public key of the proxy PK_P and random coins r, and outputs a digital signature σ.

$$\sigma \longleftarrow Sign(m, r, PK_P, SK_S)$$

The signing algorithm is performed by the multimedia server to generate a signature on the multimedia content to be disseminated.

Transcode: The trancoding algorithm $Transcode$ takes as input a multimedia content m, the signature σ on m, the random coins r, the public key of the server PK_S, and the private key of the proxy SK_P; it then outputs a transcoded content m', random coins r', and possibly auxiliary authentication information AAI.

$$(m', r', AAI) \longleftarrow Transcode(m, \sigma, r, PK_S, SK_P)$$

The auxiliary authentication information AAI results from content downscaling, and AAI will be nil, i.e., $AAI = \phi$ if there is no content downscaling involved. The transcoding algorithm is performed by the intermediary proxy.

Verify: The verification algorithm $Verify$ is a deterministic algorithm over {0, 1}. It takes as input a multimedia content m, a signature σ on m, the associated random coins r, AAI, the public key of the server PK_S, and the public key of the proxy PK_P, and outputs either 0 or 1.

$$(0, 1) \longleftarrow Verify(m, \sigma, r, AAI, PK_P, PK_S)$$

The verification algorithm can be performed by any user.

3.2 Security Requirements

The above authentication scheme must satisfy the following security requirements.

Correctness. A signature generated by the signing algorithm should be accepted by the verification algorithm.

$$\forall\, \sigma = Sign(m, r, PK_P, SK_S) \Rightarrow$$
$$Verify(m, \sigma, r, AAI = \phi, PK_P, PK_S) = 1$$

Security. Without the knowledge of the private signing key, it is computationally infeasible to generate a valid signature on a message under the corresponding public key, unless the message is a transcoded content from a *legitimate* transcoding. We next give a definition on legitimacy of transcoding.

Legitimate transcoding. Without loss of generality, let us suppose m is a set of n elements, i.e., $m = \{m_1, m_2, ..., m_n\}$. $(m', r', AAI) = Transcode(m, \sigma, r, PK_S, SK_P)$ is a legitimate transcoding of m with respect to σ if one of the following holds

C_1. $m' \subseteq m$; or
C_2. $\sigma = Sign(m', r', PK_P, SK_S)$ in case $AAI = \phi$; or
C_3. $Verify(m', \sigma, r', AAI, PK_P, PK_S) = 1$ in other cases.

We shall provide some exposition on this definition. C_1 states that transcodings that only involve content downscaling (resulting in $m' \subseteq m$) are trivially legitimate. This may seem a little strange but it is justified. To see this, recall that authentication in the context of server-proxy-user systems is to verify data origin and is not to verify data "intactness" or "completeness". We call m' a *trivial transcoded content* of m. C_2 states that if content downscaling is not involved (i.e., $AAI = \phi$), a legitimate transcoding makes the original signature σ the signature on the transcoded content m' and the random coins r'. C_3 covers all other cases, where the only way to determine a legitimate transcoding is to check the verification algorithm. We call m' in C_2 and C_3 a *non-trivial transcoded content* of m. Note that C_3 has already covered C_1 and C_2, but transcodings in C_1, C_2 yield special structures, so they are explicitly listed as separate conditions.

The **security** property is essentially **unforgeability** of digital signatures. We thus formulate the security property as an adversarial game given in the following.

Secure End-to-End Multimedia Authentication Scheme. Algorithms ($KeyGen$, $Sign$, $Transcode$, $Verify$) constitute a secure end-to-end multimedia authentication scheme if no probabilistic polynomial-time adversary $\mathcal{A}$ can win with non-negligible probability (with respect to the security parameter k) the following game:

1. A key pair for signing and a key pair for transcoding are generated:

$$(PK_S, SK_S; PK_P, SK_P) \longleftarrow KeyGen(1^k)$$

2. The adversary $\mathcal{A}$ is given:
   - the public keys PK_S and PK_P as inputs.
   - the first phase of oracle access to $Sign$ algorithm and $Transcode$ algorithm, respectively. That is, $\mathcal{A}$ is free to query $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$.
3. At the end of the first phase of the game, $\mathcal{A}$ outputs a message m and a state which represents the knowledge acquired during this phase.
4. $\mathcal{A}$ continues with the second phase of oracle queries to $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$, under the restrictions that the total number of queries issued to $\mathcal{O}^{SK_S}$ is less than q_s and to $\mathcal{O}^{SK_P}$ is less than q_{tc} during the two phases of oracle accesses.
5. Lastly, $\mathcal{A}$ outputs a signature σ on m, together with the corresponding random r and AAI.

The adversary $\mathcal{A}$ wins the game (i.e., breaking the security of the authentication scheme) if all of the following conditions are met:

- $Verify(m, \sigma, r, AAI, PK_P, PK_S) = 1$.
- m is never queried to $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$.
- m is not a trivial transcoded content of any message queried to $\mathcal{O}^{SK_S}$ and $\mathcal{O}^{SK_P}$ during the game.

The advantage of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game, which is computed over the random coins generated by $\mathcal{A}$.

4 An End-to-End Authentication Scheme

In this section, we present our construction of an end-to-end authentication scheme for multimedia content in server-proxy-user systems. We first briefly review the idea of sanitizable signatures [1], which we rely on to construct our scheme. We then present a construction to achieve end-to-end authentication for generic multimedia content, and analyze its security properties. Finally, we apply and tailor the generic construction to authenticate JPEG2000 code-streams.

4.1 Sanitizable Signatures

We only review the basic notion of sanitizable signature [1] which is sufficient for the understanding of our authentication scheme. To generate a sanitizable signature on a message, a *trapdoor* hash function is first applied to the message, and then the resulting hash value is signed by a digital signature algorithm such as RSA.

An example trapdoor hash function is the following. Let p be a prime such that $p = 2q + 1$, where q is also a prime, and g is a generator of the subgroup of squares of order q. Let $(y = g^x, x)$ be a public and private key pair where x is the private key serving as the secret trapdoor. Let $H()$ be a regular cryptographic hash function (e.g., SHA1). The trapdoor hash function on m under the public key y is defined as $TH_y(m, r) = (\rho - (y^e g^\delta \mod p)) \mod q$, where $r = (\rho, \delta)$ consists of two random numbers (ρ, δ) and $e = H(m, \rho)$.

Given a hash value $v = TH_y(m, r)$, only the party that knows the secret trapdoor x can compute a collision as follows: first choose a random value $k' \in Z_q^*$; then computes $\rho' = (V + (g^{k'} \mod p)) \mod q$, $e' = H(m', \rho')$ and $\delta' = k' - e'x \pmod q$. It is easy to see that $(m', r') = (m', \rho', \delta')$ is a collision of (m, r). Security of the trapdoor hash function can be reduced to the twin Nyberg-Rueppel signature [18]. A desirable feature of this trapdoor hash function is that revelation of multiple collisions does not reveal the secret trapdoor.

We shall use this trapdoor hash function in our below construction. Note that in principle, our construction can use any hash function that can find collisions with the help of certain secret information. An example is the fourth variant of the VSH [3]. However, it seems that the VSH is not as efficient as the above trapdoor hash function, since it uses a large composite as modulus. We also notice that the sibling intractable hash [33] is not suitable to our construction, as it cannot provide unlimited number of collisions, and the collisions must be pre-determined by who that sets up the function.

4.2 Scheme

4.2.1 Basic Scheme

We first present a construction for authenticating generic data modality, i.e., we assume a multimedia content m to be dispatched by the multimedia server is a data set consisting of n items, $m = \{m_1, m_2, ..., m_n\}$.

At a high level, our scheme works as follows. The multimedia server (1) inserts blank items at the places where the intermediary proxy is permitted to insert new content; (2) hashes each of the blank items and the items that the proxy is authorized to alter using the above trapdoor hash function under the public key of the intermediary proxy, and hashes each of the remaining items using a regular hash function; (3) signs the concatenation of the hash values; and distributes all the items along with the signature to end users. Clearly, with the knowledge of its private key, the intermediary proxy is enabled to perform the expected transcoding operations upon the desginated items. In what follows, we provide a more detailed description of the scheme.

Let (PK_S, SK_S) be the key pair of the multimedia server for signing and (PK_P, SK_P) be the key pair of the intermediary proxy for transcoding. Let $\mathcal{H}()$ be a regular hash function (e.g., SHA1) modelled as random oracle [2], and $TH_{PK_P}()$ be the trapdoor hash function under PK_P (SK_P is thus the trapdoor). Without loss of generality, we assume the multimedia server inserts a blank item BL at the end of the data set so as to enable the intermediary proxy to append additional data at the end of the original multimedia content, i.e., $m_{n+1} = BL$. Further we assume that the server expects the proxy to modify l items of the data set, m_{i_1}, m_{i_2}, ..., m_{i_l}.

Sign. During the signing operation, the multimedia server computes

$$\begin{aligned} \sigma &= Sign(m, r, PK_P, SK_S) \\ &= S_{SK_S}(ID_m || PK_P || h_1, ..., h_n, h_{n+1}) \end{aligned}$$

where $S_{SK_S}()$ is the signing function of a digital signature scheme using the private signing key SK_S, ID_m is the identifier of the multimedia content m, $h_i = TH_{PK_P}(ID_m||m_i, r_i)$ for $i \in \{i_1, i_2, ..., i_l, n+1\}$, and $h_i = \mathcal{H}(ID_m||m_i)$ otherwise. The multimedia server then dispatches $\{m_1, m_2, ..., m_{n+1}\}$, σ, and $r = (r_{i_1}, r_{i_2}, ..., r_{i_l}, r_{n+1})$ to the intermediary proxy.

Transcode. Let m'_i denote the altered content of m_i, $i \in \{i_1, i_2, ..., i_l, n+1\}$. As the intermediary proxy knows the private key SK_P that corresponds to PK_P, it can compute r'_i such that $TH_{PK_P}(m'_i, r'_i) = TH_{PK_P}(m_i, r_i)$, $i \in \{i_1, i_2, ..., i_l, n+1\}$. We suppose that the intermediary proxy perform content downscaling by removing m_{j_1}, m_{j_2}, ..., m_{j_t} from the original data set. For this we assume that a removed item $m_j \notin \{m'_{i_1}, m'_{i_2}, ..., m'_{i_l}, m'_{n+1}\}$ for $j \in \{j_1, j_2, ..., j_t\}$. We use m' to denote the transcoded content of m after these transcoding operations. The intermediary proxy computes

$$(m', r', AAI) = Transcode(m, \sigma, r, PK_S, SK_P)$$

where $r' = (r'_{i_1}, r'_{i_2}, ..., r'_{i_l}, r'_{n+1})$, and $AAI = \{h_{j_1}, h_{j_2}, ..., h_{j_t}\}$. The intermediary proxy forwards m', σ, r' together with AAI to end users.

Verify. Upon receiving $\bar{m}$, $\bar{\sigma}$, $\bar{r}$, and $\overline{AAI}$, it is straightforward for a user to execute the verification algorithm $Verify(\bar{m}, \bar{\sigma}, \bar{r}, \overline{AAI}, PK_P, PK_S)$: it first computes and concatenates the hash values in the same manner as in the signing operation. Let $\bar{\mu}$ denote the concatenated value; it then outputs $V_{PK_S}(\bar{\mu}, \bar{\sigma})$, where $V_{PK_S}(.)$ is the verification function of the digital signature scheme using PK_S.

In this basic scheme, the size of the auxiliary authentication information AAI is proportional to the number of the removed items. AAI can be minimized by employing the Merkle hash tree, as we will demonstrate below.

4.2.2 Optimization

In the signing operation, instead of directly signing the concatenation of the hash values, the multimedia server organizes the hash values into a Merkle hash tree [16] as depicted in Figure 2. The leaf nodes are the set of the hash values to be signed; the value of each internal node is derived from its child nodes under a regular hash function. In our case, we can use the same hash function $\mathcal{H}(.)$ used to compute the leaf node values. Referring to Figure 2, we have $h_{12} = \mathcal{H}(h_1, h_2)$, $h_{34} = \mathcal{H}(h_3, h_4)$, and $h_{1234} = \mathcal{H}(h_{12}, h_{34})$. Finally, a unique root value h is obtained. The multimedia server then generates a signature on h as $\sigma = S_{SK_S}(ID_m||PK_P||h)$.

Suppose m_1, m_2, m_3 and m_4 are removed in the subsequent transcoding operation, it suffices for the intermediary proxy to attach $AAI = \{h_{1234}\}$ to the original signature to assist the end user for verification. Comparing $AAI = \{h_{1234}\}$ to $AAI = \{h_1, h_2, h_3, h_4\}$ in the basic scheme, we see a significant reduction in the amount of auxiliary authentication information. While we use an binary tree as an example in Figure 2, the Merkle hash tree is not necessarily binary and balanced.

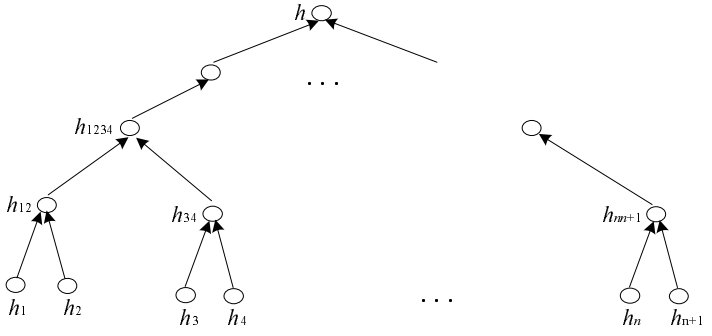


Fig. 2. Organizing hash values into a Merkle hash tree

4.2.3 Security Analysis

We only analyze security of the basic scheme as security of the optimized scheme is evident given the security of the basic scheme. It should be also clear that computing the values of some of the internal nodes using the trapdoor hash function does not compromise the security either.

Our construction trivially satisfies **correctness** requirement, as we employ a digital signature scheme for signing. We thus focus on the **security** requirement. We call a probabilistic polynomial-time adversary a (k, q_s, q_{tc}, t, ε)-breaker of our construction if the adversary makes at most q_s queries to Sign algorithm and at most q_{tc} queries to Transcode algorithm ($q_{h'}$ queries to the oracle of the trapdoor hash function and q_h queries to the oracle for the regular hash function, such that $q_{tc} = q_{h'} + q_h$), runs in at most t steps, and breaks the security property of our construction (instances of size k) at the probability of less than ε. We have the following theorem:

Theorem 1. Let $\mathcal{A}$ be a (k, q_s, q_{tc}, t, ε)-breaker of our construction, then there exist a (k, q_s, t_0, ε_0)-breaker of the underlying digital signature scheme, a (k, q_{tc}, t_1, ε_1)-breaker of either the trapdoor hash function or the (regular) hash function, that satisfy

$$\varepsilon_0 + \varepsilon_1 \geq \varepsilon$$
$$t_0 \leq t + q_{h'} T_{td.hash} + q_h T_{hash}$$
$$t_1 \leq t + q_s T_{sign}$$

where $T_{td.hash}$ is the running time for computing a collision of the trapdoor hash function, T_{hash} is the running time of the regular hash function, and T_{sign} is the running time of the signing function of the digital signature scheme, all on instances of size k. The proof is a slight modification of the proof for the sanitizable signature [1]. For limit of space we do not include it here.

4.2.4 Provision of Locality

The above construction does not provide locality, i.e., in the case that the verification algorithm outputs 0, it is not possible to know which items caused the

verification failure. We can choose to provide locality to the items to be altered, but at the price of extra communication overhead. The method is simple: we need an extra auxiliary authentication information AAI_{ext} that contains the hash values generated by the trapdoor hash function; in the verification algorithm, each received $(\bar{m}_i, \bar{r}_i)$, $i \in \{i_1, i_2, ..., i_l, n+1\}$, is first used to compute a hash value under the trapdoor hash function. The newly computed hash value is then compared with the corresponding hash value in AAI_{ext}; those that do not match indicate the location of authentication failure.

4.2.5 Against Lossy Channels

It appears not difficult to incorporate the techniques against erasure channels such as [25,10,13] into our construction. What deserves mentioning here is that there are two consecutive segments of lossy channels along the multimedia distribution en route in a server-proxy-user system: one from the multimedia server to the intermediary proxy, and the other from the proxy to end users. As a result, measures must be taken at both segments in an attempt to counter the lossy channels. In particular, the multimedia server enforces the adopted anti-lossy channel mechanism first and dispatches the content to the intermediary proxy; upon receiving the content, the proxy decodes and verifies the content, and then re-enforces the mechanism and continues to transmit the content to end users.

4.2.6 Enforcing Data Confidentiality

Data confidentiality is another important security issue in the server-proxy-user multimedia delivery systems. In this context, data confidentiality refers to keeping data hidden from the eavesdroppers en route during transmission. We next describe an encryption technique which integrates naturally into our authentication scheme. We require that the key pair for transcoding owned by the intermediary proxy be a key pair for a public key encryption scheme. Moreover, each user also has a key pair for public key encryption. Instead of sending out the multimedia content in the clear, the multimedia server generates a random session key, encrypts the content using a symmetric encryption scheme under the session key, and encrypts the session key under the public key of the intermediary proxy. Then the server dispatches all the ciphertexts together with its signature σ and r to the intermediary proxy. With the private key for transcoding, the intermediary proxy first decrypts to get the session key, and then decrypts to obtain the multimedia content using the session key. The proxy continues to transode the multimedia content as usual. Finally, the proxy generates a new session key, encrypts the transcoded content using the new session key, and encrypts the session key under the user's public key. At the user side, the user proceeds with decryptions in a similar way as the intermediary proxy.

4.3 Authentication of JPEG2000 Code-Streams

Unlike generic data format, actual multimedia content are rich in structures and have semantics. Security services should exploit this in order to achieve better efficiency. In this subsection, as a case study we extend and tailor the above basic

construction to authenticate JPEG2000 code-streams, based on their respective semantics and data structures.

We first briefly introduce the semantics of JPEG2000, and please refer to [29,4,5] for details. A JPEG2000 image can be devided into rectangular non-overlapping regions each of which is a *tile*. Tiles are compressed independently and it suffices for us to consider a single tile. An image is comprised of one or more color *components* (e.g., red, green). Given a component, a $(n_R - 1)$-level dyadic wavelet transform is performed. The first level of transform wavelet decomposes the component into four frequency subbases, i.e., LL_1 (horizontally lowpass and vertically lowpass), LH_1 (horizontally lowpass and vertically highpass), HL_1 (horizontally highpass and vertically lowpass) and HH_1 (horizontally highpass and vertically highpass). The second level of transform further decomposes LL_1 into another four subbases LL_2, LH_2, HL_2 and HH_2. Eventually, the $(n_R - 1)$ level of transform decomposes LL_{n_R-2} into LL_{n_R-1}, LH_{n_R-1}, HL_{n_R-1} and HH_{n_R-1}. As a consequence, a $(n_R - 1)$-level wavelet transform ends up generating n_R sets of subbases, denoted as $R_0 = LL_{n_R-1}$, $R_1 = \{LH_{n_R-1}, HL_{n_R-1}, HH_{n_R-1}\}$, ..., $R_{n_R-1} = \{LH_1, HL_1, HH_1\}$. We refer R_i as *resolution-increment i*. These n_R resolution-increments corresponds to n_R *resolutions* or image sizes. In particular, the resolution 0 image is constructed from solution-increment 0, i.e., R_0; the resolution 1 image is constructed from R_0 and R_1; and so on. Following the wavelet decomposition, wavelet coefficients are quantized and each quantized subband is partitioned into small rectangular blocks, referred to as *code-blocks*. Each code-block is independently entropy encoded to create a compressed bit-stream which is distributed across n_L quality *layers*. Layers determine the quality or signal-to-noise ratio of the reconstructed image. Let L_0 denote the code-stream data needed to form a layer 0 image, and L_l be the additional code-stream data required to form a layer l image given $L_0, L_1, ..., L_l - 1$, $l \in \{1..n_L - 1\}$. In other words, a layer l image is constructed by $\{L_0, L_1, ..., L_{l-1}, L_l\}$. L_l is referred to as *layer-increment l*. The layer $n_L - 1$ image is the original image given that the total number of layers of the image is n_L. To facilitate accessing certain portions such as ROI of an image, JPEG2000 provides an intermediate space-frequency structure known as *precinct*. A precinct is a collection of spatially contiguous code blocks from all subbases at a particular resolution. Packet is the most fundamental building block in a JPEG2000 code-stream. A packet uniquely corresponds to a layer-increment l, resolution-increment r, component c, and precinct p. Finally, JPEG2000 supports image rendering in four dimension: layer (L), resolution (R), precinct (P), and component (C). They can be "mixed and matched" within a single code-stream. The JPEG2000 standard defines five redering order: LRCP, RLCP, RPCL, PCRL, and CPRL.

We apply the idea of the optimized scheme, i.e., using the Merkle hash tree, to the authentication of JPEG2000 code-streams, and our approach is built upon the authentication schemes in [4,5]. Without losing generality, we consider a code-stream that has one title and one component. As such, the code-stream comprises a set of resolution-increments $\{R_r : r = 0, 1, ..., n_R - 1\}$, a set of

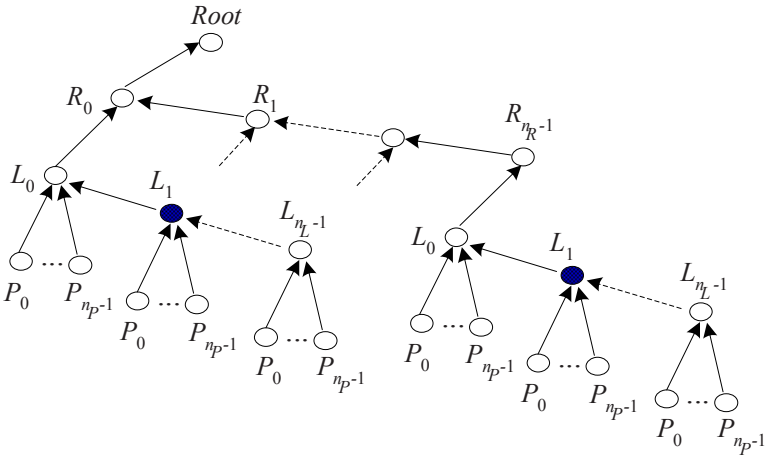


Fig. 3. Organizing the Merkle hash tree following order of resolutions, layers, and precincts

layer-increments $\{L_l : l = 1, 2, ..., n_L - 1\}$, and a set of precincts $\{P_p : p = 1, 2, ..., n_P - 1\}$. From the above, JPEG2000 inherently supports different image sizes and image qualities controlled by resolutions and layers, respectively. This feature provides a natural way to downgrade JPEG2000 images based on resolutions and layers. Here we only consider content downscaling based on resolutions, and generalization to other strategies is straightforward.

One of our main concerns is to minimize the amount of auxiliary authentication information AAI resulting from content downscaling. To achieve this, we should place the nodes that correspond to resolution-increments as high as possible along the Merkle hash tree. Figure 3 shows an example of organizing the Merkle hash tree following the order of resolutions, layers, and precincts. (recall that a packet uniquely corresponds to a resolution, a layer, and a precinct). So the path from the root to a leaf node identifies a packet. For example, the path to the leftmost leaf node P_0 is specified by resolution-increment 0 (R_0), layer-increment 0 (L_0), and precinct 0 (P_0). Thus the value of the leaf node of this path is the hash value of the packet corresponding to R_0, L_0, and P_0. As an example, let us suppose the image is downscaled to resolution 0, then AAI contains only the hash value represented by node R_1. It turns out that the Merkle hash tree in Figure 3 yields the best minimization of the auxiliary authentication information under the resolution-based downscaling strategy.

What remains to consider is how to accommodate transcoding operations of content alteration. This depends on the way the intermediary proxy is expected to modify an image. For example, if the proxy is expected to modify content involving all layers except layer 0, then the hash values corresponding to nodes L_1's (blue nodes in Figure 3) are computed by applying the trapdoor hash function while all other nodes are computed using the regular hash function. This

allows the intermediary proxy to modify any number of the leaf nodes rooted at L_1's while keeping the original hash values of L_1's unchanged (i.e., to find collisions). It is clear that to achieve the same objective, we can instead apply the trapdoor hash function to the leaf nodes of the subtrees rooted at L_1's, and use the regular hash function upon all the remaining nodes. Note however that the trapdoor hash function is much more expensive than the regular hash function, thus the latter alternative is undesirable. For efficiency consideration, it is wise to avoid using the trapdoor hash function upon individual leaf nodes, and it seems that in JPEG2000, content alteration often affects the data at higher levels such as resolution and layer. As such, a main distinction in the authentication of JPEG2000 code-streams from the earlier generic construction is that trapdoor hash function is applied to higher level nodes, and this clearly does not compromise security if the trapdoor hash function is secure.

5 Conclusions

As multimedia is becoming pervasive and playing an increasingly important part in sensitive applications, it is often a requirement to assure end users of the authenticity of the content they received. The main challenge in achieving end-to-end authentication in the server-proxy-user multimedia delivery systems is that authorized transcoding operations performed by intermediary proxies will definitely invalidate the digital signature generated by the multimedia server if a standard use of digital signature is assumed.

In this paper, we first introduced a formal model for the end-to-end authentication problem in the server-proxy-user multimedia delivery setting. We then proposed an authentication scheme for generic data content based on the sanitizable signature and the Merkel hash tree, and formally proved its security. Finally, we applied and tailored our scheme for the authentication of JPEG2000 code-streams.

References

1. G. Ateniese, D. H. Chou, B Medeiros, G. Tsudik. Sanitizable Signatures. *European Symposium on Research in Computer Security, ESORICS'05*, pp. 159-177, 2005.
2. M. Bellare, P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *ACM. Computer and Communication Security*, pp. 62-73, 1993.
3. S. Contini, A. K. Lenstra, and R. Steinfeld. VSH, an Efficient and Provable Collision Resistant Hash Function, *Advances in Cryptology, EUROCRYPT'06*, pp. 165-182, Springer Verlag, 2006.
4. R. H. Deng, D. Ma, W. Shao, Y. Wu. Scalable Trusted Online Dissemination of JPEG2000 Images, *ACM Multimedia Systems*, Vol. 11(1), pp. 60-67, 2005.
5. R. H. Deng, Y. Wu, D. M. Securing JPEG2000 code-streams, *Security in the 21st Century*, Springer Verlag, 2004.
6. C. Gentry, A. Hevia, R. Jain. End-to-End Security in the Presence of Intelligent Data Adapting Proxies: The Case of Authenticating Transcoded Streaming Media, *IEEE Journal on Selected Areas in Communications*, Vol 23(2), pp. 464-473, 2005.

7. P. Golle, N. Modadugu, Authenticated Streamed Data in the Presence of Random Packet Loss, *Network and Distributed System Security Symposium, NDSS'01*, 2001.
8. J. L. Huang, M. S. Chen, H. P. Hung. A QoS-aware Transcoding Proxy Using On-demand Data Broadcasting, *IEEE International Conference on Computer Communications, INFOCOMM'04*, 2004.
9. R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic Signature Schemes *CT-RSA*, LNCS 2271, pp. 244-262, Springer Verlag, 2002.
10. M. N. Krohn, M. J. Freedman, D. Mazires. One-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution, *IEEE Symposium on Research in Security and Privacy, S&P'04*, pp. 226-240, 2004.
11. M. Libsie, H. Kosch. Content Adaptation of Multimedia Delivery and Indexing Using MPEG-7, *ACM Multimedia, MM'02*, pp. 644-646, 2002.
12. A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham. Sequential Aggregate Signatures from Trapdoor Permutations, Prof. Advances in Cryptology, EUROCRYPT'04, pp. 74-90, Springer Verlag, 2004.
13. A. Lysyanskaya, R. Tamassia, N. Trandopoulos. Multicast Authentication in Fully Adversarial Networks, *IEEE Symposium on Research in Security and Privacy, S&P'04*, pp. 241-255, 2004.
14. T. Li, H. Zhu, Y. Wu. Multi-Source Stream Authentication Framework in case of Composite MPEG-4 Stream, *International Conference on Information and Communications Security, ICICS'05*, LNCS 3783, pp. 389-401, Springer Verlag, 2005.
15. K. Miyazaki, G. Hanaoka, and H. Imai. Digitally Signed Document Sanitizing Scheme Based on Bilinear Maps, *ACM Symposium on Information, Computer and Communications Security, ASIACCS'06*, pp. 343-354, 2006.
16. R. C. Merkle. A Certified Digital Signature, *Advances in Cryptology, CRYPTO'89*, LNCS 0435, pp. 218-238, Springer Verlag, 1989.
17. S. Miner, J. Staddon. Graphy-based Authentication of Digital Streams, *EEE Symposium on Research in Security and Privacy, S&P'01*, pp. 232-246, 2001.
18. D. Naccache, D. Pointcheval, J. Stern. Twin Signatures: An Alternative to the Hash-and-Sign Paradigm. *ACM Conference on Computer and Communications Seciryt, CCS'01*, pp. 20-27, 2001.
19. W. T. Ooi, R. van Renesse. Distributing Media Transformation Over Multiple Media Gateways, *ACM Multimedia, MM'01*, pp. 159-168, 2001.
20. A. Perrig. The BiBa One-Time Signature and Broadcast Authentication Protocol, *ACM Conference on Computer and Communication Security, CCS'01*, pp. 28-37, 2001.
21. A. Perrig, R. Canetti, J. D. Tygar, D. Song. Efficient Authentication and Signing of Multicast Streams Over Lossy Channels, *IEEE Symposium on Research in Security and Privacy, S&P'00*, pp. 56-73, 2000.
22. J. M. Park, E. Chong, H. J. Siegel. Efficient Multicast Packet Authentication Using Erasure Codes, ACM Transactions on Information and Security Security, Vol 6(2), pp. 258-285, 2003.
23. A. Perrig, R. Canetti, D. Song, J. D. Tygar. Efficient and Secure Source Authentication for Multicast, *Network and Distributed System Security Symposium, NDSS'01*, 2001.
24. C. Peng, R. H. Deng, Y. Wu, W. Shao. A flexible and scalable authentication scheme for JPEG2000 image codestreams, *ACM Multimedia, MM'03*, pp. 433-441, 2003.
25. A. Pannetrat, R. Molva. Efficient Multicast Packet Authentication, *Network and Distributed System Security Symposium, NDSS'03*, 2003.

26. P. Rohatgi. A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication, *ACM Conference on Computer and Communication Security, CCS'99*, pp. 93-100, 1999.
27. R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. *International Conference on Information Security and Cryptology, ICISC'01*, pp. 285-304, 2001.
28. T. Suzuki, Z. Ramzan, H. Fujimoto, C. Gentry, T. Nakayam, R. Jain. A System for End-to-End Authentication of Adaptive Multimedia Content, *IFIP Conference on Communications and Multimedia Security CMS'04*, 2004.
29. D. S. Taubman, M. W. Marcellin. *JPEG2000 - Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, 2000.
30. Y. Wu, R. H. Deng. Scalable Authentication of MPEG-4 Streams, IEEE Trans. on Multimedia, Vol.8(1), pp.152-161, 2006.
31. M. Wagner, W. Kellerer. Web Services Selection for Distributed Composition of Multimedia Content, *ACM Multimedia, MM'04*, pp. 104-107, 2004.
32. T. Yuuichi, T. Kaori, O. Takeshi, S. Shinji, M. Hideo. ASIA: Information Sharing System with Derived Content Restriction Management, *IEICE Transactions on Communications*, Vol 428, pp. 1463-1475, 2003.
33. Y. Zheng, T. Hardjono, and J. Pieprzyk. Sibling Intractable Function Families and Their Applications, *Advances in Cryptology, ASIACRYPT'91*, LNCS 739, pp. 124-138, Springer Verlag, 1991.

Scalable Group Key Management Protocol Based on Key Material Transmitting Tree

Minghui Zheng[1,2], Guohua Cui[1], Muxiang Yang[1], and Jun Li[1]

[1] College of Computer Science, Huazhong University of Science & Technology, Wuhan, Hubei, 430074, China
[2] Department of Computer Science, Hubei institute for Nationalities, Enshi, Hubei, 445000, China
mhzheng@smail.hust.edu.cn

Abstract. The group key management is one of the most crucial problems in group communication. In dynamic and large-scale groups, the overhead of key generating and key updating is usually relevant to the group size, which becomes a performance bottleneck in achieving scalability. Therefore, scalable group key management protocol, which is independent from group size, is the basis for wide applications of group communication. The paper proposes a novel group key management protocol, which designates un-trusted routers over Internet as transmitting nodes to organize a key material transmitting tree for transmitting key material. Members in group that are partitioned into subgroups attach to different transmitting nodes, and compute SEK using received key material and own secret parameter. The overhead of key management can be shared by the transmitting nodes which can not reveal the data of group communications, and the overhead for key management of each transmitting node is independent of the group size. In addition, the new protocol conduces to constant computation and communication overhead during key updating.

Keywords: group communication, group key management, scalability, key material, security.

1 Introduction

Multicasting is an efficient communication approach based on UDP/IP protocols [1] for group-oriented applications, such as video conferencing, multi-player games, video on demand (VoD), TV over Internet, e-learning, collaboration workspaces, database replication and broadcasting stock quotes. Nevertheless, multicast addresses are public known and any node of the network can join or leave the group freely using the Internet Group Management Protocol (IGMP) [2]. This simplicity which makes the strength of multicast routing, presents however, many vulnerabilities. Therefore, security mechanisms such as authentication, access control, integrity verification and confidentiality are required in order to provide secure group communications. Most of these mechanisms rely generally on encryption using one or several keys. The management of these keys, which included generating, distributing and updating the

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 301–313, 2007.

keys, constitutes then a basic block to build secure group communication applications. This paper focuses on group key management used in assuring group communication confidentiality. For achieving confidentiality, a Session Encryption Key (SEK), which is used to encrypt/decrypt the data of group communication, is shared by all legitimate members in group. Because of dynamic membership, the SEK must be updated to ensure that the new member cannot decrypt data before it joins the group, namely, backward security. The SEK must also be updated to ensure that the former member cannot decrypt data after it leaves the group, namely, forward security [2]. For a large group with a highly dynamic membership, the cost of key management could be quite substantial, and therefore scalability of group key management protocols becomes an important issue that must be addressed.

This paper proposes a novel group key management protocol based on key material transmitting tree. At first, we construct a key material transmitting tree using a trusted Key Generation Center (KGC) as the root node, and designated routers over Internet as inner nodes first. Members in communications group are divided into many sub-groups which attach to the inner nodes in the tree respectively. The KGC transmits key material, which is used to generate the SEK, to each member via key material transmitting tree. Ultimately, the SEK can only be worked out by legitimate group members who use key material and secret parameter held by it. The proposed protocol possesses the following characteristics as well as the basic security requirements of group key management protocol: (1) it is applicable to the large dynamic group because of better scalability; (2) key material transmitting nodes share the load of key management with the KGC. No matter how large of the group size is, both communication and computation cost of KGC (or material transmitting nodes) are constant; (3) when updating SEK, we only need to reconfigure system parameters in a small range; (4) key material transmitting nodes in the tree do not need to be trusted in process of transmitting key material securely.

The remainder of this paper is organized as follows: we review related work about group key management protocols and classify these protocols into three classes in Section 2. Section 3 describes a novel group key management protocol in details. Section 4 discusses security and scalability of proposed protocol. Comparison of performance with typical previous protocols is presented in Section 5. Finally, in Section 6 we conclude.

2 Related Work

The existing group key management protocols can be divided into three main classes according to approach of generating SEK [3-5]: *centralized*, *decentralized* and *distributed*. Depending on the technique used to distributed the SEK, centralized protocols are further classified into two subcategories: flat key management [6, 7, 11] and hierarchical key management [8-10]. In terms of object of driving re-keying, decentralized protocols are further divided into two subcategories: membership-driven re-keying protocols [12, 15, 16] and time-driven re-keying protocols [13, 14, 17]. Depending on the virtual topology created by the members for cooperation, we further classify distributed protocols into three subcategories: ring-based cooperation [18], hierarchical cooperation [19-21, 24, 25] and broadcast cooperation [22, 23]. Fig. 1

illustrates this classification. The characteristic of distributed protocols is that group members cooperate to establish SEK by key agreement. This improves the reliability of the overall system and reduces the bottlenecks in the network in comparison with centralized protocols and decentralized protocols. But distributed protocols are difficult to be used in large dynamic group with frequent change of membership, due to the computation cost and delay. So this kind of protocols will not be discussed in this paper.

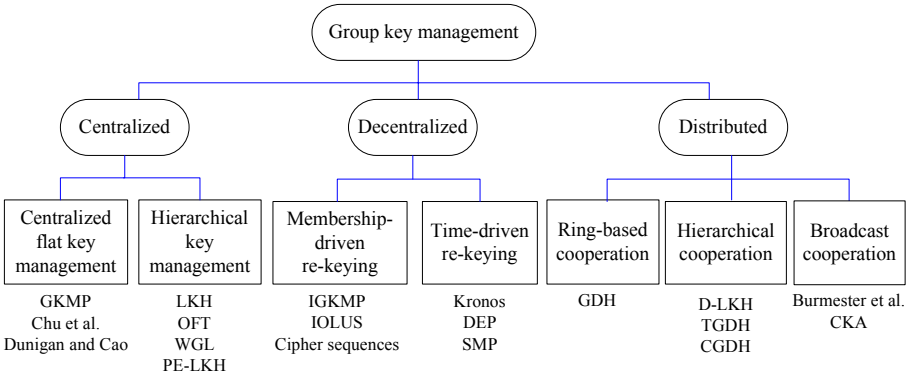


Fig. 1. Taxonomy of Group Key Management Protocols

In centralized protocols, a single key server is responsible for computing and distributing the SEK. Using this approach, Harney and Muckenhirn [6] proposed a centralized flat protocol: the Group Key Management Protocol (GKMP). In GKMP, the key server shares a Key Encryption Key (KEK) with each valid group member in order to establish a secure channel to distribute the current SEK. The key server generates a Group Key Packet (GKP) that contains two keys: a Group SEK (GSEK) and a Group KEK (GKEK). The GSEK is used to encrypt data and the GKEK is used to securely distribute a new GKP whenever required. When a new member joins the session, the key server generates a new GKP (which contains a new GSEK to assure backward security) and sends it securely to the new member encrypted with the KEK established with this new member, and broadcasts it to other members encrypted with the old GSEK. The key server refreshes the GKP periodically and uses the GKEK for its distribution to the group members. When a member leaves, the key server generates a new GKP and sends it to each remained member encrypted with the KEK that it shares with each member. Thus to assure forward security, the number of required messages for re-keying in GKMP are $O(n)$, where n is the size of group. Hao-hua Chu et al. [7], Dunigan and Cao [11] also proposed similar group key management protocols that suffer from the same shortcoming liked GKMP. It is clear that these solutions do not scale to large groups with highly dynamic members.

In order to reduce the number of re-keying messages, a Logical Key Hierarchy (LKH) tree protocol and a Key Graphs protocol using key hierarchy method are presented in [8] and [9] respectively. They organize a set of cryptographic keys into a tree structure. The root node of the tree is the SEK shared by all group members, and

the other nodes are KEKs assist in distribution of the SEK. In LKH and Key Graphs, the number of storing keys in the key server is $O(n)$, and the number of re-keying messages reduce to $O(\log n)$ when re-keying. But the number of storing keys of each group member increase to $O(\log n)$. Balenson et al. proposed an improvement over LKH called One-Way Function Tree (OFT) protocol in [10]. In OFT, the KEK is calculated by members rather than attributed by the key server. This protocol allows reducing the number of re-keying messages from $2\log n$ to $\log n$. Nevertheless, when the size of group becomes larger, the key server has to store and maintain a huge keys tree, which could become the bottleneck of performance in these schemes.

The characteristic of the decentralized group key management protocol is that the whole communication group is divided into many sub-groups for achieving scalability, and each sub-group has its individual SEK and manager. So, if some sub-group members change, only does the SEK of the sub-group need to be updated, and which will not affect other sub-groups. IOLUS [12] protocol partitioned group into sub-groups, each sub-group has a group security agent (GSA) and all GSAs form a new group which is managed by group security controller (GSC). In IOLUS, each sub-group possesses individual SEK. The multicast packets from sub-group B to C need to be decrypted using B's SEK, and then encrypted using C's SEK. Scalability is achieved by making each sub-group relatively independent and thus re-keying due to group membership change can be confined to the respective sub-groups. So IOLUS avoids the "1-affects-*n*" problem which is existed in the centralized protocols. However, the system performance is affected by the delay from the GSA decryption and re-encryption of every data packet. Additionally, the SEK of sub-group is generated by GSA, so the trust in third party GSA is also another complicated problem. Dondeti et al. proposed a similar DEP protocol in [13]. Compared with IOLUS, DEP uses the technique of re-encryption to protect the multicast packets, and the transmitting node (similar with GSA in IOLUS) can not obtain content of packet, so the trust in third party can be solved. But DEP also has some drawbacks, such as the large cost of computing and delay of transmitting. Yang et al. presented SMP protocol in [14], it does not need to trust the third parties (namely, multicast routers), but the cost of generating secret keys for multicast packet is a significant overhead, which affects the system performance. The group key management protocol proposed in this paper is a centralized protocol adopting sub-group technology.

3 Proposed Protocol

In this section we describe the group key management protocol based on key material transmitting tree for scalable secure multicasting. First, we construct a key material transmitting tree for key distribution, and assign parameters to each entity in the tree. Then we describe the procedure of SEK generating and re-keying. The following notation is used throughout the remainder of this paper.

- N_0 KGC
- N_i Key material transmitting node, where $i \geq 1$
- U_i The set of members (namely sub-group) attached to N_i
- A One legitimate group member in U_i
- k SEK

- k_i Key-deriving parameter to be assigned to members in U_i
- x_i Key material transmitting parameter of N_i
- r_i Secret parameter of N_i
- Z_p^* Cyclic algebraic group of prime order p
- g Exponentiation base; generator in the group Z_p^*
- R Key material of generating k.

3.1 Key Material Transmitting Tree

In proposed protocol, key distribution operates in a key material transmitting tree and Fig. 2 illustrates an abstract model of it. The root node is the KGC, which is responsible for generating and re-keying. The KGC may be the multicast group creator, one of the group members or a trusted third party. Inner nodes (or key material transmitting nodes) are either router or hosts over Internet that can handle the workload of managing a subgroup of the multicast group. Members in multicast group are divided into many sub-groups which attach to the inner nodes in the tree respectively, namely, these sub-groups act as leaf nodes. In addition, each node in the key material transmitting tree is associated with parameters. Procedure of configuring these parameters will be described in the following subsection.

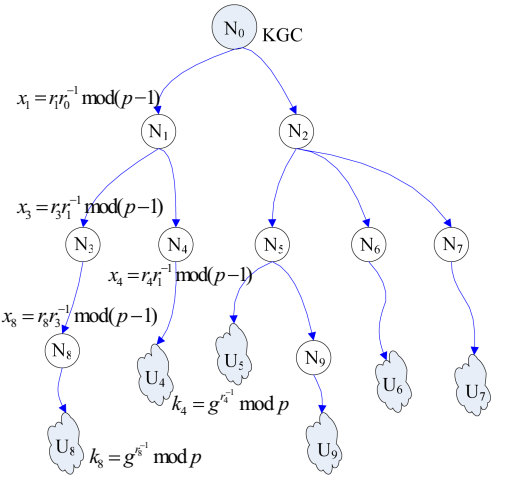


Fig. 2. Key Material Transmitting Tree

3.2 System Initialization

In system initialization phase, The KGC first configures parameters for each node in the tree. Fig. 2 shows an example of the key material transmitting tree with part of configuration parameters. The procedure for assigning parameters is described in details as follows:

Step1. chooses a large secure prime p, and g is generator of cyclic group Z_p^*.
Step2. chooses secret parameter $r_i \in Z_{p-1}^*$ for N_i randomly, $i \in [0, m]$.

Step3. computes $x_i = r_i \cdot r_l^{-1} \bmod(p-1)$ for node N_i, and sends x_i to N_i, where $i \in [1,m]$, r_l is the secret parameter of N_l which is the children node of N_i.
Step4. computes $k_i = g^{r_i^{-1}} \bmod p$ for members in U_i.
Step5. publishes p and g.

Notice that the secret parameter r_i for N_i is not sent to N_i but only kept in secret by KGC for defeating collusion attack.

3.3 SEK Generation

In the SEK generation phase, the KGC establishes a SEK for secure group communications. Instead of transmitting the determined SEK in the tree, only the parameters (namely, key material) for computing SEK are delivered. Along the path from the KGC to legitimate group members, each key transmitting node performs a transformation on the received key material and forwards the result to its children nodes and subgroup members. Finally, every legitimate group member computes the SEK by using key-deriving parameter and the key material received from the parent transmitting node. The SEK generation phase is described as follows:

Step1. KGC chooses s randomly, satisfying $\gcd(s, p-1) \neq 1$, and computes $k = g^s \bmod p$ as SEK.
Step2. KGC computes $R = s \cdot r_0 \bmod(p-1)$, sends R to its children nodes.
Step3. Updating and forwarding R. After receiving R from KGC or the parent transmitting node, N_i computes $R = R \cdot x_i \bmod(p-1)$, then sends the new R to its children transmitting nodes and members in U_i.
Step4. Repeat step 3 until all leaf nodes receive R.
Step5. After receiving R, the member A computes SEK by the following function: $k = (k_i)^R \bmod p$.

Theorem 1. The legitimate group member A in U_i can derive the SEK by using the key material R which is received from its parent node and the key-deriving parameter k_i which is held by himself as inputs to the function $k = (k_i)^R \bmod p$.

Proof: Assume that there are h key transmitting nodes along the path from the KGC to the group member A in U_i. Let $N_1^{'}$, $N_2^{'}, \ldots, N_h^{'}$ denote these h transmitting nodes. Notes that $N_h^{'} = N_i$. We compute the key material R received by A.

$$\begin{aligned} R &= s \cdot r_0 \bmod(p-1) \cdot \prod_{j=1}^{h} x_j \bmod(p-1) \\ &= s \cdot r_0 \cdot \prod_{j=1}^{h} (r_j r_{j-1}^{-1}) \bmod(p-1) \\ &= (s \cdot r_0 \cdot r_0^{-1} r_h \cdot \prod_{j=1}^{h-1} (r_j r_j^{-1})) \bmod(p-1) \\ &= s \cdot r_h \bmod(p-1) \end{aligned}$$

Since $N_h^{'} = N_i$, then $r_i = r_h$. According to Euler's generalization of Fermat's little theorem, we have

$$\begin{aligned} k &= (k_i)^R \bmod p \\ &= (g^{r_i^{-1}})^{s \cdot r_h \bmod(p-1)} \bmod p \end{aligned}$$

$$= g^{r_i^{-1} \cdot r_h \cdot s} \bmod p = g^{r_i^{-1} \cdot r_i \cdot s} \bmod p$$

$$= g^{s} \bmod p$$

From discussion above, each legitimate member can derive the SEK by itself. □

In the practical applications, the group members need to verify the consistency of SEKs generated by the KGC and himself respectively. At first, KGC computes $H(k)$, where $H(\ )$ is a secure hash function, then signs the $H(k)$ using its private key, and forwards the signature to members along with the key material R. After receiving the signature, group member verifies the signature using the KGC's public key which is obtained from the Public Key Infrastructure (PKI), and compares $H(k)$ with the value of hashing SEK derived by himself. If they are the same, it means that message resource is KGC, KGC and group member generated SEKs consistently.

3.4 Join Event

This subsection presents the procedure for dealing with the group member join event. When a member A wants to join the communication group, it must send the registration request to the KGC at first. If the request is granted, the KGC replies with a message containing A's key-deriving parameter and the address of a key material transmitting node which KGC allocates member A to. Then the KGC reconfigures parameters of the key material transmitting tree in a small range. Let N_i denote the key material transmitting node which assigned to the new member A. In the procedure of reconfiguration, the KGC only modified the parameters that are relevant to the nodes N_i and its children nodes, as illustrated in Fig. 3. The procedure is described in details as follows.

Step1. KGC replaces r_i with $r_i^{'} \in Z_{p-1}^{*}$, replaces x_i with $x_i^{'} = r_i^{'} \cdot r_{i-1}^{-1} \bmod(p-1)$.
Step2. for any node N_j which is the children of N_i, KGC replaces x_j with $x_j^{'} = r_j \cdot r_i^{'-1} \bmod(p-1)$.
Step3. KGC replaces k_i with $k_i^{'} = g^{r_i^{'-1}} \bmod p$, and sends it to A securely.
Step4. replaces old members' k_i in U_i with $k_i^{'}$.
Step5. go to SEK generation phase in Subsection 3.3 to renew SEK.

In the Step4, in order to reduce the communications load of KGC and guarantee security of key-deriving parameter $k_i^{'}$ (i.e. $k_i^{'}$ can't be obtained by key material transmitting nodes), the KGC computes $y = g^{r_i^{'-1} - r_i^{-1}} \bmod p$, and sends y to the node N_i but not to each member in U_i directly. After receiving y, N_i sends it to all local members securely. This can be achieved by using unicast secure communications or other secure group communications for a small multicast group. The old group members in U_i can derive the new key-deriving parameter $k_i^{'}$ using the old parameter k_i and y by the following expression.

$$k_i^{'} = k_i \cdot g^{r_i^{'-1} - r_i^{-1}} \bmod p$$

$$= g^{r_i^{-1}} \cdot g^{r_i^{'-1} - r_i^{-1}} \bmod p$$

$$= g^{r_i^{'-1}} \bmod p$$

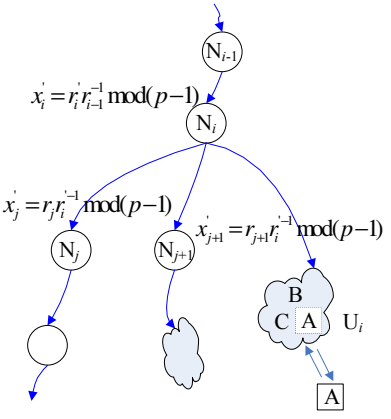


Fig. 3. Reconfiguration of Tree for Member Join/Leave

3.5 Leave Event

In general, leave event occur under two conditions: (1) a group member wishes to voluntarily leave the communication group in which case it sends a LEAVE request to the KGC, or (2) the KGC expels a group member and sends a notification to that effect to the expelled member. If member A leaves from U_i, the SEK must be updated for the sake of forward security. The procedure of leave event is described in details as follows.

Step1. KGC replaces r_i with $r_i^{'} \in Z_{p-1}^{*}$, replaces x_i with $x_i^{'} = r_i^{'} \cdot r_{i-1}^{-1} \bmod(p-1)$.
Step2. for each children node N_j of N_i, KGC replaces x_j with $x_j^{'} = r_j \cdot r_i^{'-1} \bmod(p-1)$.
Step3. replaces remained members' k_i in U_i with $k_i^{'} = g^{r_i^{'-1}} \bmod p$.
Step4. go to SEK generation phase in Subsection 3.3 to renew SEK.

The Step3 for leave event is very similar to the step 4 for join event. There is one slight difference: After receiving y, N_i forwards it to all remained local members securely only via unicast communications.

4 Analysis of Proposed Protocol

We have presented a scalable group key management protocol for the purpose of supporting large and dynamic group communications over Internet. In this section, we will analysis the security and the scalability of proposed protocol.

4.1 Security Analysis

In the following, we show how the proposed group key management protocol meets the security requirements of secure-group communications at first.

(1) Data confidentiality

Data confidentiality means that the non-group members can't obtain current communication data. The proposed group key management protocol guarantees that

outsiders and intermediate key material transmitting nodes (i.e. multicast routers) are unable to access the communication data. To acquire the communication data, the adversary must either hold the SEK or perform brute force attack on the encrypted multicast packets. Brute force attacks can be easily defeated by using cryptographic algorithms with appropriate key length. According to the formula $k = (k_i)^R \bmod p$, we know that obtaining the SEK requires key material R and key-deriving parameter k_i. Even though the adversary can eavesdrop key material R, it can't obtain the key-deriving parameter k_i held by legitimate group members only. Therefore, the adversary can't compute the SEK. Our protocol meets the requirement of data confidentiality.

(2) Attack of compromising secret parameter

Here, we consider some possible attacks on the proposed protocol that an adversary might attempt to compromise secret parameters. In the system initialization phase, KGC chooses a secret parameter $r_i \in Z_p^*$ randomly for each key material transmitting node N_i, and these secret parameters are known by the KGC only. A former member who held key-deriving parameter k_i might try to derive the secret parameter r_i from the equation $k_i = g^{r_i^{-1}} \bmod p$. However, it requires the knowledge of $r_i^{-1} (\bmod p-1)$, which is protected by the Discrete Logarithm Problem (DLP) by the equation. As a result, the former member cannot derive the secret parameter r_i of node N_i. In addition, an adversary attempts to collect all x_i on path from the KGC to some member, for the purpose of obtaining secret parameters of transmitting nodes in this path. An example of this attack is illustrated on the path from the root node N_0 to the node N_8 in Fig. 2. Adversary can construct the following simultaneous linear congruence system:

$$\begin{cases} x_1 = r_1 r_0^{-1} \bmod(p-1) \\ x_3 = r_3 r_1^{-1} \bmod(p-1) \\ x_8 = r_8 r_3^{-1} \bmod(p-1) \end{cases}$$

In linear algebra, although the knowledge of x_1, x_3 and x_8 can be obtained by eavesdropping, the adversary cannot solve the value of four secret parameters r_0, r_1, r_3 and r_8, according to the congruence system above. Therefore, the adversary cannot compromise secret parameter.

(3) Backward and forward security

When a member joins or leaves the subgroup U_i which attach to key material transmitting node N_i, the secret parameter k_i held by each member in U_i is updated as k_i'. The new key-deriving parameter k_i' cannot be used for generating either the old key-deriving parameter or the old session encryption key. As a consequence, historical communication data are not accessible to new members (backward security). Similarly, a former member with an old key-deriving parameter cannot derive the session encryption key that is used to encrypt and decrypt future data (forward security).

4.2 Analysis of Scalability

(1) Processing scalability

The number of processing of each key material transmitting node to cope with key management is independent of the group size. First, in our protocol, when forwarding

the key material R, the size of the message only depends on the size of the original message which come from the KGC, and the cost of computation about R is irrelevant to the number of group members. Second, the number of messages transmitted by a node (namely, KGC or key material transmitting node) does not depend on the group size but depends only on the number of child nodes attached.

(2) Transmitting nodes scalability

The scalability provided in our protocol also depends on the structure of the key material transmitting tree. Designated multicast routers over Internet act as key material transmitting nodes in the tree, and each multicast router manages one sub-group only. If group members are sparsely distributed, it is better to use one material transmitting node to manage members of several nearby sub-groups. Even it might also happen that all of the group members attached to a transmitting node leave the multicast group after a period of time. This would result in a huge but sparse key material transmitting tree. To avoid this, the KGC can periodically delete or combine transmitting nodes to improve compactness and efficiency. On the other hand, when the number of group members increasing, new key material transmitting nodes can be included to trade-off the key management overhead incurred by new group members.

(3) Membership scalability

The proposed group key management protocol employs a distributed approach to achieve key distribution, and the cost of each component for key distribution is independent of the group size. Specifically, the cost of key renewing is independent of the group size and therefore our scheme is scalable in terms of membership.

5 Performance Comparison

We will evaluate the performance of our proposed protocol on three aspects: communication, computation and storage complexities. The major measure of communication complexity is the bandwidth consumption, e.g., the number of messages as well as the length of the messages for transmitting key material messages. The major measure of computation complexity is the operation overhead of the KGC, the key material transmitting nodes and group members for generating, forwarding and deriving the SEK, respectively. Storage complexity is concerned with the total length of parameters stored by each entity.

In our protocol, the length of each message for re-keying is less than $\log(p-1)$ *bits*, which is irrelevant to the size of group. As to the number of messages, the KGC sends two messages for system reconfiguration and a messages to the directly connected key material transmitting nodes, where a represents the degree of the key material transmitting tree. As a consequence, the total number of message transmitted by KGC is $(2+a)$ per re-keying, which is independent of the size of the group. Similarly, each material transmitting node N_i sends a messages to its children nodes. In addition, the number of messages for transmitting R to members by nodes N_i is only relevant to the number of members in U_i. In terms of storage complexity, the KGC must store the secret parameter r_i of node N_i. Thus, it is proportional to the number of transmitting nodes in the tree. Each material transmitting node is required to store the parameter x_i only, and each group member need to store two secrets: one

is the SEK and the other is the key-deriving parameter. With regard to the computation, one exponential modular operation is needed in SEK generation by KGC and SEK derivation by each group member respectively. One modular multiplication operation is needed when key material R is forwarded every time.

Table 1 compares our protocol with other well-known group key management protocols. SGM represents the subgroup manager in subgroup-based protocols, and represents key material transmitting node in proposed protocol also. Table 1 show that proposed protocol provides better scalability than existing protocols, and possesses less computational overhead than many protocols. In addition, the proposed protocol does not require group members to trust third parties but the KGC and does not need encryption/decryption operation in process of transmitting key material.

Table 1. Comparison of Secure Group Key Management Pprotocols

Items \ Protocols		IOLUS	LKH	DEP	SMP	WGL	Proposed
Total no. of keys		$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
No. of keys at a member		3	$O(\log_a n)$	4	2	$1+d$	2
No. of keys at a SGM		4	—	5	2	—	1
No. of keys at KGC		2	$O(n)$	$2+c$	$O(m)$	$O(n)$	$O(m)$
Cost of a join event	old member	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
	joining member	$O(1)$	$O(\log_a n)$	$O(1)$	$O(1)$	$O(\log_a n)$	$O(1)$
	involved SGM	$O(1)$	—	$O(1)$	$O(1)$	—	$O(1)$
	no. of key encryptions	$O(m)$	$O(\log_a n)$	$O(m)$	1	$2(d-1)$	0
	no. of messages	$O(1)$	$O(1)$	$O(1)$	$O(1)$	d	$O(1)$
Cost of a leave event	remained member	$O(1)$	$O(\log_a n)$	$O(1)$	$O(1)$	$O(\log_a n)$	$O(1)$
	involved SGM	$O(h)$	—	$O(h)$	$O(h)$	—	$O(h)$
	no. of key encryptions	$O(m)$	$O(\log_a n)$	$O(m)$	1	$2(d-1)$	0
	no. of messages	$O(h)$	$O(\log_a n)$	$O(h)$	$O(h)$	d	$O(h)$
1 affects n		No	Yes	Yes	No	Yes	Yes
Trusted intermediate nodes		Yes	—	No	No	—	No

n: size of group a: degree of tree d: depth of tree c: number of children of the sender.
m: number of subgroups h: average size of subgroup.

6 Conclusions

Security mechanisms are an urgent requirement for multicasting in order to ensure a safe and wide deployment for confidential group communications. Key management protocols play a crucial role in the whole secure multicast architecture. In real multicast communications, members can join and leave the group dynamically during the whole session. This dynamicity considerably affects the performance of the key management protocol. In this paper, we proposed a scalable group key management protocol to support secure communications in large dynamic multicast groups. In the

proposed key material transmitting tree, key material transmitting nodes share the load of key management with the KGC. No matter how large of the group size is, both communication and computation cost of KGC or material transmitting nodes are constant. As a result, in each node the cost of performing key management is independent of the group size. Therefore, it is applicable to the large dynamic group because of better scalability. When join or leave event occurs, the KGC only need to reconfigure system parameters in a small range for re-keying. Although inner nodes in the tree are used to transmit key materials, our protocol can protect message privacy against them. Only legitimate group members have access to the message contents. So the proposed protocol can overcome drawback that the trusted third parties are needed.

Acknowledgment. This work was supported by the National Science Foundation of China under Grant No.60403027, and by the Science Research Program of Education Bureau of Hubei Province under Grant No. 200629001.

References

1. Quinn, B., Almeroth, K.: IP Multicast Applications: Challenges and Solutions. IEFT RFC3170, (2001)
2. Cain, B., Deering, S., and Kouvelas, I., Thyagarajan, A.: Internet Group Management Protocol, version 3. IEFT RFC3376, (2002)
3. Sandro, R., David, H.: A Survey of Key Management for Secure Group Communication, ACM Computing Surveys. Vol.35, No.3, (2003), 309–329
4. Amir, Y., Kim, Y.D., Cristina, N.R., Tsudik, G.: On the Performance of Group Key Agreement Protocols, ACM Transactions on Information and System Security, Vol. 7, No. 3, (2004), 457–488
5. Challal, Y., Seba, H.: Group Key Management Protocols: A Novel Taxonomy, International Journal of Information Technology, Vol.2, No.2, (2005), 105–118
6. Harney, H., Muckenhirn, C.: Group Key Management Protocol (GKMP) Architecture, RFC 2093, (1997)
7. Chu, H.H., Qiao, L., Nahratedt K.: A Secure Multicast Protocol with Copyright Protection, ACM SIGCOMM Computer Communications Review, 32, 2, (2002), 42–60
8. Wallner, D., Harder, E., Agee, R.: Key Management forMulticast: Issues and Architectures, IETF RFC2627, (1999)
9. Wong, C.K., Gouda, M., Lam, S.S.: Secure Group Communications Using Key Graphs, IEEE/ACM Transactions on Networking, Vol.8, No.1, (2000), 16–30
10. Balenson, D., McGrew, D., and Sherman, A.: Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization, draft-balenson-groupkeymgmt-oft-00.txt, Internet-Draft, (1999)
11. Dunigan, T., Cao, C.: Group Key Management, Technical Report ORNL/TM-13470, (1998)
12. Mittra, S.: Iolus: A Framework for Scalable Secure Multicasting, ACM SIGCOMM, Vol.27, No.4, (1997), 277–288
13. Dondeti, L., Mukherjee, S., Samal, A.: A Dual Encryption Protocol for Scalable Secure Multicasting, In Proceedings of the IEEE Symposium on Computer and Communications, Red Sea, Egypt, (1999), IEEE Computer Society Press, Los Almitos, Calif.

14. Yang, W.H., Fan, K.W., and Shieh, S.P.: A Secure Multicast Protocol for the Internet's Multicast Backbone, ACM/PH International Journal Network Management, Vol.11, (2001), 129–136
15. Hardjono, T., Cain, B., and Monga, I.: Intra-domain Group Key Management for Multicast Security, IETF Internet draft, (2000)
16. Molva, R., Pannetrat, A.: Scalable Multicast Security with Dynamic Recipient Groups, ACM Transaction Information and System Security, Vol.3, No.3, (2000), 136–160
17. Setia, S., Koussih, S., Jajodia, S.: Kronos: A Scalable Group re-keying Approach for Secure Multicast, In Proceedings of the IEEE Symposium on Security and Privacy, Oakland Calif., IEEE Computer Society Press, (2000), 215–288
18. Steiner, M., Tsudik, G., Waidner, M.: Diffie-Hellman Key Distribution Extended to Group Communication, In Proceedings of the 3rd ACM Conference on Computer and Communications Security, (1996), 31–37
19. Rodeh, O., Birman, K., and Dolev, D.: Optimized Group Rekey for Group Communication Systems, Network and Distributed System Security, (2000)
20. Kim, Y., Perrig, A., Tsudik, G.: Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups, In proceeding of the 7th ACM conference on Computer and Communications Security. New York: ACM Press, (2000), 235–244
21. Abdel-Hafez, A., Miri, A., Orazco-Barbosa, L.: Scalable and Fault-tolerant Key Agreement Protocol for Dynamic Groups, International Journal of Network Management, Vol.16, (2006), 185–291
22. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System, EUROCRYP'94, LNCS 950, (1994), 275–286
23. Boyd, C.: On Key Agreement and Conference Key Agreement, Information Security and Privacy: Australasian Conference, LNCS 1270, (1997), 294–302
24. Lu, H.B.: A Novel High-Order Tree for Secure Multicast Key Management, IEEE Trans. on Computers, Vol.54, No.2, (2005),214–224
25. Zheng, S.Y., David, M, Jim, A.F.: A Communication-Computation Efficient Group Key Algorithm for Large and Dynamic Groups. Available online 22 May 2006, Computer Networks, 51, (2007), 69–83

A Time-Based Key Management Protocol for Wireless Sensor Networks*

Jiyong Jang[1], Taekyoung Kwon[2], and Jooseok Song[1]

[1] Department of Computer Science, Yonsei University
{souljang, jssong}@emerald.yonsei.ac.kr.
[2] Department of Computer Engineering, Sejong University
tkwon@sejong.ac.kr.

Abstract. It is not easy to achieve secure key establishment in wireless sensor networks without public key cryptography. Many key management protocols have been proposed for the purpose. Among them, LEAP is a simple and elegant protocol that establishes multi-level keys in an efficient way, but its security mainly relies on that of a single initialization key. Though it is assumed that the initial deployment phase is secure and the key is erased from sensor nodes after the initialization in LEAP, the assumption could not be viable for two reasons. First, the same key should be used again for node addition after the initialization phase whereas the new node can be captured before removing the key. Second, the initial deployment of dense networks may not take short as LEAP expected in many cases. This paper rethinks the security of LEAP and proposes a more secure scheme with a new notion of probabilistic time intervals. Rather we localize the impact of key compromise within the time intervals.

Keywords: Wireless Sensor Network, Security Protocol.

1 Introduction

Wireless sensor networks (WSNs) are dense wireless networks of sensor nodes which are constrained in their computation, communication, and storage capabilities. They are expected to play an important role in many applications such as environment monitoring, building management, health care, and military operation. Since they are deployed in unattended or even hostile environments, security mechanisms are required for various mission critical applications [1,2]. However, it is widely recognized that securing WSNs is quite challenging due to the limited features of sensor nodes.

Many key management schemes have been proposed to make secure links in WSNs. A probabilistic key pre-distribution scheme is proposed in [1]. In this scheme, a randomly chosen set of keys from a large key pool is assigned to each

* This work was supported by grant No. R01-2006-000-10614-0[1] and No. R01-2005-000-11261-0[2] from the Basic Research Program of the Korea Science & Engineering Foundation.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 314–328, 2007.

sensor node before node deployment. And then, two sensor nodes can share at least a common key with a certain probability. This scheme is improved in [3]. Two sensor nodes are required to share at least q secret keys to establish a pair-wise key. A random pair-wise key scheme is also introduced in [3] to provide perfect security against node capture. [4,5] use a threshold-based technique: If the number of compromised nodes does not exceed a threshold value, the rest of network is not affected by compromised ones. Recently researchers have suggested to utilize the expected location of sensor nodes after node deployment to improve the security and scalability of key establishment schemes [6,7,8]. However, it is limited to take advantage of the knowledge of locations since it is very difficult to guarantee the exact positions of sensor nodes.

Lately an efficient key management protocol called LEAP (Localized Encryption and Authentication Protocol) [9] has been proposed by Zhu, Setia, and Jajodia in order for supporting secure key establishment and in-network processing for large-scale WSNs by establishing four types of keys such as individual key, group key, cluster key, and pairwise shared key.

Most of the key management protocols including LEAP assume that an adversary may attack sensor networks after the initial key establishment phase, but the assumption could be incorrect while considering node addition phases in a hostile environment. Security of LEAP mainly depends upon that of the initialization key which is erased from sensor nodes after the initialization phase. However, the same key should be used again for node addition after that phase while the new node can be captured before removing the initialization key. In this paper, we rethink the security of LEAP and introduce a time-based key management protocol which improves security with a new notion of probabilistic time intervals.

This paper is structured as follows: We describe existing key management protocols in Section 2, and then rethink the LEAP protocol in terms of security in Section 3. We propose a time-based key management protocol in Section 4, and then analyze its performance and security in Section 5. We conclude this paper in Section 6.

2 Related Works

2.1 Key Management Protocols in WSNs

To provide secure communications in WSNs, sensor nodes first need to set up pair-wise keys with each other. There are generally three types of key agreement schemes: the trusted-server scheme, the self-enforcing scheme, and the key pre-distribution scheme [6]. The trusted-server scheme assumes that there is a trusted server for key establishment between nodes. However, this is not suitable for distributed sensor networks since it is usually hard to construct a trusted server. The self-enforcing scheme uses asymmetric cryptography, such as a public key certificate. However, a public key algorithm is not suitable for sensor networks because of limited power and computation resources of tiny sensor nodes. In key pre-distribution schemes, keying materials are pre-loaded into sensor nodes prior

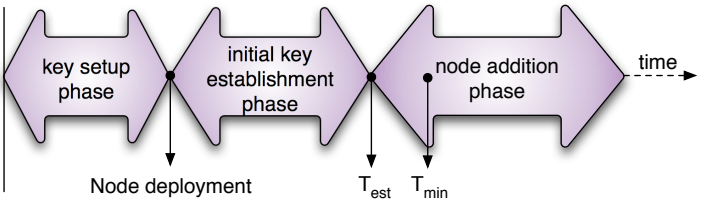


Fig. 1. 3 phases of key pre-distribution scheme

to the deployment. If we can utilize the location information of sensor nodes, we could make the scheme more efficient.

A key pre-distribution scheme consists of three phases in general: a key setup phase prior to deployment, an initial key establishment phase, and a node addition phase. We refer readers to Figure 1. During a key setup phase, a center generates keying materials which will be used to make a secure link, and then pre-loads some keying materials into sensor nodes prior to node deployment. After each sensor node discovers neighbor nodes sharing a common keying material, sensor nodes are able to establish pair-wise keys with each other during an initial key establishment phase. Sensor nodes which do not share any common keying materials, but are within the wireless communication range, can establish a path-key via a proxy node that already has pair-wise keys with both nodes. During the node addition phase, some additional nodes are deployed in sensor networks for several reasons, such as maintenance, replacement, routing, and so on. Now, we summarize two basic key pre-distribution schemes.

There are a lot of key pre-distribution schemes. At first, we can easily think about two naive solutions. One solution is to use a single master key in a whole network. Any pair of nodes can establish pair-wise keys using this master key. However, if one node is compromised, the whole network can be threatened by an attacker. The other solution is to assign a unique pair-wise key to every pair of nodes. Each sensor is required to store $N-1$ pair-wise keys so that whole sensor nodes are required to store $N(N-1)/2$ secret keys. Since compromising one node does not affect the rest of network, this scheme is perfectly resilient to node compromise. However, this scheme is not suitable for a large sensor network because of a limited memory constraint of sensor nodes.

2.2 EG Scheme

Overview. Eschenauer and Gligor proposed a probabilistic key pre-distribution scheme in [1]. This scheme relies on probabilistic key sharing between sensor nodes. At first, a center generates a large key pool P, and then randomly select k keys from pool P for each sensor node without replacement. k keys form a key ring of a sensor node, and the key ring is pre-loaded into a node prior to node deployment. As a result, two nodes share at least one key with a certain probability. After node deployment, each sensor node discovers its neighbors which share a common key in wireless communication range to establish a

pair-wise key. Such shared-key discovery phase establishes the topology of sensor network and path-key establishment phase reinforces a key connectivity of sensor network.

2.3 LEAP

Overview. LEAP is a cryptographic protocol allowing secure key establishment for WSNs [9]. In LEAP, it is assumed that sensor nodes are not mobile and every node has enough space to store hundreds of bytes of keying materials. It is also assumed that an adversary can eavesdrop on all traffic and extract all the information from the compromised node. LEAP provides confidentiality and authentication for secure communications in WSNs. LEAP is designed to support in-network processing such as data aggregation and passive participation. In-network processing can remove transfer of redundant messages so that energy consumption can be reduced. LEAP offers multiple keying mechanisms resulting from that different types of messages having different security requirements. LEAP also provides one-way key chain based authentication for inter-node traffic.

Establishment of Four Types of Keys. LEAP offers four types of keys to each sensor node - an individual key shared with the center, a pairwise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a group key shared by all the nodes in the network - and we summarize how to establish those keys below.

- **Individual Key:** Each node has a unique key shared with the center. This key is used for secure communication between a sensor node and the center. The individual key is generated as $K_u^m = f_{K^m}(ID_u)$ where f is a pseudo-random function and K^m is a master key known only to the center and ID_u is the id of a node u. This generated key is pre-loaded into a sensor node prior to node deployment.

- **Pairwise Key:** Every node has a pairwise shared key with its immediate neighbors respectively. Pairwise shared keys are used for secure distribution of cluster keys to its direct neighbor nodes and secure transmission of data. The center generates an initial key K_I and a node u computes a master key $K_u = f_{K_I}(ID_u)$. During neighbor discovery stage, node u broadcasts a HELLO message within its id and waits for response from neighbor v. The response message from node v contains id of v and message authentication code (MAC) for verifying node v's identity. And then, node u is able to authenticate node v since it can compute MAC value with the master key K_v which is derived as $K_v = f_{K_I}(ID_v)$.

$$u \rightarrow * : ID_u, Nonce_u$$
$$v \rightarrow u : ID_v, MAC_{K_v}(Nonce_u | ID_v)$$

  After authentication, node u computes a pairwise key with v as $K_{uv} = f_{K_v}(ID_u)$. Node v can also derive K_{uv} in the same way.

– **Cluster Key:** Each node has a common shared key with all its neighbors for supporting in-network processing. This key is used for encrypting or authenticating local broadcast messages. Node u first generates a random cluster key K_u^c and then encrypts the key with the pairwise shared key with each neighbor respectively and sends the encrypted key to each neighbor v_i. Each neighbor node decrypts the received message to get the cluster key K_u^c.

$$u \text{ generates cluster key } K_u^c$$
$$u \rightarrow v_i : (K_u^c)_{K_{uv_i}}$$

When one of the neighbor nodes is compromised, node u needs to generate a new cluster key and transmits it to the remaining neighbor nodes.

– **Group Key:** Every node has a globally shared key in the whole network. Group key is used for encrypting broadcast messages by the center. We can simply pre-load a group key into each node. We need to update the key periodically or when a compromised node is detected, that is, group rekeying problem must be considered.

– **Multi-hop Pairwise Shared Key:** A node has a pairwise shared key between a node and the aggregation node. This key is used by a node for transmitting data to an aggregation node which is multiple hops away. A node u first broadcasts a message with its id ID_u and id of the cluster head ID_c. The nodes which already have a pairwise shared key with both the node u and the cluster head c send reply messages to the node u. Now the intermediate nodes become the proxies. To establish a pairwise key S with cluster head c, node u splits S into m shares, such that $S = sk_1 \bigoplus sk_2 \bigoplus ... \bigoplus sk_m$, and then transmits each sk_i to the cluster head c via proxy v_i.

$$u \rightarrow v_i : (sk_i)_{K_{uv_i}}, f_{sk_i}(0)$$
$$v_i \rightarrow c : (sk_i)_{K_{v_ic}}, f_{sk_i}(0)$$

where $f_{sk_i}(0)$ is the verification key of key sk_i since the cluster head c can verify the validation of sk_i. After the cluster head c receives all shares, it restores a pairwise shared key S.

LEAP assumes that T_{min}, the time interval for an attacker to compromise a node, is larger than T_{est}, the time for a newly deployed node to complete neighbor discovery stage, as depicted in Figure 1. In LEAP, all nodes erase an initial key K_I and all the neighbors' master keys after time T_{min}. Therefore, an attacker compromising a node after T_{min} can obtain only the keying materials of the compromised node, not those of other nodes. Therefore, the affected fraction of the network due to node compromise can be localized. When a node compromise is detected, its neighbor nodes just erase the keys shared with the compromised node.

Local Broadcast Authentication. LEAP supports one-way key chain based authentication for local broadcast messages. Each node generates a one-way key

chain composed of keys called AUTH key, and sends the commitment (the first key) of key chain to its neighbors encrypting with each pairwise key. When a node transmits a message, it attaches the next AUTH key in the key chain to the message. The AUTH keys are disclosed in a reverse order. The commitment and received AUTH key allow a receiving node to verify the received message. Unlike μTESLA [11] which uses delayed key disclosure and requires time synchronization between neighboring nodes, this mechanism can provide immediate authentication.

3 Rethinking LEAP and Its Security

Most of the key agreement schemes assume that sensor networks are relatively secure against attacks during the initial key establishment phase and an adversary may capture sensor nodes to compromise the network after the phase. The LEAP protocol also has a similar assumption like that T_{est} is smaller than T_{min}. However, this assumption is often not true. For example, packet losses due to reasons such as narrow bandwidth or bad channel condition of sensor networks may happen while sensor nodes transmit data to each other during initial key establishment phase. This can cause several retransmissions of packets so that the time for sensor nodes to establish pair-wise keys each other, T_{est}, may take longer than expected, even T_{min}. Also T_{est} may be on the order of tens of minutes in certain deployment schemes.

If sensor nodes are scattered from airplanes or helicopters, then the nodes may settle sparsely and need enough time to set up the network and establish pairwise keys. During this time, an adversary can capture a sensor node and get an initial key K_I. Moreover some researches [12] show that it takes less than 1 minute to dump all of the EEPROM, program Flash, and a chip's SRAM. An initial key K_I can be disclosed if it only takes on the order of seconds to compromise a node. LEAP also assumes that the node moves K_I from non-volatile memory into volatile memory to make the scheme more secure. However, this assumption is not true since both RAM and flash are accessible to an adversary [12]. In above cases, an adversary is able to get an initial key K_I, and then inject erroneous information or add new nodes at her pleasure.

Moreover, in case of LEAP, a newly deployed node which is added to the network after the initial key establishment phase will carry the initial key K_I and may be captured in an hostile environment. Thus, as for node addition in LEAP, an initial key K_I should never be used after the initial time T_{min} without permission even for the legitimate new nodes since an adversary is able to capture a node in the initial T_{min} and find out an initial key K_I within T_{min} afterward.

4 A Time-Based Key Management Protocol

4.1 A Time-Based Deployment Model

As we have already mentioned in the previous section, the time for sensor nodes to establish pair-wise keys each other, T_{est}, may take longer than the time

interval for an adversary to compromise a node, T_{min}. If an initial key K_I is disclosed within T_{min} time, then the whole sensor network is threatened by an attacker. Even though an initial key K_I is disclosed by an attacker, the portion of network compromised must be minimized. For that reason we split the time domain to disperse the damage resulting from the disclosure of an initial key K_I. Now we introduce more secure key pre-distribution protocol with time-based multiple K_I. Selection of K_I with probabilistic time intervals is as follows:

Key Setup Phase

- First a center generates a pool of P initial keys and divides whole lifetime of sensor network into P time slots.
- A center assigns an initial key to each time slot.
- The initial key of the deployment time slot and m master keys of randomly-chosen time slots are pre-loaded into sensor nodes prior to deployment. When the deployment time slot is T_i, sensor nodes stores an initial key K_{Ii} and m randomly-chosen master keys.

Initial Key Establishment Phase

- According to the original LEAP protocol, an initial key establishment phase means the first time slot T_1.
- Since all sensor nodes deployed at an initial key establishment phase contains the initial key K_{I1}, they can establish pair-wise keys using K_{I1}.
- After a node u computes a master key $K_{u1} = f_{K_{I1}}(ID_u)$, node u broadcasts a HELLO message within its ID and then waits for a response from neighbor node v. Node v sends node u a response message including its ID and MAC.

$$u \rightarrow * : ID_u, Nonce_u$$
$$v \rightarrow u : ID_v, MAC_{K_{v1}}(Nonce_u | ID_v)$$

- Now both u and v can compute a pair-wise key $K_{uv} = f_{K_{v1}}(ID_u)$.

Node Addition Phase

- During a node addition phase, newly deployed nodes first discover neighbor nodes which share common keying materials.
- Sensor nodes are able to generate pair-wise keys with other nodes which are deployed at the same time slot using the same initial key.
- And then, sensor nodes can establish pair-wise keys with other nodes which are deployed at different time slots, but have the master key derived from the current initial key.

  Let u and v be a newly deployed node at time slot T_k and a pre-deployed node respectively. If a node v has the master key K_{vk} derived from the

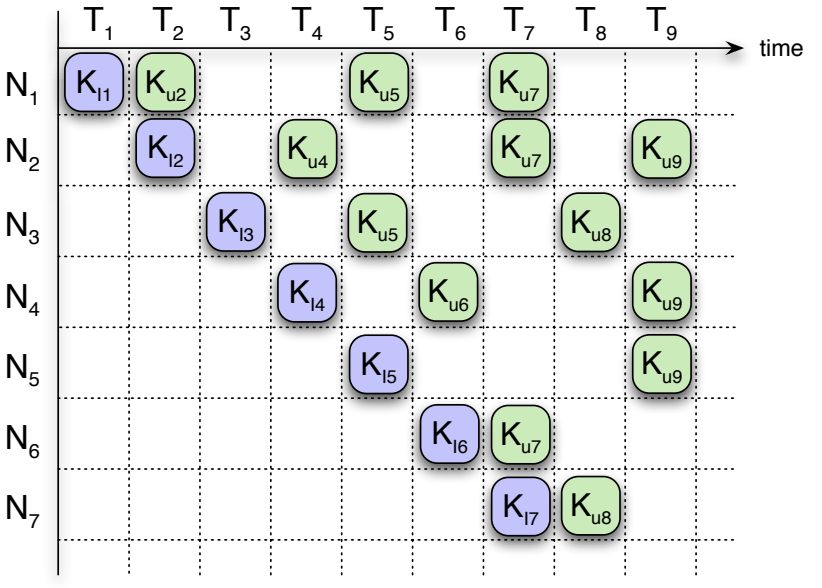


Fig. 2. Example of selecting K_I with probabilistic time intervals

current initial key K_{Ik}, both node u and v can compute a pair-wise key $K_{uv} = f_{K_{vk}}(ID_u)$ because node u is also able to generate a master key of v, K_{vk}, using the current initial key K_{Ik} and ID of v.

- After all, during the deployment time slot, sensor nodes can establish pair-wise keys with each other by using the initial key. For the other m time slots, sensor nodes are able to establish a secure link with other nodes by using an appropriate one of the m master keys.
- After that, a pair of sensor nodes that do not share a keying material but are in wireless communication range can establish path-keys via proxy nodes.

4.2 Example of Time-Based Deployment Model

Readers are referred to Figure 2. T_n and N_n represent each time slot and group of nodes deployed at T_n time slot, respectively. As shown in Figure 2, nodes of group N_1 store the initial key K_{I1} of time slot T_1 and m master keys of randomly-chosen time slots including K_{u2}, K_{u5} and K_{u7}. Since, according to original LEAP protocol, master key is derived from an initial key and node ID, $K_{u2} = f_{K_{I2}}(NodeID)$. All nodes of group N_1 are able to establish pair-wise keys each other using the initial key K_{I1} during time slot T_1. Then, they are able to establish a secure communication with nodes of group N_2 using the master key K_{u2} during time slot T_2. Note that all nodes of group N_2 can also derive the master

key K_{u2} from the initial key K_{I2}. Similarly, they can also generate pair-wise keys with nodes of group N_5 and N_7 using master keys K_{u5} and K_{u7}, respectively.

4.3 Practical Application of Time-Based Deployment Model

Location-based key management protocols are very efficient methods in terms of key connectivity and storage overhead. However, location information of sensor nodes is crucial since these schemes utilize the locations of sensor nodes. Even though the exact deployment locations of sensor nodes are not necessarily in these schemes, we must know the dense spots of sensor nodes prior to node deployment. Moreover, in case that we divide the whole region into many small areas, these schemes can provide higher resilience, but have a great difficulty in deploying sensor nodes. On the other hand, in case that we have the large size of areas, these schemes would be more vulnerable to attacks, but have no difficulty in deploying sensor nodes. That is, these schemes have the tradeoff between the security and the easy deployment.

On the contrary, location information of sensor nodes does not required to employ a time-based deployment model so that additional sensor nodes are easily added on WSN without the restriction of deployment points in our scheme. Therefore, our scheme is more beneficial to node deployment. Furthermore, as we described above, a time-based deployment model divides time into many small parts; thus, the effect of node capture attack can be localized into a small part. That is, we are able to have both high resilience and no difficulty in deploying sensor nodes by taking advantage of a time-based deployment model.

A time-based deployment model can be employed in a more practical way. In this model, every time slot appears continually, which means node addition must occur periodically. In fact, however, after initial deployment phase, additional nodes will be added to WSN irregularly. Therefore, we suggest the practical application of time-based deployment model as depicted in Figure 3. During a normal operation phase, WSN performs a normal operation such as sensing and transferring data without node additions. To deploy additional nodes, a center broadcasts an authenticated packet which notices pre-deployed sensor nodes to prepare node additions. The packet will contain information such as when the following time slot starts and for how many time slots node addition lasts. We assume that the length of a time slot is very short, in that sensor nodes just need to time to exchange keying materials and generate pair-wise keys during a time slot.

In a practical application of time-based deployment model, the beginning of new time slots means the occurrence of node additions so that the number of time slots is approximately equal to the number of occurrences of node addition. The number of time slots also means the size of key pool. Therefore, the size of key pool is influenced by the frequency of node additions. If the additional sensor nodes are deployed for the purpose of complementing sensor networks, the frequency of node additions is not necessarily high; thus, the size of key pool could be small.

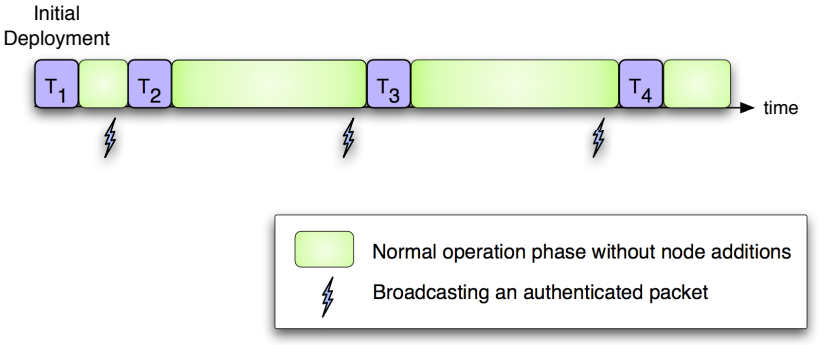


Fig. 3. Practical application of time-based deployment model

5 Performance and Security Analysis

5.1 Performance Analysis

Key Connectivity. When sensor nodes are deployed at a specific time slot, pre-deployed sensor nodes must have the master key, which is derived from the initial key of that time slot, to establish pair-wise keys each other. For example, N_7 group of sensor nodes are deployed at time slot T_7. To make secure links, N_1-N_6 group of sensor nodes must have the master keys which is derived from the initial key K_{I7}. At first, consider N_1's probability of sharing a keying material with N_7. The probability is [the number of cases where $m-1$ master keys are chosen from key pool except for K_{I1} and K_{I7}]/[the number of cases where m master keys are chosen from key pool except for K_{I1}]. That is, $\binom{P-2}{m-1}/\binom{P-1}{m} = \frac{m}{P-1}$. Since the master keys of every time slot would be chosen with the same probability, N_1's probability of sharing keying materials with other prospective sensor nodes(N_2-N_P) is $\frac{m}{P-1}$. Now, N_i's probability of sharing keying materials with prospective sensor nodes, $p_{pros}(i)$, can be calculated as:

$$p_{pros}(i) = \frac{\binom{P-i-1}{m-1}}{\binom{P-i}{m}} = \frac{m}{P-i} \tag{1}$$

as long as $(P-i)$ is greater than m; otherwise, $p_{pros}(i) = 1$. Figure 4 describes Eq. 1 for various values of m where the size of key pool P is 500. Although the size of 500 seems small for the key pool, it will be enough to operate the sensor network. (Refer to section 4.3.) After $(P-i)$ becomes smaller than m, the master keys of all remaining time slots will be chosen so that the probability $p_{pros}(i)$ is 1. As shown in the figure, the probability p_{pros} becomes higher as the index of time slot i increases, in that the size of key pool from which master keys are chosen becomes smaller. Also, the more master keys sensor nodes have, the higher the probability becomes.

Now we are able to compute $p_{time}(t)$, which means the probability that N_t group of sensor nodes share keying materials with pre-deployed sensor nodes and

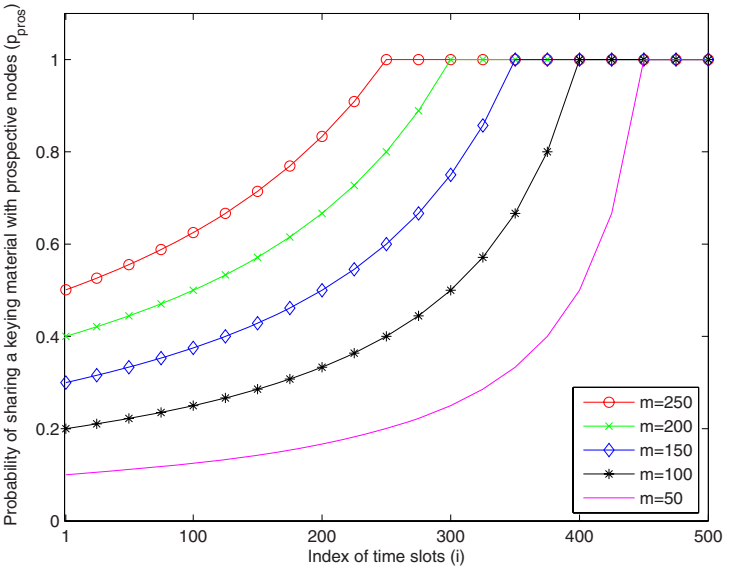


Fig. 4. [p_{pros}] The probability that N_i shares keying materials with prospective sensor nodes

N_t themselves at time slot T_t, using p_{pros}. If we assume that sensor nodes are uniformly distributed at every time slot, p_{time} is:

$$p_{time}(t) = (\sum_{i=1}^{t-1} p_{pros}(i) + 1)/t \tag{2}$$

which means the average value of the probabilities of sharing keying materials with each pre-deployed group of sensor nodes. In Eq. 2, 1 means that all sensor nodes deployed at the same time slot can make secure links. Figure 5 draws Eq. 2 for various values of m where the size of key pool P is 500. At time slot T_1, all sensor nodes have the same initial key so that the probability is 1. As time passes, sensor nodes would be added to the sensor network and the probability p_{time} drops, in that p_{pros} is much smaller than 1. After the probability p_{time} nearly decreases to the minimum value of p_{pros}, p_{time} becomes higher since p_{pros} increases.

The direct key connectivity C that a sensor node is able to generate pair-wise keys with at least one of the immediate neighbor nodes (i.e., 1-hop connectivity) is derived as follows:

$$C = 1 - (1 - p_{time})^d \tag{3}$$

where d means the average number of neighbor nodes. To calculate the key connectivity and show the effect of neighbor nodes on the key connectivity, we

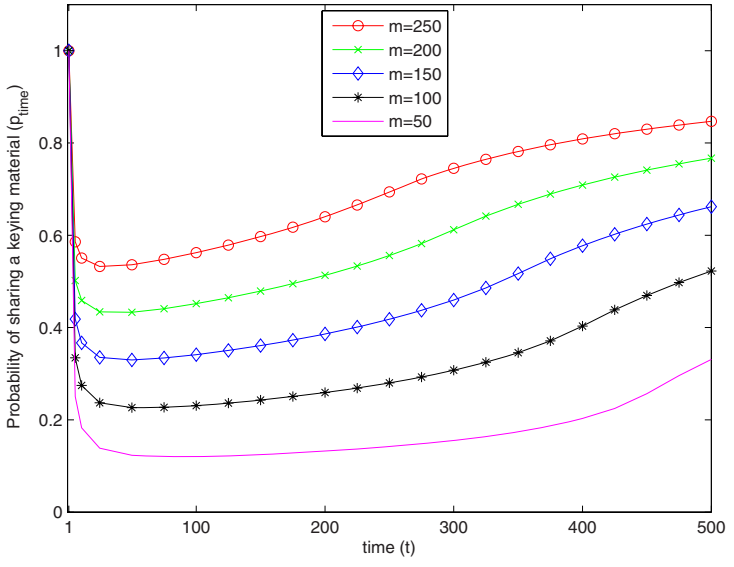


Fig. 5. $[p_{time}]$ The probability that N_t shares keying materials with pre-deployed sensor nodes and the nodes being deployed in the same time interval

choose the lowest value of p_{time} from Figure 5. Figure 6 describes Eq. 3 for various values of m where the size of key pool P is 500 and p_{time} has the minimum value. As shown in Figure 6, while the number of keys stored in a sensor node remains the same, the key connectivity goes high as the number of neighbor nodes increases.

Storage Overhead. As for the storage requirement, we can evaluate that our scheme needs a reasonable cost in modern sensor nodes such as MICA-Z and Telos. Sensor nodes choosing about 100 keys from key pool of size 500 are able to share a keying material with at least one of the 10 neighbor nodes with more than 0.9 probability (refer to Figure 6). Note that this requirement only corresponds to the node addition phase, while the connection probability is 1, saying, deterministic at the initial deployment. Also note that we only consider the direct key connectivity (1-hop connectivity) as for the probability in Figure 6. The remaining unconnectivity can be resolved by the help of other (proxy) neighbor nodes easily. When the size of a key is 64 bits, a center and a sensor node need approximately 4KB and 0.8KB memory space, respectively. The number of keys saved in memory will decrease as old keys can be discarded. For strengthening the security of WSNs with regard to node capture, we believe the initial requirements of 0.8KB are reasonable for the modern sensor nodes such as MICA-Z having 128KB program memory, 4KB runtime memory, and 512KB external memory [13].

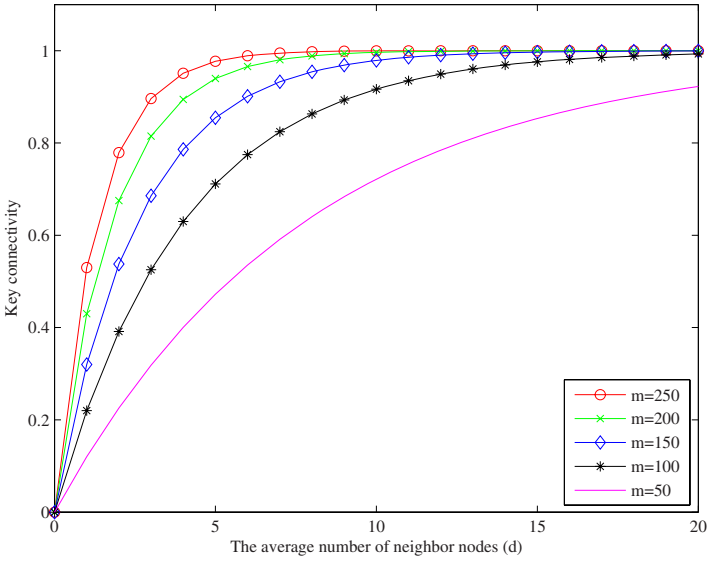


Fig. 6. Key connectivity that a sensor node can make a secure link with at least one of the immediate neighbor nodes

5.2 Security Analysis

When node compromise is detected, the keying materials which are associated with the compromised node must be revoked to prevent an adversary from attacking the rest of network using information extracted from compromised node. Since detection of node compromise is not easy, the additional portion of network that an adversary can compromise using the keying materials obtained from x captured nodes represents the resilience of schemes. In other words, the resilience of schemes means the survivability of network against node compromise.

Since an initial key K_I is removed from a node after T_{est} in original LEAP protocol, an attacker can use only pair-wise keys and cluster keys even if she succeeded in capturing a node. An attacker is not able to use the compromised node in other areas so that the affected fraction of network due to node capture is localized. Since the portion of network compromised is localized, wormhole attack or sinkhole attack [14] can be protected. However, if an adversary succeeds in capturing a node before T_{est}, she is able to get an initial key K_I so that the whole network can be compromised.

The damage resulting from a disclosure of an initial key K_I can be minimized by using selection of K_I with probabilistic time intervals scheme. Even if an adversary is able to get an initial key K_{Ia} at time slot T_a, only the nodes deployed at time slot T_a, not whole network, are compromised. Even if an attacker knows an initial key K_{Ia}, she cannot retrieve the previous initial keys so that she is never able to get an information from transmitted data before time slot T_a. Therefore, the selection of K_I with probabilistic time intervals scheme provides backward

confidentiality. As only master keys, not initial keys, are stored in sensor nodes for the other m time slots, this scheme also provides forward confidentiality.

5.3 Comparison

Table 1 shows key connectivity, key storage overhead and resilience of EG scheme, LEAP, and our scheme where N is the size of the sensor network, d is the average number of neighbor nodes, and t is the index of time slots. In EG scheme, m keys out of key pool P are chosen for each sensor node.

Table 1. Comparison with EG scheme and original LEAP

		EG scheme [1]	LEAP [9]	our scheme
Key connectivity		$p_1 = 1 - \frac{((P-m)!)^2}{P!(P-2m)!}$	1	$(\sum_{i=1}^{t-1} \frac{m}{P-i} + 1)/t$
Initial key storage overhead		m	1	$m+1$
Compromised network due to node capture	after T_{est}	$p_1 \cdot (N-1)$	d	d
	before T_{est}	$p_1 \cdot (N-1)$	*whole network*	*one group of* N_i
Resilience against wormhole attack or sinkhole attack		X	O	O
Forward confidentiality		X	X	O
Backward confidentiality		X	X	O

Because of the high key connectivity between the nodes deployed at the same time slot, key connectivity of our scheme is higher than that of EG scheme while the initial key storage overhead remains unchanged. Since we select multiple K_I with probabilistic time intervals, LEAP shows the better performance than ours in terms of key connectivity and storage overhead. However, our scheme is able to minimize the impact of node capture attack even though an attacker succeeds the node capture attack before T_{est}. Note that if an adversary can compromise a sensor node before T_{est}, K_I is disclosed so that the whole network is compromised in LEAP. Ours as well as LEAP prevent an adversary from launching wormhole attack or sinkhole attack because a node knows all its neighbors after neighbor discovery. Since an attacker cannot derive previous initial keys from the current initial key or next initial keys from master keys, our scheme provides backward confidentiality and forward confidentiality.

6 Conclusions

The multiple keying mechanism of LEAP satisfies the multiple usages of sensor networks, but we have found that LEAP actually missed a possible disclosure of an initial key K_I. Its security mainly relies on that of an initialization key while the

initial deployment phase is assumed secure and the key is erased from sensor nodes after the initialization phase. However, the same key should be used again for node addition after that phase while the new node can be captured before removing the initialization key. And the assumption of security in the initial deployment phase is not viable in many cases since the initial deployment of dense networks may not take short as LEAP expected. This paper rethinks the security of LEAP and proposes a more secure scheme with a new notion of probabilistic time intervals. Rather we localize the impact of K_I disclosure within the time intervals.

References

1. L. Eschenauer and V. D. Gligor, "A Key-management Scheme for Distributed Sensor Networks," in *Proc. of the 9th ACM Conference on Computer and Communication Security (CCS'02)*, pp. 41-47, Nov. 2002.
2. A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," Communications of the ACM, vol. 47, no. 6, pp. 53-57, June 2004.
3. H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," in *IEEE Symposium on Research in Security and Privacy*, pp. 197-213, 2003.
4. W. Du, J. Deng, Y.S. Han, and P. Varshney, "A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003.
5. D. Liu, and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security(CCS'03)*, pp. 52-61, October 2003.
6. W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge," in *IEEE Transactions on dependable and secure computing*, vol. 3, no. 1, pp. 62-77, Feb. 2006.
7. D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-Aware Key Management Scheme for Wireless Sensor Networks," in *Proc. of ACM Workshop Security of Ad Hoc and Sensor Networks (SASN'04)*, pp. 29-42, Oct. 2004.
8. J. Lee, T. Kwon, and J. Song, "Location-aware key management using multi-layer grids for wireless sensor networks," Applied Cryptography and Network Security, Lecture Notes in Computer Science, Vol. 3989, Springer-Verlag, pp. 390-404, 2006.
9. S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proc. of the 10th ACM Conference on Computer and Communication Security (CCS'03)*, pp. 62-72, November 2003.
10. R. Blom, "An Optimal Class of Symmetric Key Generation Systems," in *Advances in Cryptology: Proc. EUROCRYPT '84*, pp. 335-338, 1985
11. A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar, "SPINS: Security protocols for sensor networks", in *Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 189-199, July 2001.
12. C. Hartung, J. Balasalle, and R. Han, "Node Compromise in Sensor Networks: The Need for Secure Systems," Technical Report CU-CS-990-05, University of Colorado at Boulder, Jan. 2005.
13. Crossbow Technology (http://www.xbow.com)
14. C. Karlof and D. Wagner, "Secure Routing in Sensor Networks: Attacks and Countermeasures," in *Proc. of the 1st IEEE Workshop on Sensor Network Protocols and Applications*, May 2003.

Identity-Based Threshold Decryption Revisited*

Shengli Liu[1], Kefei Chen[1], and Weidong Qiu[2]

[1] Department of Computer Science and Engineering
{liu-sl, chen-kf}@cs.sjtu.edu.cn
[2] School of Information Security Engineering
Shanghai Jiao Tong University, Shanghai 200240, P.R. China
qiuwd@sjtu.edu.cn

Abstract. We present an identity-based threshold decryption scheme, which is secure against adaptive chosen ciphertext attack (IND-CCA) in the random oracle and generic model (ROM+GM). In our scheme, a one-time Schnorr signature is used for public checkability of ciphertexts, and a new method was proposed to distribute the shares of the decryption key to decrease the number of bilinear pairings involved. Hence our scheme is more efficient than the recent proposals by Baek and Zheng in [1] and by Kiltz and Galindo in [13].

1 Introduction

Identity-based encryption system allows an arbitrary string like a name, an email address, or telephone number to serve as a public key. The model of identity-based encryption was first proposed by Shamir in 1984 in [19]. However, it was not until a few years ago that the first practical implementation was presented by Boneh and Franklin in [2] and by Cocks in [5]. Boneh and Franklin's work, called BF scheme, employs bilinear pairings and is proved to be secure against chosen ciphertext attack in the random oracle model. Since then, lots of works have followed this line to give identity-based primitives, like identity-based signatures, key exchange, hierarchical identities, etc.

Threshold decryption is used to decentralize the power of decryption by distributing shares of the decryption key to several decryption servers. Given a ciphertext, decryption servers independently decrypt the ciphertext to get decrypted shares. A combiner will collect the decrypted shares and combine them to recover the original plaintext. This paper will study threshold decryption in an identity-based scenario. There is a Private Key Generator (PKG) in an identity-based encryption system, who has a master key s. PKG uses the master key s and public parameters to derive a private key S_{ID} for a user with the submitted identity information ID. Threshold decryption in this setting has two choices. One is to share the master key s among several PKGs, and the other is to share the private key S_{ID} among several decryption servers. The latter is preferable

* The work is supported by NSFC under the grant no. 60673077, 60573030, and the National 863 Project under grant no. 2006AA01Z422.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 329–343, 2007.

to the former for two reasons, as was pointed in [1]. (1) The former approach requires the PKGs stay "on-line" for decryption. This betrayed Shamir's original intention of identity-based systems [19]. (2) The latter approach is the only choice for threshold decryption in the hierarchical identity-based encryption settings.

Related Works. Libert and Quisquater's ID-based threshold decryption scheme in [14] belongs to the former approach. Dodis and Yung's work in [7] belongs to the latter, however, they only gave a sketched description and did not consider chosen ciphertext attack. Baek and Zheng's work in [1] also belongs the the latter approach. Baek and Zheng for the first time formalizes the model of ID-based threshold decryption and constructed the first scheme secure against chosen ciphertext attack in [1], which we will refer to as BZ scheme. In [13], Kiltz and Galindo considered both approaches and gave the first threshold ID-Based Key Encapsulation Mechanism (IB-KEM) decryption scheme in the standard model, which we will refer to as KG scheme. However, KG scheme is not a full threshold ID-Based Encryption (IBE) scheme. As pointed in [13], to construct a full threshold IBE with CCA security is not easy: **full threshold IBE (CCA)= threshold Tag-IB-KEM (CCA)+ symmetric encryption(CPA)**, where **threshold Tag-IB-KEM (CCA)=threshold IB-KEM + key derivation function +MAC**. This way of construction of a full threshold IBE, of course, is preferable to long plaintexts.

This paper considers ID-based threshold decryption, which is full threshold IBE compared to KG scheme in [13], following the latter approach. In this paper, we use the Schnorr-type NIZK proof of equality of two discrete logarithms to provide decryption share validity. This idea has been used in the non-identity-based setting by Shoup and Genarro in [22,23]. We extend variants of ElGamal encryption by a one-time Schnorr signature. This idea was used in [21] in the non-identity-based setting. This paper extends the above ideas to ID-based scenarios and prove its security in the Random Oracle + Generic model. The model of ID-based threshold decryption and security notion are introduced in Section 1.1. Preliminaries are presented in Section 2. In Section 3, we propose an ID-based threshold scheme, and give a comparison to BZ scheme and KG scheme in terms of computational overhead. In Section 4, we give a security analysis to the proposed scheme. Section 5 concludes this paper.

Throughout the paper, we use $\mathbb{Z}_q$ to denote the set $\{0, 1, 2, \cdots, q-1\}$, and $\mathbb{Z}_q^*$ denote $\mathbb{Z}_q \backslash \{0\}$. By $\in_R S$, we mean choosing a random element from a set S with a uniform distribution. Let $|S|$ denote the cardinality of set S.

1.1 Model of ID-Based Threshold Decryption

We consider the model proposed by Baek and Zheng in [1]. The only difference is that we separate the KeyVer algorithm from KeyDis algorithm and ShareVer algorithm from ShareCom algorithm. Consequently, an ID-based threshold decryption (`IdThd`) scheme consists of 8 poly-time algorithms.

Setup$(1^k) \rightarrow (s, para)$**.** Taken as input a security parameter 1^k, a randomized setup algorithm outputs a master key s and public parameters $para$ for the PKG.

Der$(s, para, ID) \rightarrow S_{ID}$**.** Taken as input the master key s, the public parameter $para$ and the identity information ID of a user, a derivation algorithm, run by the PKG, derives the private key S_{ID} for the user.

KeyDis$(para, S_{ID}, n, t) \rightarrow (sk_i, y_i)$**.** Taken as input the parameter $para$, the private key S_{ID}, the total number n of the decryption servers, and t a threshold value, a randomized key distribution algorithm generates a shared key sk_i of S_{ID} and a corresponding verification key y_i for server Γ_i, $1 \leq i \leq n$. The verification keys are also given to the combiner in an authentic way.

KeyVer$(para, ID_i, y_i, sk_i) \rightarrow \{valid, \perp\}$**.** Taken as input the public parameter $para$, identity information ID_i and the shared key sk_i and verification key y_i, output "valid" if sk_i and y_i are the keys generated from the **KeyDis** algorithm, and "$\perp$" otherwise. This algorithm is run by each server to testify the validity of the key share.

Enc$(para, ID, M) \rightarrow C$**.** Taken as input the public parameter $para$, the identity information ID of the receiver, and the plaintext M, a randomized encryption algorithm outputs the corresponding ciphertext C.

DecShare$(para, sk_i, C) \rightarrow \{\delta_{i,C}, \perp\}$**.** Taken as input the public parameter $para$, the decryption key share sk_i and the ciphertext C, decryption server Γ_i runs the decryption share generation algorithm to output an invalid symbol $\perp$ or a valid decrypted share $\delta_{i,C}$ of C.

ShareVer$(para, y_i, C, \delta_{i,C}) \rightarrow \{valid, \perp\}$**.** Taken as input the public parameter $para$, the verification key y_i, the ciphertext C and the decrypted share $\delta_{i,C}$, the share verification algorithm determines whether $\delta_{i,C}$ is valid or not.

ShareCom$(para, C, \{\delta_{i,C}\}_{i \in \Phi}) \rightarrow \{M, \perp\}$**.** Taken as input the public parameter $para$, the ciphertext C and a set of decrypted share $\delta_{i,C\,i \in \Phi}$, with $\Phi \subset \{1, 2, \cdots, n\}$ and $|\Phi| \geq t$, the share combining algorithm outputs the original plaintext "M" or invalid symbol "$\perp$".

1.2 Security of an ID-Based Threshold Decryption System

The notion of chosen-ciphertext security for an ID-based threshold decryption system was formalized in [1]. An ID-based threshold decryption system against Indistinguishable Adaptive Chosen Ciphertext Attack (IND-CCA) is defined with **IND-CCA-Game**$_{\mathcal{A}}(\beta)$, a game between an attack algorithm $\mathcal{A}$ and a challenger.

IND-CCA-Game$_{\mathcal{A}}(\beta)$**.** For an algorithm $\mathcal{A}$ and a bit $\beta \in \{0, 1\}$, the game between a challenger and $\mathcal{A}$ is as follows.

Setup. The challenger runs **Setup**$(1^k) \rightarrow (s, para)$ to obtain the public parameters $para$, the master key s. It gives $para$ to $\mathcal{A}$.

Key Query 1. $\mathcal{A}$ issues a number of private key extraction queries. For each query ID, the challenger responds with $S_{ID} \leftarrow Der(s, para, ID)$.

Corruption. $\mathcal{A}$ corrupts $t-1$ out of n decryption servers. Without loss of generality, let server $1, 2, \cdots t-1$ be the corrupted servers.

Target. $\mathcal{A}$ issues a target identity ID^*, which is not queried in the corruption phase. The challenger runs the following algorithms.
- $Der(s, para, ID^*) \rightarrow S_{ID^*}$.
- KeyDis$(para, S_{ID^*}, n, t) \rightarrow (sk_i, y_i)$ for $i = 1, 2, \cdots, n$.

The challenger gives $\mathcal{A}$ the private keys $sk_1, sk_2, \cdots, sk_{t-1}$ of the corrupted servers and all the verification key $y_1, y_2, \cdots, y_n$.

Decryption Share query 1. $\mathcal{A}$ issues a ciphertext C to the uncorrupted servers for decryption. The uncorrupted servers j, $t \leq j \leq n$, responds with $\delta_{j,C}$ or $\perp$ by calling the algorithm $DecShare(para, sk_j, C)$.

Challenge. $\mathcal{A}$ outputs two plaintexts (M_0, M_1) of equal length. The challenger chooses a bit $\beta \in_R \{0,1\}$ and computes the challenge ciphertext $C^* \leftarrow Enc(para, ID^*, M_\beta)$. The challenger returns (C^*, ID^*) to $\mathcal{A}$.

Key Query & Decryption Share query 2. Like **Key query 1** and **Decryption Share Query 1**, except that the queried identity ID is not the target ID^* and the queried ciphertext C is not the challenge ciphertext C^*.

Guess. Algorithm $\mathcal{A}$ outputs its guess bit β'. It wins the game if $\beta' = \beta$.

When **Decryption Share query 1 & 2** are missing in the above game, the attack is reduced to a chosen plaintext attack (**IND-CPA**). The game is denoted by **IND-CPA-Game**$_{\mathcal{A}}(\beta)$.

Definition 1. *The advantage of an algorithm $\mathcal{A}$ that outputs $\beta' \in \{0,1\}$ in attacking an* `IdThd` *system with* **IND-CCA-Game**$_{\mathcal{A}}(\beta)$ *is defined as*

$$\begin{aligned} \mathsf{Adv}^{CCA}_{\texttt{IdThd}}(\mathcal{A}) &= |\ \boldsymbol{Pr}[\textit{IND-CCA-Game}_{\mathcal{A}}(1) = 1] - \boldsymbol{Pr}[\textit{IND-CCA-Game}_{\mathcal{A}}(0) = 1]\ | \\ &= 2\boldsymbol{Pr}[\beta' = \beta] - 1. \end{aligned}$$

$\mathsf{Adv}^{CPA}_{\texttt{IdThd}}(\mathcal{A})$ *can be similarly defined.*

2 Preliminaries

2.1 Bilinear Pairings

Bilinear pairings are building blocks in ID-based cryptography.

Definition 2. *Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. Let $a, b \in_R Z_q^*$. A bilinear pairings is a map $e : G_1 \times G_1 \rightarrow G_2$ with the following properties:*

1. *Bilinear: $e(aP, bQ) = e(P, Q)^{ab}$;*
2. *Non-degenerate: There exists P and $Q \in G_1$ such that $e(P, Q) \neq 1$;*
3. *Computable: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.*

2.2 Generic Algorithms for Groups and Generic Model

We recall the definition of generic algorithms for a group of order q, which was presented in [17,21,24]. The data of a generic algorithms is partitioned into group elements and non-group elements.

Suppose G_1 is an additive group. A **generic step** for G_1 is a multi-variate scalar muliplication: $\mathbb{Z}_q^k \times G_1^k \to G_1$ with $k \geq 0$, more precisely,

$$(a_1, \cdots, a_k, P_1, \cdots, P_k) \to \sum_{i=1}^{k} a_i P_i.$$

Suppose G_2 is a multiplicative group of order q. A generic step for group elements in G_2 are a multi-variate exponentiations:

$$\mathbb{Z}_q^k \times G_2^k \to G_2 : (a_1, \cdots, a_k, g_1, \cdots, g_k) \to \prod_{i=1}^{k} g_i^{a_i}.$$

If $k = 2, a_1 = 1, a_2 = \pm 1$, the generic step in G_2 presents multiplication/division. If $k = 0$, it means there are two elements (g, y) in G_2 for discrete logarithm $\log_g y$. Similar arguments hold true for G_1.

- A non-interactive generic algorithm for G_1 is a sequence of k generic steps.
  Input. $F_1, \cdots, F_{k'} \in G_1$ for $1 \leq k' < k$.
  Output. $F_i = \sum_{j=1}^{i-1} a_j F_j$ for $i = k' + 1, \cdots, k$, where $(a_1, a_2, \cdots, a_{i-1}) \in \mathbb{Z}_q^{i-1}$ depends arbitrarily on i, the non-group input and the collision set $\Lambda_1 = \{(j, l) | F_j = F_l, 1 \leq j < l \leq i - 1\}$ of previous collisions of group elements. Set Λ_1 excludes trivial collisions which occur with probability 1 for all choice of secret data.
- A non-interactive generic algorithm for G_2 is a sequence of k generic steps.
  Input. $f_1, \cdots, f_{k'} \in G_2$ for $1 \leq k' < k$.
  Output. $f_i = \prod_{j=1}^{i-1} f_j^{a_j}$ for $i = k' + 1, \cdots, k$, where $(a_1, a_2, \cdots, a_{i-1}) \in \mathbb{Z}_q^{i-1}$ depends arbitrarily on i, the non-group input and the collision set $\Lambda_2 = \{(j, l) | f_j = f_l, 1 \leq j < l \leq i - 1\}$ of previous collisions of group elements. Set Λ_2 excludes trivial collisions.
- Given G_1 and G_2 of order q with a bilinear mapping $e : G_1 \times G_1 \to G_2$. A **non-interactive generic algorithm** for G_1 and G_2 is a sequence of k generic steps
  Input. $F_1, \cdots, F_{k_1'} \in G_1$ for $1 \leq k_1' < k$ and $f_1, \cdots, f_{k_2'} \in G_2$ for $1 \leq k_2' < k$.
  Output. The output is interleaved with the following three operations: (1) a generic step in G_1; (2) a generic step in G_2; (3) a bilinear pairing $f_v = e(F_i, F_j)$.
- An **interactive generic algorithm** for G_1 and G_2 behaves like the above non-interactive one, except that it also queries oracles, and the non-group element input in the algorithm may depends on the response of the oracles.

In a Generic Model (GM), we assume that the adversary $\mathcal{A}$ is an interactive generic algorithm for a group G_1, or for G_2, or for two groups G_1 and G_2, depending on the settings. With a GM, Nechaev proved the hardness of discrete logarithm in [17]. GM was further extended in [24] to a wider range of problems related to discrete logarithms. By combining GM and ROM (Random Oracle Model), Schnorr proved the IND-CCA security of signed ElGamal Encryption in [21].

2.3 Threshold Secret Sharing and Lagrange Interpolation

We recall Shamir's t out of n secret sharing scheme over $\mathbb{Z}_q$. Given a secret $s \in \mathbb{Z}_q$, we pick $f_1, f_2, \cdots, f_{t-1} \in \mathbb{Z}_q^*$. Let $f(x) = s + f_1 x + \cdots, f_{t-1}x^{t-1}$. Then $s_i = f(i)$ is the i-th share of the secret s for $1 \leq i \leq n$.

This can be easily extended to a group G of order q. Without loss of generality, we assume that G is a multiplicative group. Let g be a generator of G. Give a secret $S \in G$, we pick $F_1, F_2, \cdots, F_{t-1} \in_R G$. Let $F(x) = S \cdot F_1^x \cdot \ldots \cdot F_{t-1}^{x^{t-1}}$. Then $S_i = F(i)$ is the i-th share of the secret S for $1 \leq i \leq n$. If $S = g^s$, $F_1 = g^{f_1}$, $\cdots, F_{t-1} = g^{f_{t-1}}$, then $F(x) = g^{s+f_1 x+\cdots, f_{t-1}x^{t-1}}$, and $S_i = F(i) = g^{f(i)}$.

Given any subset of $t-1$ shares, there is no information about s (resp. S) leaked. Given any subset of t shares, the secret can be easily recovered with Lagrange interpolation polynomial.

Let $(s_1, s_2, \cdots, s_n) \in \mathbb{Z}_q^n$, and $(S_1, S_2, \cdots, S_n) \in G^n$ satisfy that $s_l = f(l)$ and $S_l = F(l)$, for $l = 1, 2, \cdots, n$.

- With any t distinct shares, $(i_1, s_{i_1}), (i_2, s_{i_2}), \cdots, (i_t, s_{i_t})$, the polynomial $f(x)$ is recovered with

$$f(x) = \texttt{LI}[(i_1, s_{i_1}), (i_2, s_{i_2}), \cdots, (i_t, s_{i_t})](x) = \sum_{j=1}^{t} \prod_{l=1, l\neq j}^{t} \frac{x - i_l}{i_j - i_l} s_{i_j}.$$

- With any t distinct shares, $(i_1, S_{i_1}), (i_2, S_{i_2}), \cdots, (i_t, S_{i_t})$, we can recover $F(x)$ with

$$F(x) = \texttt{EXP-LI}[(i_1, S_{i_1}), (i_2, S_{i_2}), \cdots, (i_t, S_{i_t})](x) = \prod_{j=1}^{t} S_{i_j}^{\prod_{l=1, l\neq j}^{t} \frac{x-i_l}{i_j - i_l}}.$$

2.4 Non-Interactive Zero-Knowledge Proof of Equality of Two Discrete Logarithms

Let G be a multiplicative group of prime order q with generators g_1, g_2. Let $DLEQ(g_1, g_2, u_1, u_2)$ be the language of (g_1, g_2, u_1, u_2) such that $\log_{g_1} u_1 = \log_{g_2} u_2$, i.e., (g_1, g_2, u_1, u_2) is a decisional Diffie-Hellman tupple. Chaum and Pedersen gave an interactive zero-knowledge proof system for $DLEQ(g_1, g_2, u_1, u_2)$ in [6]. The non-interactive version of $DLEQ(g_1, g_2, u_1, u_2)$ is given as below [9]. Let $x = \log_{g_1} u_1 = \log_{g_2} u_2$, and $H : G^6 \rightarrow \mathbb{Z}_1$ be a collision-free hash function.

– The prover chooses a random element $s \in \mathbb{Z}_q$, and computes $u'_1 = g_1^s$, $u'_2 = g_2^s$, $c = H(g_1, g_2, u_1, u_2, u'_1, u'_2)$, $w = cx + s \mod q$. The prover sends (u'_1, u'_2, w) to the verifier.
– The verifier computes $c = H(g_1, g_2, u_1, u_2, u'_1, u'_2)$, and checks that $g_1^w = u_1^c u'_1, g_2^w = u_2^c u'_2$ hold.

2.5 One-Time Schnorr Signature

One-time signature requires that signing different messages with different signing/verification keys. Each pair of signing and verification keys is only used once, so the verification key usually constitutes a part of signature. Below describes how a one-time Schnorr signature works in an additive group G of order q with a generator P. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a cryptographic hash function.

SigKeyGen	**Sign**(m, a)	**Verify**(σ, e, m, Y)
signing key: $a \in_R \mathbb{Z}_q$;	Choose $r \in_R \mathbb{Z}_q^*$;	Compute $R = \sigma P + eY$;
verification key: .	Compute $R = rP, e = H(R, m)$;	(σ, e, m, Y) is *valid*
$Y = aP$	Compute $\sigma \equiv r - ae \mod q$;	iff. $e = H(R, m)$
	Output (σ, e, m, Y).	

Tsiounis and Yung [25] and Jakobsson [12] independently proposed to use Schnorr signature in ElGamal encryption to resist chosen-ciphertext attack (CCA). CCA security of the signed ElGamal encryption was proved by Schnorr and Jakobsson in [21]. In this paper, we will also apply one-time Schnorr signature to Boneh and Franklin's basic identity-based encryption, and extend the IND-CCA security to a threshold scenario.

3 ID-Based Threshold Decryption Scheme

3.1 The Non-threshold Version

In [2], Boneh and Franklin first proposed a basic identity-based encryption scheme and proved the basic scheme secure against chosen-plaintext attacks. Then they use FO-transformation [8] to convert the IND-CPA scheme to an IND-CCA one. With FO-transformation, only after a ciphertext is decrypted, can the validity of the ciphertext be checked. FO-transformation cannot be extended in a threshold setting to achieve CCA security. The reason is that if decryption servers submit their shares without checking the validity of the ciphertext, the adversary is able to collect enough shares to recover the plaintext.

A necessity for threshold decryption with CCA security is that ciphertexts should be *publicly checkable*, as suggested by Lim and Lee in [15]. With public checkability of ciphertexts, every one can check the validity of ciphertexts. The public checkability is usually accomplished by a non-interactive zero-knowledge (NIZK) proof. However, in [1] Baek and Zheng (BZ scheme) applied the idea of Boneh et. al's signature in [4] to the Boneh and Franklin's basic identity-based

encryption for public checkability. Kiltz and Galindo [13] (KG scheme) also used a different method for public checkability.

Here we will not review BZ scheme and KG scheme and only point out that (1) the ciphertext consists of 3 elements of G_1 in both BZ and KG schemes; (2) the validity check of a ciphertext needs 2 pairings in BZ scheme and at least 4 pairings in KG scheme.

Next we show how to use one-time Schnorr signature for public checkability of ciphertexts to get a non-threshold ID-based encryption scheme.

Setup$(1^\lambda) \Rightarrow (G_1, G_2, q, e, P, s, P_{pub}, H_1, H_2, H_3)$**.** Here G_1 is an additive group of prime order q with a generator P. G_2 is an multiplicative group of order q. $e : G_1 \times G_1 \rightarrow G_2$ is an bilinear pairing. $s \in \mathbb{Z}_q$ is the master key, and $P_{pub} = sP$. Let $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$, $H_2 : G_2 \rightarrow \{0,1\}^l$. $H_3 : G_1^2 \times \{0,1\}^l \rightarrow \mathbb{Z}_q$ be cryptographic hash functions. The plaintext space is $\{0,1\}^l$.

Der$(s, para, ID) \Rightarrow S_{ID}$**.** The private key of user ID is given by $S_{ID} = sH_1(ID)$.

Enc$(para, ID, M) \rightarrow C$**.** The ciphertext $C = (U, V, \mu, \sigma)$. Here $U = rP$ with $r \in_R \mathbb{Z}_q$, $V = M \oplus H_2\left(e(H_1(ID), P_{pub})^r\right)$, $R = kP$ with $k \in_R \mathbb{Z}_q$, $\mu = H_3(R, U, V)$ and $\sigma = k - r\mu \mod q$.

Decryt$(C, S_{ID}) \rightarrow M$**.** The validity of C is checked in the following way: Compute $R = \sigma P + \mu U$ and check $\mu = H_3(R, U, V)$ holds. If C is valid, the plaintext is recovered with $M = V \oplus H_2(e(S_{ID}, U))$.

The ciphertext in our scheme consists of 4 elements: 2 elements σ, μ come from $\mathbb{Z}_q$ and the other 2 elements from G_1. Our scheme needs only 1.5 scalar multiplication to compute $R = \sigma P + \mu U$ (see [16]).

Here we focus on comparing the computational overhead of our scheme, BZ scheme and KG scheme since the procedure of valid check of ciphertexts is needed in **DecShare**, **ShareVer** and **ShareCom**.

We refer to the recent results in [10,18]. Let G_1 be a subgroup of the additive group of points of an elliptic curve $E/\mathbb{F}_p$. Let k be the embedding degree. Then G_2 is a subgroup of the multiplicative group of Finite field $\mathbb{F}_{p^k}$. Let q is a 160-bit prime.

- If p is 512 bit with $k = 2$ as in [2], the time to compute one Tate pairing (with precomputation) is comparable to a scalar multiplication in G_1 according to [18].
- If p is 160 bit with $k = 6$, the computation of one Tate pairing corresponds to at least 5 scalar multiplications in G_1 according to [10].

The above analysis shows that our scheme has a much more efficient validity check for ciphertexts than BZ scheme and KG scheme. On the other hand, the ciphertext in our scheme saves 192 bits than BZ scheme and KG scheme in case of p 512 bits and $k = 2$. If p is 160 bit with $k = 6$, the ciphertext in our scheme is 160 bits more than BZ and KG schemes.

3.2 The Threshold Version

We extend our non-threshold identity-based encryption scheme to a threshold decryption scheme. In the threshold scheme, we use a new share distribution and share verification algorithms, aiming to reduce the number of pairings used.

Setup$(1^\lambda) \rightarrow (G_1, G_2, q, e, P, s, P_{pub}, H_1, H_2, H_3, H_4)$**.** Here G_1 is an additive group of prime order q with a generator P. G_2 is an multiplicative group of order q. $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing. $s \in \mathbb{Z}_q$ is the master key, and $P_{pub} = sP$. Let $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$, $H_2 : G_2 \rightarrow \{0,1\}^l$, $H_3 : G_1^2 \times \{0,1\}^l \rightarrow G_1$ and $H_4 : G_2^6 \rightarrow \mathbb{Z}_q$ be cryptographic hash functions. The plaintext space is $\{0,1\}^l$. The public parameters are given by $para = \{G_1, G_2, q, e, P, P_{pub}, H_1, H_2, H_3, H_4\}$ and the master key is s.

Der$(s, para, ID) \rightarrow S_{ID}$**.** The private key of user ID is given by $S_{ID} = sH_1(ID)$.

KeyDis$(para, S_{ID}, n, t) \rightarrow (f(i), g, g^{f(i)}, Z)$**.** Pick $f_j \in_R \mathbb{Z}_q$ for $j = 0, 1, \cdots, t-1$, and get $f(x) = f_0 + f_1 x + \cdots + f_{t-1}x^{t-1}$. The algorithm distributes $f(i)$ to decryption server Γ_i as its private key, where $i = 1, 2, \cdots, n$. Compute $g = e(H_1(ID), P)$ and $y_i = g^{f(i)}$, then (g, y_i) is the corresponding verification key. Compute $Z = S_{ID} - f_0 H_1(ID)$ as the public verification key. Let $\Upsilon = \{Z, g, y_1, \cdots, y_n\}$ denote the collection of all verification keys.

KeyVer$(para, ID_i, g, y_i, f(i)) \rightarrow \{valid, \perp\}$**:** If $g = e(H_1(ID_i), P)$ and $y_i = g^{f(i)}$, output "valid", otherwise output "$\perp$".

Enc$(para, ID, M) \rightarrow C$**.** The ciphertext $C = (U, V, \mu, \sigma)$. Here $U = rP$ with $r \in_R \mathbb{Z}_q$, $V = M \oplus H_2\left(e(H_1(ID), P_{pub})^r\right)$, $R = kP$ with $k \in_R \mathbb{Z}_q$, $\mu = H_3(R, U, V)$ and $\sigma = k - r\mu \mod q$.

DecShare$(para, f_i, C) \rightarrow \{\delta_{i,C}, \perp\}$**:** Compute $R = \sigma P + \mu U$ and check the validity of the ciphertext by testing whether $\mu = H_3(R, U, V)$ holds. If no, output $\perp$. Otherwise, Sever Γ_i computes $\tilde{g} = e(H_1(ID), U)$ and $\tilde{y}_i = \tilde{g}^{f(i)}$. At the same time, Sever Γ_i gives the NIZK proof of $DLEQ(g, \tilde{g}, y_i, \tilde{y}_i)$. More precisely,

Server Γ_i chooses a random element $s \in \mathbb{Z}_q$, and computes $u_1' = g^s$, $u_2' = \tilde{g}^s$, $c = H_4(g, \tilde{g}, y_i, \tilde{y}_i, u_1', u_2')$, $w = cf(i) + s \mod q$. Server Γ_i returns $\delta_{i,C} = (\tilde{g}, \tilde{y}_i, u_1', u_2', w)$ to the verifier.

ShareVer$(para, g, y_i, C, \delta_{i,C}) \rightarrow \{valid, \perp\}$**:** Check the validity of the ciphertext C as above. If C is valid, the algorithm checks the validity of the decryption share $\delta_{i,C}$ as follows.

The algorithm computes $c = H_4(g, \tilde{g}, y_i, \tilde{y}_i, u_1', u_2')$, and checks whether $g^w = y_i^c u_1'$ and $\tilde{g}^w = \tilde{y}_i^c u_2'$ hold. If hold, output "valid", otherwise output "$\perp$".

ShareCom$(para, C, \{\delta_{i,C}\}_{i \in \Phi}) \rightarrow \{M, \perp\}$**:** Check the validity of the ciphertext C as above. If C is valid, the algorithm recovers the plaintext with the help of $\{\delta_{i,C}\}_{i \in \Phi}$ as follows.

(1) Compute $d = \texttt{EXP-LI}[(i_1, \tilde{y}_{i_1}), (i_2, \tilde{y}_{i_2}), \cdots, (i_t, \tilde{y}_{i_t})](0)$, where $i_j \in \Phi, j = 1, 2, \cdots, t$.

(2) Compute $M = V \oplus H_2(d \cdot e(Z, U))$ and output M.

The correctness of threshold decryption is justified by

$$\begin{aligned} d \cdot e(Z,U) &= \texttt{EXP-LI}[(i_1,\tilde{y}_{i_1}),(i_2,\tilde{y}_{i_2}),\cdots,(i_t,\tilde{y}_{i_t})](0)\cdot e(Z,U) \\ &= \tilde{g}^{f(0)}\cdot e(S_{ID}-f(0)H_1(ID),U) = e(H_1(ID),U)^{f(0)} \\ &\qquad\qquad e(S_{ID}-f(0)H_1(ID),U) \\ &= e(f(0)H_1(ID),U)e(S_{ID}-f(0)H_1(ID),U) = e(S_{ID},U) \\ &\qquad\qquad = e(H_1(ID),P_{pub})^r. \end{aligned}$$

3.3 Computational Overhead Comparison with BZ Scheme and KG Scheme

Let $a+b+c$ denote that the implementation needs a pairings, b scalar multiplications in G_1 and c exponentiations in G_2. Table 1 give the comparison of our scheme versa BZ scheme and KG scheme.

Table 1. Comparison of our scheme, BZ Scheme and KG scheme

	KeyDis	**KeyVer**	**Enc**	**DecShare**	**ShareVer**	**ShareCom**
Ours	1+1+n	1+0+1	1+2+0	1+1.5+3	0+1.5+4	1+1.5+t
BZ	$n+(t-1)n+0$	1+0+0	1+2+0	5+1+0	4+0+2	2+0+t
KG	$0+3.5n+0$	3+0+0	0+4.5+0	4+2.5+0	8+0+0	$7+3t+0$

Let G_1 be a subgroup of the additive group of points of an elliptic curve $E/\mathbb{F}_p$. Let k be the embedding degree. Then G_2 is a subgroup of the multiplicative group of Finite field $\mathbb{F}_{p^k}$.

Let the security parameter $\lambda = 80$, then q is 160 bits. We have choices for p and k. Take two cases as examples.

(1) p is 512 bits and $k = 2$ as in [2]. Recent results in [18] show that the implementation of 1 Tate pairing needs 2.91ms, 1 scalar multiplication in G_1 3.08ms, and 1 exponentiation in $\mathbb{F}_{p^2}$ 1.92ms. Then 1 exponentiation in G_2 needs $1.92 \cdot 160/1024 \approx 0.3$ms. Here **ms** means **millisecond**.

(2) p is 160 bits and $k = 6$. We evaluate the implementation of 1 Tate pairing, scalar multiplication and exponentiation in $\mathbb{F}_{p^k}$ in terms of multiplication in $\mathbb{F}_p$ according to [10]. Let m denote a **multiplication** and s denote a **square** in $\mathbb{F}_p$.

- From [10], we know that $1pairing \approx 9120m$.
- With signed sliding window method and mixed/affine addition, a scalar multiplication in G_1 needs

$$(4m+4s)(1+|q|)+(8m+3s)(\frac{|q|}{r+2}+2^{r-2}-1), \tag{1}$$

where s denote a square in $\mathbb{F}_p$ and r is the size of windows. Eq.(1)is optimized to $889m+736s$ with $r=4$. Hence 1 scalar multiplication $\approx 889m+736s \leq 1626m$.

– An exponentiation in G_2 needs $\log_2 q/3$ multiplications in $\mathbb{F}_{p^6}$ according to [11]. A multiplication in $\mathbb{F}_{p^6}$ corresponds to 15 multiplications $\mathbb{F}_p$ according to [10]. Hence An exponentiation in $G_2 \approx |q|/3 \cdot 15 = 960m$.

	1 Tate pairing	1 Scalar Multi in G_1	1 Exp in G_2
p 512 bits, $k = 2$	2.91ms	3.08ms	0.3ms
p 160 bits, $k = 6$	$9120m$	$1626m$	$960m$

Now we compare our scheme with BZ and KG schemes in terms of milliseconds when p is 512 bits and $k = 2$ in Table 2, and in terms of multiplications in $\mathbb{F}_q$ when p is 160 bits and $k = 6$ in Table 3.

Table 2. Comparison when p is 512 bit and $k = 2$

$p = 512$ bits, $k = 2$	**KeyDis**	**KeyVer**	**Enc**	**DecShare**	**ShareVer**	**ShareCom**
Ours	$5.99 + 0.3n$	3.21	9.07	8.43	5.82	$7.53 + 0.30t$
BZ	$3.08tn$	2.91	9.07	17.63	12.24	$5.82 + 0.30t$
KG	$10.78n$	8.73	13.86	19.34	23.28	$20.37 + 9.24t$

Table 3. Comparison when p is 160 bits and $k = 6$

$p = 160$ bits, $k = 6$	**KeyDis**	**KeyVer**	**Enc**	**DecShare**	**ShareVer**	**ShareCom**
Ours	$10746 + 960n$	10080	11040	14439	6279	$11559+960t$
BZ	$1626t + 7494n$	9120	11040	47226	38400	$18240+960t$
KG	$5691n$	27360	7317	40545	72960	$63840+2880t$

It is obvious that our ID-based threshold decryption scheme gains advantage over BZ and KG shemes with respect to computational overhead.

4 Security Proof

4.1 From ID-Based Threshold Scheme to Non-ID-Based Threshold Scheme

We first change our ID-based threshold scheme, called `IdThd` scheme, into a non-ID-based threshold scheme, called `Thd` scheme. The difference between the two schemes is that there is no key derivation algorithm **Der**. The public key of user ID is given with $Q \in_R G_1$, instead of $H_1(ID)$, and the private key is $S_{ID} = sQ$. We will not describe `Thd` scheme in details.

Like `IdThd` scheme, we can also define an IND-CCA attack algorithm $\mathcal{A}'$ and its advantage $\mathsf{Adv}^{\mathsf{CCA}}_{\mathtt{Thd}}(\mathcal{A}')$ for `Thd` scheme.

Definition 3. IND-CCA-Game$^{\mathrm{Thd}}_{\mathcal{A}'}(\beta)$. *For an algorithm $\mathcal{A}'$ and a bit $\beta \in \{0,1\}$, the game between a challenger and $\mathcal{A}'$ is as follows.*

Setup. *The challenger runs* **Setup**$(1^k) \rightarrow (para, s, Q, S_{ID})$ *to obtain the public parameters para, the master key s, the public key Q of user ID, and the private key S_{ID} of the user. The challenger gives $para, Q$ to $\mathcal{A}'$.*

Corruption. *$\mathcal{A}'$ corrupts $t-1$ decryption servers: $\Gamma_1, \Gamma_2, \cdots \Gamma_{t-1}$. The challenger runs*

KeyDis$(para, S_{ID}, n, t) \rightarrow (f(i), g^{f(i)})$ *for $i = 1, 2, \cdots, n$. The challenger returns to $\mathcal{A}'$ the private key shares $f(1)$, $f(2)$, $\cdots, f(t-1)$ and all verification keys.*

Decryption Share query 1. *$\mathcal{A}'$ issues a query (C, j), $t \leq j \leq n$, to the decryption oracle. The oracle responds with $\delta_{j,C}$ or $\perp$ by calling the algorithm $DecShare(para, f(j), C)$.*

Challenge. *$\mathcal{A}'$ outputs two plaintexts (M_0, M_1) of equal length. The challenger chooses a bit $\beta \in_R \{0,1\}$ and returns $C^* = (U^*, V^*, \mu^*, \sigma^*)$ to $\mathcal{A}'$. Here $U^* = r^*P$ with $r^* \in_R \mathbb{Z}_q$, $V^* = M_\beta \oplus H_2(e(Q, P_{pub})^{r^*})$, $\mu^* = H_3(R^*, U^*, V^*)$ with $R^* = k^*P, k^* \in_R \mathbb{Z}_q$, and $\sigma^* = k^* - \mu^* r^* \mod q$.*

Decryption Share query 2. *Like* **Decryption Share Query 1***, except that the queried ciphertext C is not the challenge ciphertext C^*.*

Guess. *Algorithm $\mathcal{A}'$ outputs its guess bit β'. It wins the game if $\beta' = \beta$.*

The advantage of an algorithm $\mathcal{A}'$ that outputs $\beta' \in \{0,1\}$ in attacking the `Thd` *scheme with* **IND-CCA-Game**$^{\mathrm{Thd}}_{\mathcal{A}}(\beta)$ *is defined as*

$$\begin{aligned} Adv^{CCA}_{Thd}(\mathcal{A}') &= |\, \boldsymbol{Pr}[IND\text{-}CCA\text{-}Game_{\mathcal{A}'}(1) = 1] - \boldsymbol{Pr}[IND\text{-}CCA\text{-}Game_{\mathcal{A}'}(0) = 1]\,| \\ &= 2\boldsymbol{Pr}[\beta' = \beta] - 1. \end{aligned}$$

Lemma 1. *[1] Suppose there is a t_1-time IND-CCA adversary $\mathcal{A}$ with advantage $Adv^{CCA}_{IdThd}(\mathcal{A})$ for* `IdThd` *scheme, issuing $q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}$ queries to the random oracles H_1, H_2, H_3, H_4 respectively. Then another t_2-time IND-CCA adversary $\mathcal{A}'$ with advantage $Adv^{CCA}_{Thd}(\mathcal{A}')$ for* `Thd` *scheme can be constructed with the help of $\mathcal{A}$, issuing $q_{H_2}, q_{H_3}, q_{H_4}$ queries to the random oracles H_2, H_3, H_4 respectively. We have*

$$Adv^{CCA}_{Thd}(\mathcal{A}') \geq \frac{1}{q_{H_1}} Adv^{CCA}_{IdThd}(\mathcal{A}), \quad t_2 = t_1 + \max(q_E, q_{H_1})O(\lambda^3).$$

The proof of the lemma follows the idea in [2]. We refer to [1] for details.

4.2 The Plaintext Awareness of $\mathcal{A}'$

In the general model, we consider $\mathcal{A}'$ as a generic interactive algorithm for G_1 and G_2 in **IND-CCA-Game**$^{\mathrm{Thd}}_{\mathcal{A}'}(\beta)$. The input of $\mathcal{A}'$ is partitioned into elements in G_1, elements in G_2 and non-group elements.

- $P, Q = wP, P_{pub} = sP, Z = zP, U^* = r^*P \in G_1$.
- $g, y_1, \cdots, y_n \in G_2$, where $g = e(Q, P)$, $y_i = e(Q, P)^{f(i)}$ for $i = 1, 2, \cdots, n$.
- Non-group elements like $f(1), f(2), \cdots, f(t)$, $\mathcal{A}'$'s random coins, etc.

With the generic model, $\mathcal{A}'$ can use group elements in G_1 and G_2 only for generic group operation in G_1, and group operation in G_2, bilinear mappings from G_1^2 to G_2, equality tests and queries to oracles.

Let us first consider a k-step non-interactive generic algorithm over G_1 and G_2 with a bilinear mapping $e : G_1^2 \to G_2$. Suppose among the k steps, there are k_1 generic steps in G_1, k_2 generic steps in G_2 and $k - k_1 - k_2$ pairings. We can eliminate the $k - k_1 - k_2$ pairings with the following k_2 generic steps in G_2, due to the bilinear property of pairings.

Precomputation. $g_1 = e(P, P), g_2 = e(P, Q) = g_1^w, g_3 = e(P, P_{pub}) = g_1^s, g_4 = e(P, Z) = g_1^z, g_5 = e(P, U^*) = g_1^{r^*}$, $g_6 = e(Q, P_{pub}) = g_1^{ws}$, $g_7 = e(Q, Z) = g_1^{wz}$, $g_8 = e(Q, U^*) = g_1^{wr^*}$, $g_9 = e(P_{pub}, Z) = g_1^{sz}$, $g_{10} = e(P_{pub}, U^*) = g_1^{sr^*}, g_{11} = e(U^*, Z) = g_1^{zr^*}$.

k_2 **generic steps.** Let $(g_1, \cdots, g_{11}, y_1, \cdots, y_n)$ be the input and follows k_2 generic steps in G_2. From now on we use g_2, instead of g, to denote $e(P, Q)$.

Consequently, a k-step non-interactive generic algorithm over G_1 and G_2 can be reduced to two non-interactive generic algorithms.

(1) A k_1-step generic algorithm in G_1 with input group elements $P, Q = wP, P_{pub} = sP, Z = zP, U^* = r^*P$ and non-group elements $f(1), \cdots, f(t-1)$ and the random coins chosen by the algorithm. It computes group elements $F_1, F_2, \cdots, F_{k_1}$.

(2) A k_2-step generic algorithm in G_2 with input group elements $g_1, \cdots, g_{11}, y_1, \cdots, y_n$ and non-group elements $f(1), \cdots, f(t-1)$ and the random coins chosen by the algorithm. It computes group elements $f_1, f_2, \cdots, f_{k_2}$.

Lemma 2. *In* **IND-CCA-Game**$^{\mathtt{Thd}}_{\mathcal{A}'}(\beta)$*, for a k-step non-interactive generic algorithm for G_1 and G_2 with a bilinear mapping $e : G_1^2 \to G_2$, non-trivial collisions among $F_1, F_2, \cdots, F_{k_1}$ occurs with a probability at most* $\binom{k_1}{2}/q$*, and non-trivial collisions among $f_1, f_2, \cdots, f_{k_2}$ occurs with a probability at most* $2\binom{k_2}{2}/q$*, where $k_1 + k_2 \leq k$.*

Note that if a non-trivial collision $F_i = F_j$ in G_1 occurs, we can also find a non-trivial collision $f_{i'} = f_{j'}$ in G_2 due to the bilinear mapping from G_1^2 to G_2.

Lemma 3. *Let hash functions in* `Thd` *scheme be random oracles. In the IND-CCA-Game*$^{\mathtt{Thd}}_{\mathcal{A}'}(\beta)$ *game, suppose that the adversary algorithm $\mathcal{A}'$ is a k-step interactive generic algorithm. Then $\mathcal{A}'$ wins the game with advantage at most* $\frac{2k^2}{q}$.

The proof of Lemma 2 and 3 is omitted due to limited space and will appear in the full version of the paper. Combining Lemma 1 and Lemma 3, we get the following theorem.

Theorem 1. *Let hash functions in* `IdThd` *scheme are random oracles. In the IND-CCA-Game*$^{\texttt{IdThd}}_{\mathcal{A}}(c)$ *game, suppose that the adversary algorithm* $\mathcal{A}$ *is a* k*-step interactive generic algorithm. Then* $\mathcal{A}$ *wins the game with advantage at most* $\frac{2k^2}{q}q_{H_1}$*, where* q_{H_1} *is number of hash queries to* H_1*.*

5 Conclusion

Schnorr proved that the signed ElGamal encryption is secure in the Random Oracle and Generic Model (ROM+GM) in [21]. Boneh and Franklin's basic identity-based encryption scheme is an ElGamal-like scheme. In this paper, we applied one-time Schnorr signature to BF scheme to achieve public checkability of ciphertexts, hence obtained a scheme with IND-CCA security. We extended the ROM+GM model to an threshold settings and proved that our identity-based threshold decryption scheme is secure against chosen-ciphertext attacks. Our scheme is more efficient than BZ scheme and KG scheme, which are the recent results in [1,13]. The bandwidth of our scheme is smaller or bigger than BZ scheme and KG scheme, depending on the choices of different super-singular elliptic curves.

Acknowledgement. We thank the anonymous referees for the suggestions to improve the paper.

References

1. J. Baek and Y. Zheng, *Identity-Based Threshold Decryption*, Public Key Cryptography, Proceedings of PKC 2004, Lecture Notes in Computer Science, LNCS 2947, pp. 262-276, Springer-Verlag, 2004.
2. D. Boneh and M. Franklin, *Identity-Based encryption from the Weil pairing*, In: Kilian J, ed. Advances in CryptologyCrypto'2001. Berlin, Heidelberg: Springer-Verlag, pp.213-229, 2001.
3. D. Boneh, B. Lynn, H. Shacham, *Short signatures from the Weil pairing,* Advances in Cryptology – Asiacrypt'2001, LNCS 2248, Springer-Verlag, pp. 514–532, 2001.
4. D. Boneh, B. Lynn and H. Shacham, *Short signatures from the weil pairing*, Asciacrypt 2001, LNCS 2248, pp. 566-582, Springer-Verlag, 2001.
5. C. Cocks, *An identity based encryption scheme based on quadratic residues*, In Proceedings of the 8th IMA International Confreence on Cryptography and Coding, pp.8-26, 2001.
6. D. Chaum and T. Pedersen, *Wallet databases with observers,* In Proc. CRYPTO92, volume 740 of LNCS, pages 89C105. Springer-Verlag, 1992.
7. Y. Dodis and M. Yung, *Exposure-resilience for free: the hierarchical id-based encryption case*, Proceedings of IEEE security in Storage Workshop 2002. pp. 45-52, 2002.
8. E. Fujisaki and T. Okamoto, Secure Integration of Asymmetric and Symmetric Encryption Schemes, Advanced in Crypto'99, LNCS 1666, pp. 537-554, Springer-Verlag, Berlin, 1999.

9. A. Fiat, and A. Shamir, *How to Prove Yourself: Practical Solution to Identification and Signature Problems.* Advances In CRYPTO'86, pp.186-189, Springer-Verlag, 1987.
10. R. Granger, D. Page, and N.P. Smart, "High Security Pairing-Based Cryptography", Proc. ANTS-V, Springer LNCS 4096, pp 480-494, 2006.
11. p L.C.K. HUI, and K.-Y LAM, "Fast square-and-multiply exponentiation for RSA", Electron. Lett., 1994, 30, (17), pp. 1396- 1397.
12. M. Jakobsson. *A practical mix.* In K. Nyberg, editor, EUROCRYPT '98, LNCS 1403, pp.448–461, Springer-Verlag, 1998.
13. E. Kiltz, D. Galindo, *Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation without Random Oracles*, iacr.org/2006/034.ps.gz.
14. B. Libert and J. Quisquater, *Efficient revocation and threshold pairing based cryptosystems*, Principles of Distributed Computing, 2003.
15. C. H. Lim and P. J. Lee. Another method for attaining security against adaptively chosen ciphertext attacks. In Advances in Cryptology–Crypto'93, LNCS 773. SpringerVerlag, 1994.
16. W. Mao, *Modern Cryptography: Theory and Practice*, Prentice Hall PTR, pp. 528-530, 2004.
17. V. I. Nechaev, *Complexity of a determinate algorithm for the discrete logarithm.* Mathematical Notes 55, pp. 165-172, 1994.
18. M. Scott, "Implementing cryptographic pairings", 10th workshop on Elliptic Curve Cryptography (ECC 2006), September 18-20, 2006. http://www.cacr.math.uwaterloo.ca/conferences/2006/ecc2006/slides.html
19. A. Shamir, *Identity-based cryptosystems and siganture schemes.* In Advances in Cryptology - CRYPTO 1984, LNCS 196, pp.47-53, Springer-Verlag, 1984.
20. A. Shamir, *How to share a secret,* Communications of the ACM, No. 22, pp.612-613, 1979.
21. C. P. Schnorr and M. Jacobsson, *Security of signed ElGamal encryption*, Asciacrypto'2000, LNCS 1976, pp. 73-88, Springer-Verlag, 2000.
22. V. Shoup and R. Gennaro, *Securing Threshold Cryptosystems against Chosen Ciphertext Attacks.* Eurocrypt'98, LNCS 1404, pp. 1-16, 1998.
23. V. Shoup and R. Genarro, *Securing Threshold Cryptosystems against Chosen Ciphertext Attacks*, J. Cryptology, 15 (2)75-96, 2002.
24. V. Shoup, *Lower bounds for discrete logarithms and related problems*, Eurocrypt'2000, LNCS 1807, pp. 256-266, 1997.
25. Y. Tsiounis and M. Yung. *On the Security of El Gamal based Encryption.* In PKC '98, LNCS 1431, pp. 117-134, Springer-Verlag, 1998.

Visual Cryptography Schemes with Dihedral Group Access Structure for Many Images

Miyuki Uno and M. Kano

Department of Computer and Information Sciences,
Ibaraki University, Hitachi, 316-8511, Japan
umyu@mug.biglobe.ne.jp,
kano@mx.ibaraki.ac.jp
http://gorogoro.cis.ibaraki.ac.jp

Abstract. A new construction of visual cryptography scheme (VCS) with dihedral group access structure for two shares and many secret images is proposed. Let D_{2n} be a dihedral group of order $2n$, and let $\{Image(\tau) \mid \tau \in D_{2n}\}$ be $2n$ secret images corresponding to D_{2n}. In a VCS with dihedral group access structure, two shares (two transparencies) A and B are constructed so that for any element τ of D_{2n}, A and $\tau(B)$ reconstruct the secret image $Image(\tau)$. This new VCS is perfect and has contrast $1/(6n^2)$.

Keywords: visual cryptography, VCS, visual secret sharing, VSS, dihedral group, many secret images.

1 Introduction

A visual cryptography scheme (VCS), which was proposed by Shamir and Naor [5] ([4]), is a method of encoding a secret image into some shares, which are usually printed on transparencies. In k-out-of-n VCS, the secret image is encoded into n shares. If any k set of n shares are stacked together, then the original secret image is reconstructed, but any set of shares less than k does not leak any information about the secret image.

Droste [1] introduced the following new VCS and gave its construction. Let $\mathcal{F}$ be a family of non-empty subsets of $\{1, 2, \ldots, n\}$, and $\{Image(A) \mid A \in \mathcal{F}\}$ be a set of $|\mathcal{F}|$ different secret images. Then we can make n shares $S(1), S(2), \ldots, S(n)$ so that for any element $A \in \mathcal{F}$, a stack of the shares in $\{S(i) \mid i \in A\}$ reconstructs the secret image $Image(A)$, and we cannot get any information on $Image(B)$ from the set of transparencies if $B \not\subseteq A$. This kind of VCS for many secret images has been studied in some papers including [3], [6], [2].

In this paper we consider a VCS with dihedral group access structure, which is defined below, and give its construction. Let

$$D_{2n} = \{1, \alpha, \ldots, \alpha^{n-1}, \beta, \beta\alpha, \ldots, \beta\alpha^{n-1}\}$$

be a *dihedral group* of order $2n$, where α denotes a rotation with angle $2\pi/n$ and β denotes a horizontal reversion. Let $\{Image(\tau) \mid \tau \in D_{2n}\}$ be a set of $2n$

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 344–359, 2007.

secret images, each of which corresponds to an element of D_{2n} and is comprised of black and white pixels. Then two shares A and B, which are printed on transparencies, are constructed so that for any element $\tau \in D_{2n}$, by staking A and $\tau(B)$, the secret image $Image(\tau)$ is obtained (see Figure 1). This VCS is called a *VCS with dihedral group access structure* for two shares and $2n$ secret images. We give a new construction of this *perfect* VCS with contrast $1/(6n^2)$, where a perfect VCS means that a black pixel of a secret image is reconstructed into a pure black region, while a white pixel is translated into a region consisting of white and black subpixels.

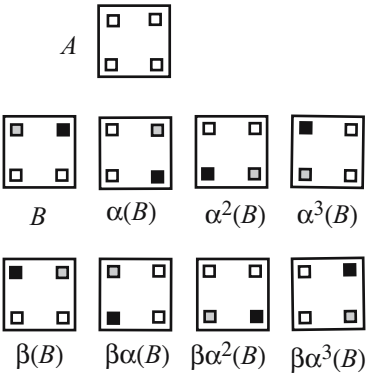


Fig. 1. Two shares A and B of VCS with D_8 access structure

This paper is organized as follows: In Sect. 2, a construction of a VCS with D_{2n} access structure is given, but it has a contrast $1/(8n^2)$. In Sect. 3, we give a revised construction of a perfect VCS with reverse access structure, which is a VCS with $\{1, \beta\}$ access structure. In Sect. 4, by using the construction given in Sect. 3 we obtain an improved construction of a perfect VCS with D_{2n} access structure, whose contrast is $1/(6n^2)$. In appendix, an example of the improved VCS with D_4 access structure is given.

2 A Construction of VCS with Dihedral Group Access Structure

In this section we give a construction of a VCS with dihedral group D_{2n} access structure for two shares and $2n$ secret images. It has contrast $1/(8n^2)$ though an improved VCS given latter has contrast $1/(6n^2)$. Let D_{2n} be a dihedral group defined in Section 1, which is generated by the rotation α with angle $2\pi/n$ and the horizontally reversion β. Let A and B be two shares, and let $\{Image(\tau) \mid \tau \in D_{2n}\}$ be a set of $2n$ distinct secret images, that is, given $2n$ secret images are assigned to the elements of D_{2n}.

We first define two 2×2 matrices $R1$ and $R2$, which are randomly chosen from the following matrices according to the color of a pixel in an image.

$$R1 \in \left\{ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}, \quad R2 \in \left\{ \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \right\}.$$

Let x and y be pixels of the shares A and B, respectively, such that x and y are in the same position when we stack A and B. Then $\{\tau(x) \mid \tau \in D_{2n}\}$ and $\{\tau(y) \mid \tau \in D_{2n}\}$ are the sets of $2n$ pixels of A and B, respectively, such that they have the same position in some $A + \tau(B)$, $\tau \in D_{2n}$ (Figures 2, 3). Notice that in the following figures, every pixel and subpixel are rectangular, but this condition is not necessary. Actually for some VCS with D_{2n} access structure, triangles and other figures can be used.

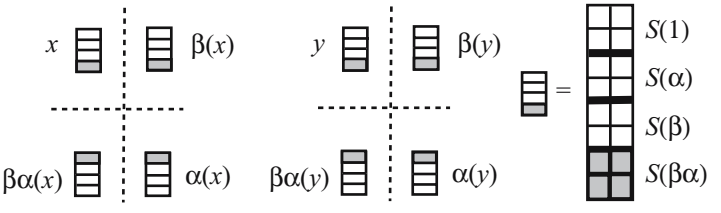


Fig. 2. A construction of VCS with D_4 access structure, where x and y denote pixels and are split into 16 subregions each

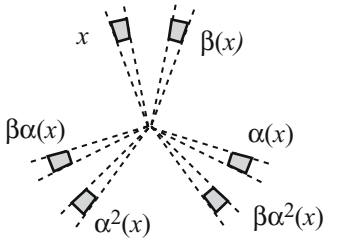


Fig. 3. A construction of VCS with D_6 access structure

Each pixel of A and B is first split into $4n^2$ *subregions*, and latter each of these subregions is split into two *subpixels*, and so finally original pixel is split into $8n^2$ subpixels. The dihedral group D_{2n} acts on the set of subregions of $\{\tau(x) \mid \tau \in D_{2n}\}$, which contains $2n \cdot 4n^2$ subregions. Every orbit of this permutation has length $2n$, and there are $4n^2$ orbits. We divide these $4n^2$ orbits into $2n$ disjoints subsets, each of which contains $2n$ orbits, and label them $\{S(\tau) \mid \tau \in D_{2n}\}$.

Example 1: Our construction of VCS with D_4 access structure, where $D_4 = \{1, \alpha, \beta, \beta\alpha\}$, will be given. In Figure 4, (1) denotes the set of $2n \cdot 4n^2 = 16 \cdot 2^2$ subregions, which consists of $4 \cdot 2^2$ orbits, and each orbit contains four subregions.

We mark some subregions with some symbols to emphasis its orbit. Using this example, we explain the rule of determining the color of each subpixel. In order to do so easily, we rearrange all the subregions so that each orbit consists of the subregions lying on the same position (see (2)). Moreover, these orbits are partitioned into four subsets $S(1), S(\alpha), S(\beta), S(\beta\alpha)$. Of course, there is a bijection between the subregions of (1) and those of (2), and thus if we determine the colors of subregions of (2), the colors of original subregions are determined. Let $\tau \in D_4$. For any element $\rho \in D_4$, choose one subregion $sub(\rho, x, \tau)$ from $\rho(x) \cap S(\tau)$ so that the four subregions $sub(\rho, x, \tau), \rho \in \{1, \alpha, \beta, \beta\alpha\}$, are contained in four distinct orbits of $S(\tau)$. Furthermore, for any fixed ρ, we can choose $sub(\rho, x, \tau)$ so that it is placed at the same position in every $S(\tau), \tau \in D_4$ (see (2)).

For a pixel y of B, which is placed at the same position as x of A, we first split it into $16 \cdot 2^2$ subregions. Then for any $\gamma, \rho \in D_4$, choose a subpixel $sub(\rho, y, \gamma)$ from $\rho(y) \cap S(\gamma)$ of B that is placed at the same positions as

$$sub(\gamma^{-1}\rho, x, \gamma) \quad \text{of } A \quad \text{(see (3))}.$$

Thus, by $\tau \in D_4$, the subregion $sub(\rho, y, \tau)$ in B is moved to the following subregion in $\tau(B)$:

$$sub(\tau\tau^{-1}\rho, x, \tau) = sub(\rho, x, \tau),$$

which is in the same position of $sub(\rho, x, \tau)$ in A. Namely, the subregions $sub(\rho, y, \tau)$ in B and $sub(\rho, x, \tau)$ in A are placed at the same position in $A + \tau(B)$, and so they are used to make a color of the pixel $\rho(z)$ of $Image(\tau)$, which is reconstructed by $A + \tau(B)$.

Consider the secret image $Image(\tau)$. We determine the basis matrix of $sub(\rho, x, \tau)$ and $sub(\rho, y, \tau)$, $\rho \in D_{2n}$, by using matrices $R1$ and $R2$ as follows according to the color of a pixel $\rho(z)$ of $Image(\tau)$, which is placed at the same position as $\rho(x)$.

$$\begin{bmatrix} sub(\rho, x, \tau) \\ sub(\rho, y, \tau) \end{bmatrix} = \begin{cases} R1 & \text{if } \rho(z) \text{ is black,} \\ R2 & \text{if } \rho(z) \text{ is white,} \end{cases}$$

and all the non-chosen subregions are $[1, 1]$,

Notice that all the subregions are split into two subpixels each. We repeat the same procedure for every non-chosen pixel $z \in Image(\tau)$ and for all secret images until the colors of all the subpixels of shares A and B are determined. By the definition of $sub(\rho, x, \tau)$ and $sub(\rho, y, \gamma)$, when we stack A and $\tau(B)$, $Image(\tau)$ is reconstructed and its contrast is $1/32$ since each pixel is finally split into 32 subpixels.

Now we explain our construction of VCS with general dihedral group D_{2n} access structure. Each pixel of shares A and B is split into $4n^2$ subregions, and for every $\rho \in D_{2n}$, one subregion $sub(\rho, x, \gamma)$ is chosen from $\rho(x) \cap S(\gamma)$ of A as Example 1, and $sub(\rho, y, \gamma)$ is the subregion placed at the position as $sub(\tau^{-1}\rho, x, \gamma)$. For any image $Image(\tau)$ and any element $\rho \in D_{2n}$, let $\rho(z)$ be

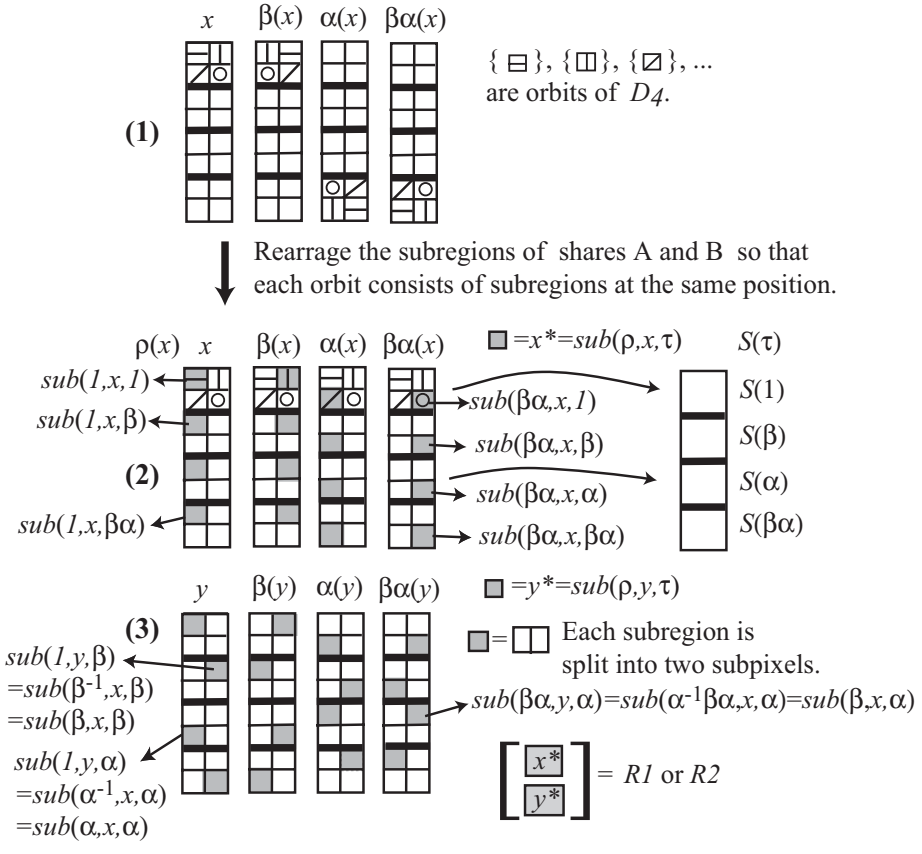


Fig. 4. A construction of VCS with D_4 access structure. Gray squares denote $sub(\rho, x, \tau)$ of A and $\rho(\rho, y, \tau)$ of B.

a pixel of $Image(\tau)$, and let $\rho(x)$ and $\rho(y)$ be the pixels of A and B being in the same position of $\rho(z)$. We determine $sub(\rho, x, \tau)$ and $sub(\rho, y, \tau)$ as

$$\begin{bmatrix} sub(\rho, x, \tau) \\ sub(\rho, y, \tau) \end{bmatrix} = \begin{cases} R1 & \text{if } \rho(z) \text{ is black,} \\ R2 & \text{if } \rho(z) \text{ is white,} \end{cases}$$
$$\text{and all the non-chosen subregions are } [1, 1],$$

where all the subregions are split into two subpixels each.

By the definition of $sub(\rho, x, \tau)$ and $sub(\rho, y, \tau)$, the color of the region $\rho(z)$ in $A + \tau(B)$ is determined by $sub(\rho, x, \tau)$ and $\tau(sub(\rho, y, \tau))$, and thus $Image(\tau)$ is reconstructed. It is easy to see that its contrast is $1/(8n^2)$ and we cannot get any information about the secret images from one of $\{A, B\}$ since (i) each $sub(\rho, x, \tau)$ in A is $[0, 1]$, $[1, 0]$ or $[1, 1]$, (ii) $[1, 1]$ means that the color of the pixel is determined by other subregion, and (iii) there is no difference between

white pixel and black pixel of the image; and the same conditions hold for B. Consequently we can construct a VCS with dihedral group access structure.

3 A Revised Construction of VCS with Reverse Access Structure

We give a new construction of VCS with reverse access structure, which will play important role in the next section. Our method of constructing VCS with reverse access structure is different from the method given in the previous section. The contrast of our construction is 1/6, but that of the preceding construction is 1/8. Moreover, it will be shown that the construction given here is best possible in some sense. Namely, we will prove in the appendex that it is impossible to construct a perfect VCS with reverse access structure with contrast 1/5 or more.

For a share X, we briefly denote by $\tilde{X}$ the share $\beta(X)$, which is obtained by horizontally reversing X (Figures 5, 6). Suppose that two distinct secret images $Image1$ and $Image2$ are given. We want to encode these two secret images into two shares A and B so that we can reconstruct $Image1$ and $Image2$ by stacking A and B, and by A and $\tilde{B}$, respectively (see Figures 5, 6). We call this VCS a *VCS with reverse access structure*.

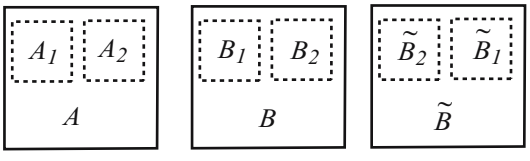


Fig. 5. A VCS with reverse access structure

We now explain our construction of VCS with reverse access structure. Let A_1 and A_2 be two pixels of A, and B_1 and B_2 be the two pixels of B such that A_i covers B_i in orderly stacking, and A_i covers $\tilde{B}_j$ in reversely stacking, where $\{i, j\} = \{1, 2\}$ (Figures 5, 6). We split each of these pixels into six subpixels. Then A_i and B_i are expressed as $A_i = (x_{ij})$ and $B_i = (y_{ij})$, where x_{ij} and y_{ij} denote subpixels. For convenience, we also regard $A_i = (x_{ij})$ and $B_i = (y_{ij})$ as their basis matrices, that is, we assume that x_{ij} and y_{ij} express subpixels and their colors $x_{ij}, y_{ij} \in \{0, 1\} = \{white, black\}$. Let us write $A_i = (x_{ij})$ and $B_i = (y_{ij})$ as Figure 6, where the suffixes of A_2 and B_2 are reversed.

If we orderly stack A and B, then the subpixels of the resulting region are

$$x_{ij} + y_{ij} \quad \text{for} \quad 1 \le i \le 2, \quad 1 \le j \le 6,$$

where $+$ denotes OR of two elements, and if we reverse B and stack it and A together, then the the subpixels of the resulting region are

$$x_{ij} + y_{i'j}, \quad \text{where} \quad \{i, i'\} = \{1, 2\}, \quad 1 \le j \le 6.$$

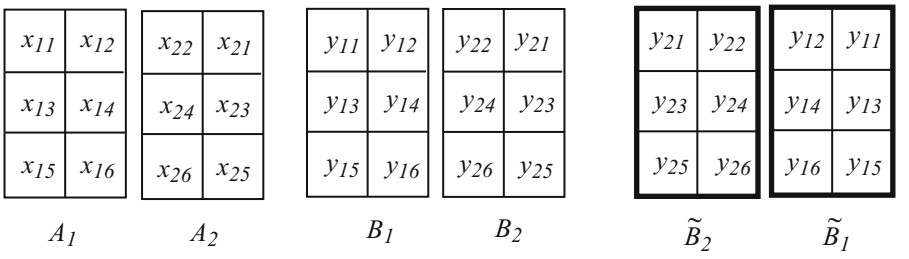


Fig. 6. Subpixels of A, B, and $\tilde{B}$

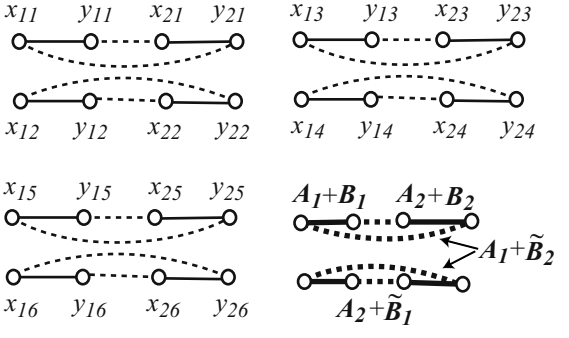


Fig. 7. The diagram representing the pairs of " + "

The pairs of this operation " + " between elements of A_i and those of B_i or $\tilde{B}_{i'}$ are represented by the diagram given in Figure 7. Thus, for example, if the pixel of $Image1$ placed at A_1 is black, then the basis matrices should satisfy

$$x_{11} + y_{11} = x_{12} + y_{12} = x_{13} + y_{13}$$
$$= x_{14} + y_{14} = x_{15} + y_{15} = x_{16} + y_{16} = 1$$

because our VCS is perfect. Similarly, if the pixel of $Image2$ placed at A_1 is white, then at least one of the following six elements is equal to 0 because the contrast of our VCS is 1/6.

$$x_{11} + y_{21},\ x_{12} + y_{22},\ x_{13} + y_{23},$$
$$x_{14} + y_{24},\ x_{15} + y_{25},\ x_{16} + y_{26}.$$

We now define the three 2×2 matrices as follows:

$$M1 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix},\ M2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix},\ \text{and}\ \ M3 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{1}$$

Then we tentatively define the two basis matrices A_1 and A_2 as follows:

$$[A_1, A_2] = \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \\ x_{14} & x_{24} \\ x_{15} & x_{25} \\ x_{16} & x_{26} \end{bmatrix} = \begin{bmatrix} M1 \\ M2 \\ M3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2}$$

We next define the matrices B_1 and B_2 according to the set of colors $\{A_1+B_1,$ $A_2+\tilde{B}_1,\ A_2+B_2,\ A_1+\tilde{B}_2\}$. For example, if their colors are

$$\begin{aligned} &A_1 + B_1 = black,\ A_2 + \tilde{B}_1 = black, \\ &A_2 + B_2 = white,\ A_1 + \tilde{B}_2 = black, \end{aligned} \tag{3}$$

then by considering Figure 7, we define

$$[B_1, B_2] = \begin{bmatrix} y_{11} & y_{21} \\ y_{12} & y_{22} \\ y_{13} & y_{23} \\ y_{14} & y_{24} \\ y_{15} & y_{25} \\ y_{16} & y_{26} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} M2^* \\ M1^* \\ M3 \end{bmatrix},$$

where Mi^* is obtained from Mi only by exchanging the first and the second rows. Then it is clear that these matrices satisfy the color condition (3).

In fact, when we distribute subpixels, we randomly permute the six rows of (A_1, A_2) and (B_1, B_2) simultaneously. Thus there are no difference between the above two matrices $[A_1, A_2]$ and $[B_1, B_2]$.

We shall show that for any set of colors, we can define the basis matrices B_1 and B_2 that satisfy the given color condition and posses the following properties: (i) (B_1, B_2) consists of the three matrices choosing one from each $\{M1, M1^*\}, \{M2, M2^*\}, \{M3, M3*\}$; (ii) the VCS is perfect; and (iii) the VCS has contrast $1/6$. By the property (i) and symmetry, our construction guarantees that the VCS is secure, that is, we cannot get any information about secret images from one of $\{(A_1, A_2),\ (B_1, B_2)\}$.

We prove that for any set of colors, we can always define the matrices (B_1, B_2) possessing the above properties. We consider the following three cases.

Case 1. *Two consecutive colors in* $(A_1 + B_1,\ A_2 + \tilde{B}_1,\ A_2 + B_2,\ A_1 + \tilde{B}_2)$ *are white.*

Suppose first $A_1 + B_1 = A_2 + \tilde{B}_1 = white$. In this case we define the first and second rows of (B_1, B_2) as follows:

$$[B_1, B_2] \supset \begin{bmatrix} y_{11} & y_{21} \\ y_{12} & y_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = M2,$$

Then $A_1 + B_1 = A_2 + \tilde{B}_1 = white$, and the other colors $A_2 + B_2$ and $A_1 + \tilde{B}_2$ are determined by the remaining rows of (B_1, B_2). We define the remaining rows of (B_1, B_2) as follows according to $(A_2 + B_2,\ A_1 + \tilde{B}_2) = (black, black)$, $(black, white)$, $(white, black)$, $(white, white)$.

$$[B_1, B_2] \supset \begin{bmatrix} y_{13}\ y_{23} \\ y_{14}\ y_{24} \\ y_{15}\ y_{25} \\ y_{16}\ y_{26} \end{bmatrix} = \begin{bmatrix} 1\ 1^* \\ 0\ 0 \\ 1\ 0 \\ 1\ 1^* \end{bmatrix},$$

$$\begin{bmatrix} 0\ 0^* \\ 1\ 1 \\ 1\ 0 \\ 1\ 1^* \end{bmatrix}, \begin{bmatrix} 1\ 1^* \\ 0\ 0 \\ 1\ 1 \\ 1\ 0^* \end{bmatrix}, \text{ or } \begin{bmatrix} 0\ 0^* \\ 1\ 1 \\ 1\ 1 \\ 1\ 0^* \end{bmatrix},$$

where only 0^* and 1^* guarantee the desired colors, and the remaining elements are determined so that two matrices coming from each of $\{M1,\ M1^*\}$ and $\{M3,\ M3^*\}$ appear.

By the symmetry of the diagram in Figure 7, we can similarly construct the desired basis matrices (A_1, A_2, B_1, B_2) in the other cases. For example, in the case of $A_2 + \tilde{B}_1 = A_2 + B_2 = white$, $A_1 + \tilde{B}_2 = X$, and $A_1 + B_1 = Y$, where $X, Y \in \{white, black\}$, we first define $\tilde{B}_1$ and $\tilde{B}_2$ as (2) then define the remaining matrix A_2 and A_1, which correspond to B_1 and B_2 in the above construction. Hence in this case we obtain the desired basis matrices A_1, A_2, B_1, B_2.

Case 2. *Two consecutive colors in* $(A_1 + B_1,\ A_2 + \tilde{B}_1,\ A_2 + B_2,\ A_1 + \tilde{B}_2)$ *are* $(white, black)$.

Suppose first $A_1 + B_1 = white$ and $A_2 + \tilde{B}_1 = black$. We define (A_1, A_2) by (2), and B_1 by

$$B_1 = \begin{pmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \\ y_{16} \end{pmatrix} = \begin{pmatrix} 1 \\ y_{12} \\ 0 \\ y_{14} \\ y_{15} \\ 1 \end{pmatrix}.$$

Then $A_1 + B_1 = white$ and $A_2 + \tilde{B}_1 = black$. We can easily determined the matrix B_2 and the remaining elements y_{12}, y_{14}, y_{15} of B_1 so that the desired colors of $A_2 + B_2$ and $A_1 + \tilde{B}_2$ are reconstructed and the three matrices $M1$, $M2$ and $M3$ appear.

By the symmetry of the diagram in Figure 7, we can similarly construct the desired basis matrices (A_1, A_2, B_1, B_2) in the other cases.

Case 3. $A_1 + B_1 = A_2 + \tilde{B}_1 = A_2 + B_2 = A_1 + \tilde{B}_2 = black$.

The (A_1, A_2) of (2) and the following (B_1, B_2) have the desired colors and properties.

$$(B_1, B_2) = \begin{bmatrix} 1\,1 \\ 0\,0 \\ 1\,0 \\ 1\,1 \\ 0\,1 \\ 1\,1 \end{bmatrix}.$$

Since Cases 1,2,3 covers all the cases, the proof is complete. Therefore we can construct a perfect VCS with reverse access structure with contrast $1/6$.

4 An Improved Construction of VCS with Dihedral Group Access Structure

We present an improved construction of VCS with dihedral group access structure for two shares and $2n$ secret images by applying the VCS with reverse access structure given in the previous section. Let D_{2n} be the dihedral group defined in the preceding sections. In our construction the contrast is $1/(6n^2)$.

First let $\Omega = \{(1, \beta), (\alpha, \beta\alpha), \ldots, (\alpha^{n-1}, \beta\alpha^{n-1})\}$ be the set of pairs of elements in D_{2n}. We construct a VCS in such a way that we apply the construction of VCS with reverse access structure to two secret images $Image(\rho)$ and $Image(\beta\rho)$ and some regions of two shares A and $\rho(B)$ for every $(\rho, \beta\rho) \in \Omega$. Namely, we split each pixel of shares A and B into $2n$ subregions, then for every $(\rho, \beta\rho) \in \Omega$, we encode two images $Image(\rho)$ and $Image(\beta\rho)$ into one subregion of every pixel of A and B so that $A + \rho(B)$ and $A + \beta\rho(B)$ reconstruct $Image(\rho)$ and $Image(\beta\rho)$, respectively. We begin with an example of this construction before giving a construction in general case.

Example 2: We construct a VCS with D_4 access structure. Let $D_4 = \{1, \alpha, \beta, \beta\alpha\}$. Then $\Omega = \{(1, \beta), (\alpha, \beta\alpha)\}$. Let x and y be pixels of the shares A and B, respectively, such that x and y are in the same position when we stack A and B (Figure 8).

We split every pixel in $\{\rho(x) \mid \rho \in D_4\}$ of A into four subregions. Then D_4 acts on the set of these subregions, and every orbit of this permutation has length four. We choose two subregions from each orbit such that they are transformed each other by β and exactly two subregions are chosen from each pixel, and denote these subregions by $sub(i, x), \beta(sub(i, x))$ $(1 \le i \le 4)$ (Figure 8). We also choose eight subregions $sub(i, y), \beta(sub(i, y))$ $(1 \le i \le 4)$ that are placed at the same positions as

$$\begin{aligned} &sub(1, x),\ \beta(sub(1, x)), \\ &sub(2, x),\ \beta(sub(2, x)), \\ &\alpha^{-1}(sub(3, x)),\ \alpha^{-1}\beta(sub(3, x)), \\ &\alpha^{-1}(sub(4, x)),\ \alpha^{-1}\beta(sub(4, x)),\ \text{respectively.} \end{aligned}$$

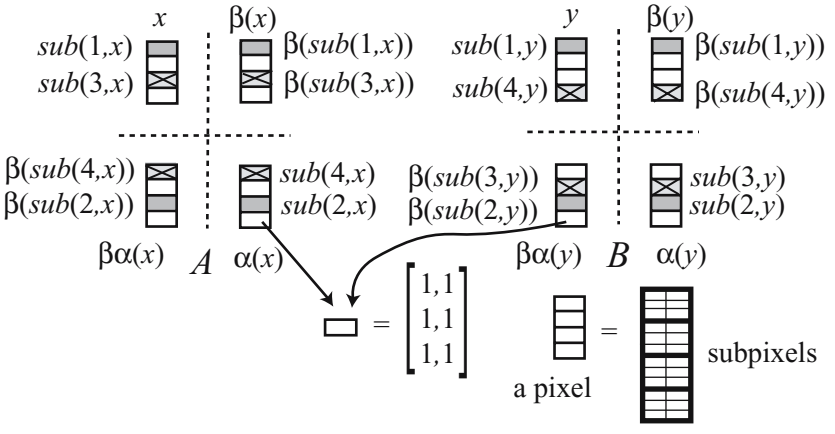


Fig. 8. A construction of VCS with D_4 access structure. Every pixel is split into four subregions, and each subregion is split into six subpixels. Gray subregions reconstruct $Image(1)$ and $Image(\beta)$, and regions with cross reconstruct $Image(\alpha)$ and $Image(\beta\alpha)$.

By using the construction of VCS with reverse access structure, we can encode two secret images $Image(1)$ and $Image(\beta)$ into the eight subregions $sub(1,x), \beta(sub(1,x))$, $sub(2,x)$, $\beta(sub(2,x))$ of A and $sub(1,y), \beta(sub(1,y))$, $sub(2,y)$, $\beta(sub(2,y))$ of B. Of course, we split each subregion into six subpixels. Next we encode two secret images $Image(\alpha)$ and $Image(\beta\alpha)$ into eight subregions $sub(3,x)$, $\beta(sub(3,x))$, $sub(4,x)$, $\beta(sub(4,\ x))$ of A and $sub(3,y)$, $\beta(sub(3,y))$, $sub(4,y)$, $\beta(sub(4,y))$ of B. Then, for example, we can reconstruct $Image(1)$ by stacking A and B, and $Image(\alpha)$ by stacking A and $\alpha(B)$. The contrast of this VCS is $1/24 = 1/(6 \cdot 2^2)$.

We can similarly construct a VCS with general dihedral group D_{2n} access structure as Example 2. We first consider the share A. We split every pixel of A into n^2 subregions. Then D_{2n} acts on the set of subregions of pixels in $\{\rho(x) \mid \rho \in D_{2n}\}$, where every orbit has length $2n$ and there are n^2 orbits (Figures 9, 10). We divide these n^2 orbits into n disjoints subsets, each of which contains n orbits, and label them $\{T(j) \mid 0 \le j \le n-1\}$ (Figure 10). For every α^k $(0 \le k \le n-1)$, we choose the n subregions of $T(k) \cap \alpha^k(x)$, and the n subregions of $T(k) \cap \beta\alpha^k(x)$ (Figure 10).

We next consider the share B. First split every pixel of B into n^2 subregions. For every $0 \le h \le n-1$, choose the n subregions from $\alpha^h(y)$ that are placed at the same positions as the following subregions of A.

$$\{\alpha^{-j}(sub(h,j,x)) \text{ in } A \mid \text{the } (j+1)\text{-th chosen subregion } sub(h,j,x) \text{ of } \alpha^{j+h}(x),\ \ j = 0,1,\ldots,n-1\}\ ,$$

where the indexes of $T(h+j)$ are expressed module $2n$.

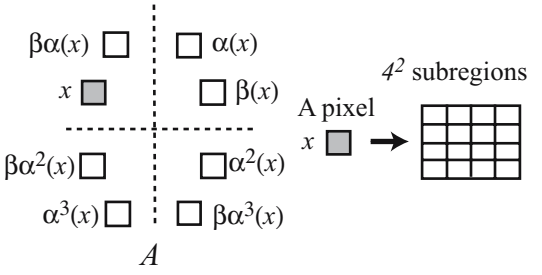


Fig. 9. A construction of VCS with D_8 access structure

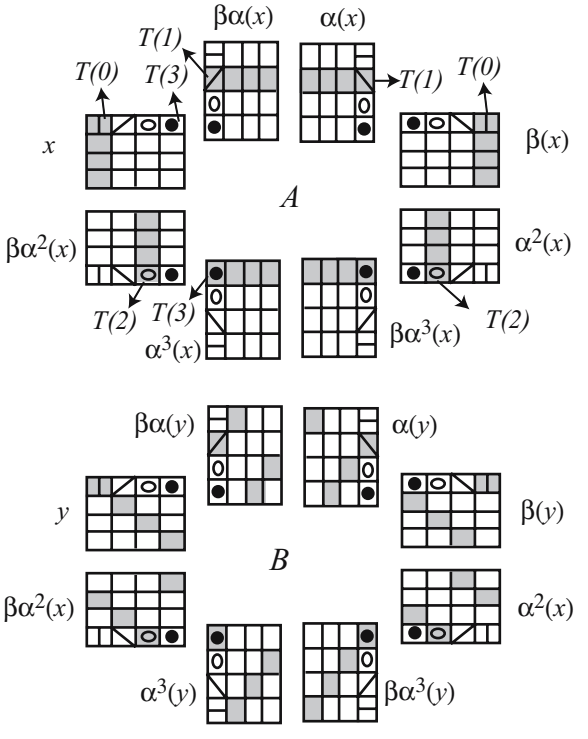


Fig. 10. A construction of VCS with D_8 access structure. Gray rectangles denote the chosen subregions of A and B.

Similarly, we next choose the n subregions from $\beta\alpha^h(y)$ that are placed at the same positions as the following subregions in A.

$$\{\alpha^{-j}(sub(h,j,x)) \text{ in } A \mid \text{the } (j+1)\text{-th chosen subregion } sub(h,j,x) \text{ of } \alpha^j\beta\alpha^h(x) = \beta\alpha^{h-j}(x),\ \ j = 0,1,\ldots,n-1\}\ .$$

For every $0 \leq k \leq n-1$, the two secret images $Image(\alpha^k)$ and $Image(\beta\alpha^k)$ corresponding to $(\alpha^k, \beta\alpha^k) \in \Omega$ are encoded into the above $(k+1)$-th chosen subregions of A and B by using the construction of VCS with reverse access structure given in Section 4. Namely, we can reconstruct $Image(\alpha^k)$ by stacking A and $\alpha^k(B)$, and $Image(\beta\alpha^k)$ by A and $\beta\alpha^k(B)$. The first reconstruction follows from the fact that when we stack A and $\alpha^k(B)$, for every $0 \leq h \leq n-1$, the $(k+1)$-th chosen subregion of $\alpha^{k+h}(x)$ of A is $sub(h, k, x)$, and the subregions of B corresponding this subregion is placed at the $(k+1)$-th subregion of $\alpha^h(y)$ in B. Hence they are matched in $A + \alpha^k(B)$. Similarly, the subregion of $\beta\alpha^h(x)$ of A and its corresponding subregion of B are matched.

The similar situation holds when we stack A and $\beta\alpha^k(B)$, and so we can reconstruct $Image(\beta\alpha^k)$ by staking A and $\beta\alpha^k(B)$.

Consequently, we can construct the desired perfect VCS with dihedral group D_{2n} access structure having contrast $1/(6n^2)$.

5 Conclusions

In this paper, we consider constructions of perfect VCS with dihedral group D_{2n} access structure for two shares and $2n$ secret images. We first give a construction by using orbits of a permutation group. Next we give a revised construction of VCS with reverse access structure, which is essentially different from the previous one. Then by using this new method, we give an improved construction of perfect VCS with dihedral group D_{2n} access structure, whose contrast is $1/(6n^2)$. It is difficult to find a construction with higher contrast, and so it might be possible to show that our new construction is best possible for some n.

References

1. S. Droste, New results on visual cryptography, *Advances in Cryptology-CRYPTO'96, LNCS*, **1109** (1996) 401–415.
2. M. Iwamoto and H. Yamamoto, A construction method of visual secret sharing schemes for plural secret images, *IEICE Trans. Fundamentals*, **E86-A** (2003) 2577-2588.
3. T. Kato and H. Imai, An extended construction method of visual secret sharing scheme, *IEICE Trans. Fundamentals (in Japanese)*, **J79-A** (1996) 1344-1351.
4. H. Koga, A general formula of the (t, n)-threshold visual secret sharing scheme, *Advances in cryptology–ASIACRYPT 2002*, LNCS **2501** (2002) 328–345.
5. M. Naor and A. Shamir, Visual cryptography, *Advances in Cryptology, Eurocrypt'94, LNCS*, **950** (1995) 1–12.
6. Y. Suga, K. Iwama, K. Sakurai and H. Imai, Extended graph-type visual secret sharing schemes with embedded plural images, *Inf. Process. Sco. Japan*, **42** (2001) 2106–2113.

Appendix A: An Example of Improved VCS with D_4 Access Structure

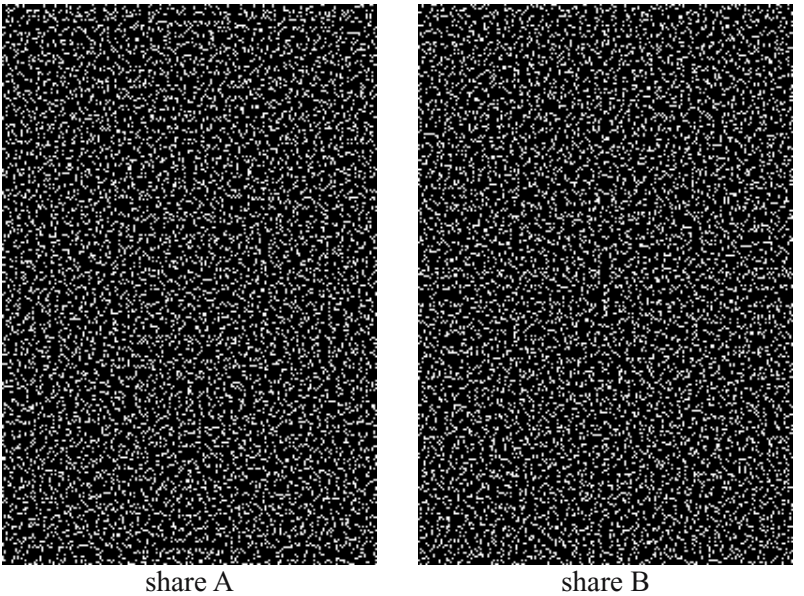


Fig. 11. An example of improved VCS with D_4 access structure ($shareA$ and $shareB$)

Appendix B: A Proof of Sharpness of Improved VCS with Reverse Access Structure

We prove that it is impossible to construct a perfect VCS with reverse access structure of contrast 1/5. It is easy to prove the non-existence of such a VCS with higher contrast in the same way. We shall use the same notation of Section 3. Let

$$A_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}) \text{ and } B_i = (y_{i1}, y_{i2}, y_{i3}, y_{i4}, y_{i5}), \quad i \in \{1, 2\}.$$

Then the colors of $A_i + B_i$ and $A_i + \tilde{B}_i$ are determined by $\{x_{ij} + y_{ij}\}$ and $\{x_{ij} + y_{i'j}\}$, respectively, where $\{i, i'\} = \{1, 2\}$ and $1 \leq j \leq 5$. Because of security, the sets $\{(x_{1j}, x_{2j}), 1 \leq j \leq 5\}$ and $\{(y_{1j}, y_{2j}), 1 \leq j \leq 5\}$ must consist of the same elements, respectively, for any colors of $\{A_i + B_i,\ A_i + \tilde{B}_i,\ i \in \{1, 2\}\}$. We consider the following two cases. Case 1 $(x_{1j}, x_{2j}) = (0, 0)$ for some j; and Case 2 neither (x_{1j}, x_{2j}) nor (y_{1j}, y_{2j}) is $(0, 0)$. Here we consider only Case 1 since Case 2 can be considered in a similar way. Without loss of generality, we may assume that $(x_{11}, x_{21}) = (0, 0)$.

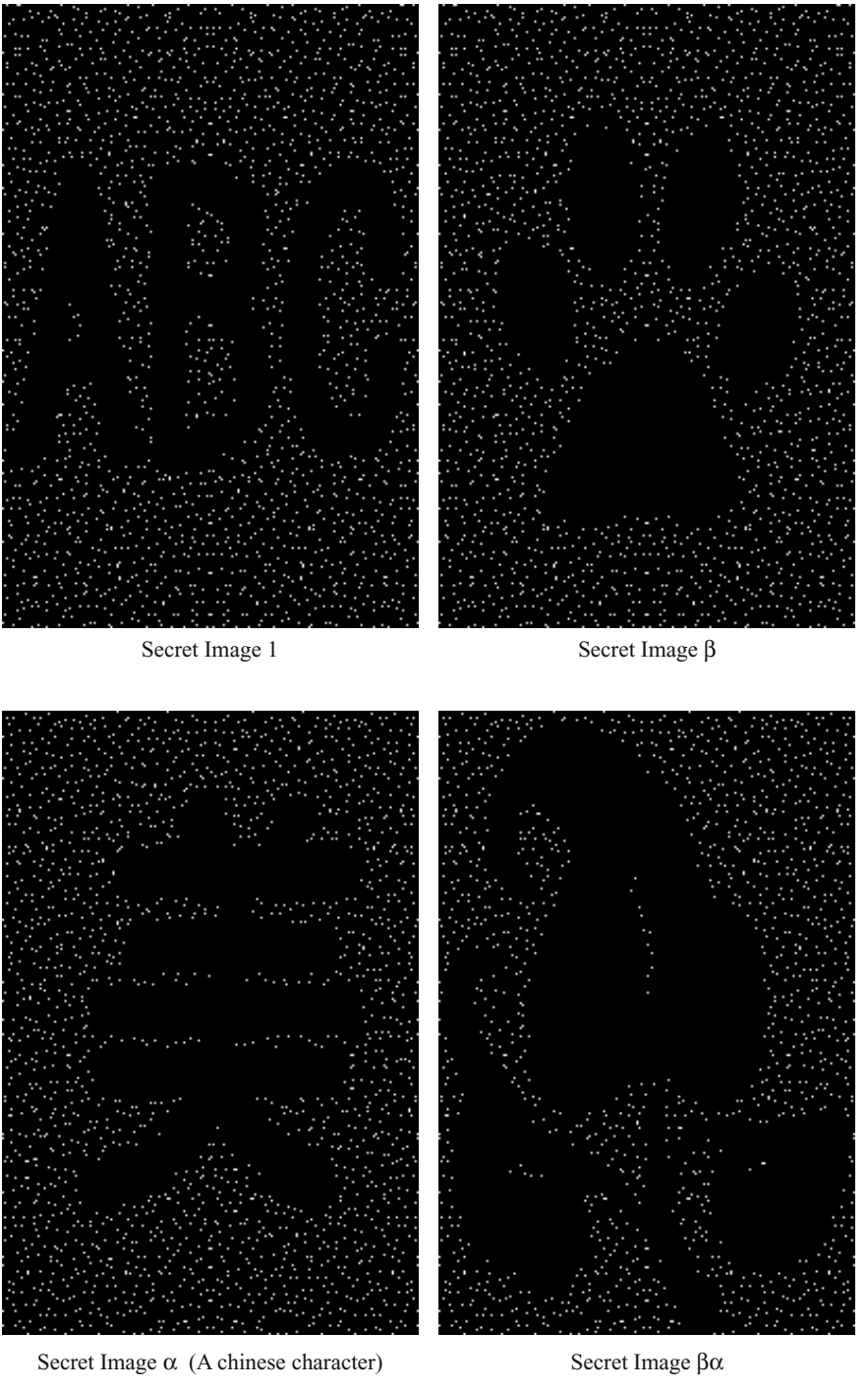


Fig. 12. An example of improved VCS with D_4 access structure (reconstructed images)

Consider the case that the colors of $(A_1 + B_1,\ A_2 + \tilde{B}_1,\ A_2 + B_2,\ A_1 + \tilde{B}_2)$ are $(0, 1, 0, 1)$, where $1 = black$ and $0 = white$. Then for some $a, b \in \{1, 2, 3, 4, 5\}$, $a \neq b$, we have

$$(x_{1a}x_{1b}) = (01),\ (y_{1a}y_{1b}) = (01),\ (x_{2a}x_{2b}) = (10),\ (y_{2a}y_{2b}) = (10).$$

Hence we may assume that $\{(x_{1j}x_{2j}), j = 1, 2, 3\} = \{(00), (01), (10)\}$. By considering the case that the colors of $(A_1 + B_1,\ A_2 + \tilde{B}_1,\ A_2 + B_2,\ A_1 + \tilde{B}_2)$ are $(1, 1, 1, 1)$, we have

$$(y_{11}y_{12}y_{13}y_{14}y_{15}) = (11101),\ (y_{21}y_{22}y_{23}y_{24}y_{25}) = (11110),$$

where $(y_{14}y_{15}) = (y_{1a}y_{1b}) = (01)$ and $(y_{24}y_{25}) = (y_{2a}y_{2b}) = (10)$. By considering the case that the colors of $(A_1 + B_1,\ A_2 + \tilde{B}_1,\ A_2 + B_2,\ A_1 + \tilde{B}_2)$ are $(0, 0, 0, 0)$, we have

$$(x_{14}x_{15}) = (00),\ (x_{24}x_{25}) = (00).$$

Hence we may assume $\{(x_{1j}, x_{2j}), 1 \leq j \leq 5\} \supset \{(00), (00), (01), (10)\}$. Finally again by considering the case that the colors of $(A_1 + B_1,\ A_2 + \tilde{B}_1,\ A_2 + B_2,\ A_1 + \tilde{B}_2)$ are $(1, 1, 1, 1)$, we have $\{(y_{1j}, y_{2j}), 1 \leq j \leq 5\} \supset \{(11), (11), (11), (11)\}$, which contradicts the above fact that $(y_{1a}y_{1b}) = (01)$ and $(y_{2a}y_{2b}) = (10)$. Consequently the statement is proved.

Author Index